JBuilder 2008

April 2008

Getting Started

Getting Started

Concepts

Concepts

Procedures

Reference

Getting Started

Getting Started

The topics in this section describe how to get started using your CodeGear product.

In This Section

Introducing JBuilder
Introduces your JBuilder product, which makes collaborative development fast and reliable for Java, open source and the web.

What's New
Presents an overview of new features in the JBuilder products on Eclipse and the JGear products.

Tour of the User Interface (UI)
Describes the JBuilder or JGears User Interface (UI).

Help on Help
Describes online Help and typographic conventions.

Migrating from Legacy Versions of JBuilder
This section provides information on migrating from legacy versions of JBuilder (JBuilder 2006, or earlier).

Introducing JBuilder

JBuilder 2008 is the first enterprise-class integrated development environment (IDE) built on the open-source Eclipse platform. It embraces and integrates the most popular "best of breed" plug-ins, tools, and frameworks.

The JBuilder product provides a certified and managed turn-key development solution on which organizations of any size can rely. It is designed to increase developer velocity while bringing balance and confidence to Java development through both commercial and open source components.

Your JBuilder 2008 product may include the following development solutions (depending upon the edition and version of the software that is installed):

- Provides the same collaborative capabilities, Optimizeit profiling, EJB design, and enterprise-class Rapid Application Development (RAD) features that the legacy JBuilder's reputation is built on, with improvements made possible by the Eclipse platform. You can migrate your legacy JBuilder projects to the JBuilder on Eclipse platform, as well as adding other Eclipse plugins that you are currently using.
- Installs pre-configured versions of several of the most popular runtime servers, including: JBoss, Apache Tomcat, Geronimo, and Glassfish to get you developing Web Applications, Web Services, and EJBs faster. Yet JBuilder is also compatible with Borland Application Server, WebLogic, WebSphere, Oracle and other products that you may have purchased separately. In addition, the Web Services Explorer helps you implement your WSDL, WSIL, and UDDI files with ease.
- Includes a Modeling Perspective that delivers Borland's Together — an innovative and highly productive visual "drag and drop" environment with enterprise-class project management capabilities designed to increase the speed and productivity of individuals and development teams
- Provides the ability to collaborate on development projects and integrate open source and commercial development in a single, managed environment through the ProjectAssist/TeamInsight features and components.
- Increases the development velocity of Java teams and individuals via ProjectAssist's Requirements Tracking, Bug Tracking, Source Code Management, and Project Management.
- Decreases database application development time with developer versions of CodeGear's InterBase and Blackfish SQL for Java applications.
- Increases the speed of development and reduces the learning curve through the use of the Application Factory feature and functionality to create application modules for rapid coding, deployment and reuse of applications.

For more detailed information about these new features, see What's New, or go directly to the overview for each of the features described above. If you wait to get started coding, you can go to the Tasks overview links page and link to your favorite topics and tasks.

Getting Started

Review the following topics to become familiar with JBuilder 2008 features and tools:

Introducing JBuilder	This topic provides a high level overview of JBuilder 2008, introduces important features and concepts, and introduces available help resources.
What's New in JBuilder	This topic provides a high level overview of JBuilder 2008, with information about migrating to JBuilder 2008 from a legacy JBuilder release.
Using the Eclipse Help System	This topic provides information about Eclipse platform functionality.

Using Online Help

Use the Online Help system to find conceptual, procedural, and reference information about CodeGear's JBuilder 2008 plug-in for Eclipse. Where appropriate, this Help system provides references to online help provided by Eclipse, as well as to online help provided by third parties and other plug-ins that are shipped with JBuilder 2008.

Product and Plug-in Versions

This version of JBuilder includes many other third-party plug-ins that you can use with your existing Eclipse implementations. You can also add other Eclipse plug-ins to this version of Eclipse with JBuilder. For a detailed list of the included products and plug-ins, choose the **Contents** section of the **Welcome to CodeGear JBuilder** page.

You can also see a list of all installed features and plug-ins with your CodeGear product configuration by selecting **Help** ▶ **About JBuilder** and selecting the appropriate **Feature Details**, **Plug-in Details**, or **Configuration Details** buttons.

Cheat Sheets

You can also view Cheat Sheets through the Help menu. Cheat Sheets walk you through specific tasks and open the wizards associated with these tasks to give you easy-access to the components you need. View the Cheat Sheets by selecting **Help** ▶ **Cheat Sheets** and further selecting the desired cheat sheet.

Video Demos

Narrated videos of specific JBuilder features, such as importing projects, and developing web services are also available at the
CodeGear Developer Network TV
.

Developer Support and Resources

CodeGear provides a variety of support options and information resources to help developers get the most out of CodeGear products. These options include a range of CodeGear Technical Support programs, as well as free services on the Internet, where you can search our extensive information base and connect with other users of CodeGear products. Other product information is available on
the JBuilder product web site.
Access the CodeGear Developers Network (CDN) using the link below.

Related Concepts

- Getting Started
- What's New
- Help on Help
- Legacy JBuilder 2006/JBuilder on Eclipse Differences
- Legacy JBuilder Project Migration Overview
- Application Factory Overview
- Java EE Applications Overview
- Enterprise Java Bean (EJB) Applications Overview
- Web Services Overview
- Web Applications Overview
- Modeling Applications Overview
- The Web Tools Project (WTP) in JBuilder
- ProjectAssist and TeamInsight Overview

Related Reference

- Using the Eclipse Help System
- CodeGear Developers Network

What's New

This topic describes the new features and functions of JBuilder.

New in This Release

New functionality and features in this release includes:

- Application Factories
- Swing Designer
- Eclipse 3.3 (Europa)
- Web Tools Platform (WTP) 2.0
- Struts 1.x
- Enhanced EJB Modeler
- Updated Application Server support
- TeamInsight support for StarTeam and OpenAPI
- Updated Optimizeit
- Updated InterBase
- Updated BlackfishSQL (formerly JDataStore)
- TPTP ProbeKit and CPU Root Filter
- Single Binary delivery

Application Factory

The Application Factory functionality introduces an application-driven development paradigm, where the structure, evolution, and logic behind the development of the application are checked into version control along with the source code for the application itself.

The Application Factory functionality provides tools to:

- Organize code visually
- Track changes
- Associate changes to actions
- Data mine actions from the past
- Associate all project artifacts in the context of the desired user story or task.

Application Factory allows the user to store application-specific information along with the application in an Application Module. An Application Module is a set of Application Projects associated with an application, in combination with the metadata project for the application. Each workspace can contain only one metadata project. The project that accompanies an application includes:

- Tags—keywords associated with a piece of information
- Application Diagram—visual representation of application architecture and functionality
- Scripts—code generating/templating mechanism providing a way to generate template code
- Readme—overview of application functionality
- Cheat Sheet—cheat sheet providing important steps for using the application and scripts

Refer to the Application Factory concepts, procedural and dialog reference topics at:

- Links to Application Factory Concept Topics
- Links to Application Factory Task Topics
- Links to Application Factory Perspectives Reference Topics
- Links to Application Factory Wizards and Dialogs Reference Topics

Swing Designer

A new visual Swing Designer, from Instantiations Inc., is now bundled with JBuilder. This designer provides significant improvements over the Eclipse VE designer. To invoke the designer on a new class, select File | New | Other | Designer, and select the child node, such as JFrame or JDialog, for the new class you want to visually design.

Ability to Migrate Your Projects from Legacy Versions of JBuilder

The Java perspective contains a code editor, a **Package Explorer** that is similar to the legacy JBuilder (2006 and before) **Project** pane, an **Outline** view that is similar to the legacy JBuilder **Structure** pane, and a tabbed lower pane, for searching and error display, that is similar to the legacy JBuilder **Message** pane. There is also a **Debug** perspective, a **Java Browsing** perspective, and a **Java Type Hierarchy** perspective that are similar to panes in the legacy JBuilder IDEs.

You can import any type of JBuilder project created with a legacy version of JBuilder into the JBuilder 2007 or later/ Eclipse workspace, using one of the Import JBuilder Project wizards (Java or Java EE). Java EE conversion includes conversion from XML descriptors to XDoclet annotations (EJB 2.1) or to EJB 3.0 annotations.

Enhanced EJB Modeler

JBuilder's EJB Modeler is an enhanced version of legacy JBuilder EJB designers. It includes support for EJB 2.x and 3.0 features, and support for JSR 181.

Server Runtime Installation for Web Applications, Web Services, and EJBs

JBuilder 2008 allows you to install pre-configured versions of several of the most popular runtime servers, including:

- Apache Geronimo 1.1.1 and 2.0
- Apache Tomcat 5.5 and 6.0.14
- JBoss 3.2.3, 4.0.5, and 4.2.2
- Sun GlassFish 1.1 and 2.0

Tip: A best practice when using the runtime servers is to choose one of the versions that has (CodeGear or Borland) after it. These versions have been extended to support specific JBuilder features.

JBuilder 2008 is also compatible with:

- Borland Application Server 6.7
- IBM WebSphere 6.0 and 6.1 (with EJB3 feature pack)
- BEA WebLogic Application Server 8.1, 9.2, and 10.0
- Oracle Application Server 10.1.3.2

- Oracle Containers for Java (OC4J) 10.1.3.2

In addition, the Web Services Designer helps you implement your WSDL, WSIL, and UDDI files with ease.

Developer Versions of InterBase and Blackfish SQL for Java

JBuilder includes two database systems: InterBase and Blackfish SQL for Java (formerly known as JDataStore).

You can create a visual model of your database application using the JBuilder Modeling Perspective.

ProjectAssist and TeamInsight

As a product, JBuilder functionality is now split into two separate installs. One is ProjectAssist, and the other is the JBuilder IDE. Previously, ProjectAssist functionality was available in the JBuilder IDE; now only TeamInsight is available in the JBuilder IDE. TeamInsight is also available in ProjectAssist. You can now install ProjectAssist on the appropriate machines to manage your servers, and install the JBuilder IDE on the appropriate machines for developers.

For writing custom ProjectAssist plug-ins, a partner Developer Guide now exists. Additionally, the ProjectAssist API JavaDoc is provided. To browse, from inside Project Assist, select Help | Help Contents, then select the ProjectAssist – TeamInsight Developer Guide.

Designed to help organizations manage and balance complex development projects across teams and locations and across open source and proprietary software, ProjectAssist and TeamInsight provides a blended development "stack in the box" that contains:

- Requirements Tracking
- Bug Tracking
- Source Code Management
- Project Management across organizations and time zones

To accomplish this, JBuilder 2008 embraces and integrates the most popular "best of breed" open source plug-ins, tools and frameworks and provides a certified and managed turn-key development solution on which organizations of any size can rely.

This team-coordination feature is based on two user types, the Administrator (who performs the ProjectAssist install and server configuration outside of the JBuilder install) and the User (who uses the installed TeamInsight client tools to develop or test software with the integrated products listed above).

Single Binary Delivery

JBuilder is now delivered as a single binary, regardless of SKU. Previously, different SKUs had different binaries. Users can now upgrade their JBuilder, for example, from Turbo to Professional, simply by entering a new license key. There is no need to do a new install to upgrade.

3rd-Party Plug-ins

This version of JBuilder includes many other third-party plug-ins that you can use with your existing Eclipse implementations. You can also add your other Eclipse plug-ins to this JBuilder on Eclipse version. For a full list of the included products and plug-ins, choose the **Contents** section of the **Welcome to CodeGear <productname>** page.

You can also see a list of all installed features and plug-ins with your CodeGear product by selecting **Help ▸ About <productname>** and selecting the appropriate **Feature Details**, **Plug-in Details**, or **Configuration Details** buttons.

Related Concepts

Help on Help
Tour of the User Interface (UI)
Migrating from Legacy Versions of JBuilder
Application Factory Concepts
Java EE Applications Overview
Enterprise Java Bean (EJB) Applications Overview
Web Applications Overview
Web Services Overview
Modeling Applications Overview
ProjectAssist and TeamInsight Overview
Peer to Peer Collaboration

Tour of the User Interface (UI)

This topic introduces the following JBuilder 2008 UI subjects:

- Eclipse-specific UI elements
- JBuilder Compatibility
- Peer to Peer Collaboration
- ProjectAssist/TeamInsight
- Application Factory

Using the Eclipse Workbench

The JBuilder 2008 user interface is integrated into the Eclipse Workbench. To become familiar with the JBuilder 2008 UI, see the Eclipse help topic in the "Workbench User Guide".
Eclipse Help Topic: "Workbench"
, which discusses the following user interface elements:

- Resources
- Resource Hierarchies
- Linked Resources
- Path Variables
- Working Sets
- Builds
- Local History

In addition to reviewing the primary workbench interfaces, see the following Workbench Help subtopics:

- Perspectives (
  Eclipse Help Topic: "Perspectives"
)
- Editors (
  Eclipse Help Topic: "Editors"
)
- Views (Eclipse Help Topic: "Views")
- Toolbars (Eclipse Help Topic: "Toolbars")
- Markers (Eclipse Help Topic: "Markers")
- Bookmarks (Eclipse Help Topic: "Bookmarks")
- Label decorations (Eclipse Help Topic: "Label Decorations")
- External tools (Eclipse Help Topic: "External tools")
- Accessibility features in Eclipse (
  Eclipse Help Topic: "Accessibility features"
)

In addition to the above workbench information, review the Tasks section from the Eclipse *"Workbench User Guide"* for information on performing specific tasks in the workbench.

Tip: The Eclipse Workbench and Perspective views will be new to legacy JBuilder users. Explore the suggested Eclipse help topics to become familiar with these key user interface elements in Eclipse.

JBuilder Compatibility

Legacy JBuilder projects (prior to JBuilder 2007) are easily converted into projects with the **Project Import Wizard**. Review the following topics to become familiar with compatibility features:

- Legacy JBuilder 2006/JBuilder on Eclipse Differences
- Legacy JBuilder Project Migration Overview
- Importing a Java EE Project from Legacy JBuilder
- Building an Imported Project

Peer to Peer Collaboration

Peer to peer collaboration is easily available through the **Peers** view in the workbench. Review the following topics to become familiar with Peer to Peer collaboration features:

- Peer to Peer Collaboration
- Tasks for Peer to Peer Collaboration

ProjectAssist and TeamInsight

The ProjectAssist and TeamInsight open-source development tools provide project management and coordination tools for development groups. ProjectAssist lets the Administrator install and configure the TeamInsight Tools for the development team. ProjectAssist is provided as a separate installation and configuration entity from JBuilder. This allows the ProjectAssist Administrator to install and configure the TeamInsight tools and users without requiring a complete install of the JBuilder product on the Administrator's computer.

Review the following topics to become familiar with ProjectAssist and TeamInsight:

- ProjectAssist Component Help
- ProjectAssist and TeamInsight Overview
- TeamInsight Procedures

Application Factory

Several new innovations to the workbench and user interface have been made to accommodate the Application Factory functionality. Refer to the concept topic Workbench Features of Application Factory .

Related Concepts

ProjectAssist and TeamInsight Overview
Application Factory Concepts
Workbench Features of Application Factory
Legacy JBuilder 2006/JBuilder on Eclipse Differences
Legacy JBuilder Project Migration Overview
Importing Legacy Projects
Peer to Peer Collaboration
ProjectAssist and TeamInsight Overview

Related Tasks

JBuilder Project Migration
Peer to Peer Collaboration
Using Application Factory
Building an Imported Project
Enabling Peer to Peer Collaboration
TeamInsight Procedures

Related Reference

Eclipse Help Topic: “Workbench”
Eclipse Help Topic: “Perspectives”
Eclipse Help Topic: “Working with Perspectives”
Eclipse Help Topic: “Working with editors and views”

Help on Help

You can access multiple Help resources to find information about JBuilder 2008.

Online Help

Online Help provides detailed information about the features available in your JBuilder or JGear product. To view the HelpTable of Contents for your product, choose the menu path **Help ▸ Help Contents**, then JBuilder 2008 or JGear product name in the left-side column. To locate help topics via searching, enter a search term into the **Search** field. You can narrow the search scope to selected topics by clicking the **Search Scope** button.

Online Help uses a multi-tiered, top-down approach to help you become familiar with the tools and features of JBuilder 2008. When you open online Help; conceptual, task, and Wizards and Dialogs Reference information is available. Conceptual information gives you access to general overview information. Task information provides step-by-step instructions to perform many of the tasks described at the conceptual level. Reference information includes topics on the wizards and dialog boxes. For additional help resources, see the Release Notes (Readme) that accompanies the product.

Refer to the following list to determine the type of online Help information that specifically addresses your needs:

- If you are new to JBuilder 2008 or just want a product overview, see the Getting Started section.
- To learn about migrating your legacy JBuilder projects into the Eclipse workspace, see the Migrating from Legacy Versions of JBuilder conceptual topics and JBuilder Project Migration task topics.
- For information about the managing your projects with the TeamInsight features, see ProjectAssist and TeamInsight Concepts for conceptual topics and TeamInsight Procedures for task topics. For detailed ProjectAssist procedures, see the separate ProjectAssist component help at ProjectAssist Component Help.

Cheat Sheets

You can also use Cheat Sheets, located in the Help menu to follow step-by-step procedures that allow you to open wizards and dialog boxes as you perform specific tasks. All of the Eclipse Cheet Sheets are found under the **Help ▸ Cheat Sheets**. Select **JBuilder | JGear** folder under this path for cheat sheets specific to your JBuilder or JGear product.

Videos

Videos and our video tutorials are another way to learn more about your JBuilder or JGear product. Narrated videos of specific JBuilder features, such as importing projects, and developing web services are also available at the CodeGear Developer Network TV
.

Typographic Conventions

The following typographic conventions are used throughout this JBuilder or JGear online Help.

Typographic conventions

Convention	Used to indicate
`Monospace type`	Source code and text that you must type, file names, and directories.
Boldface	References to windows, dialog boxes and tools.
Italics	Book titles and new terms.

KEYCAPS	Keyboard keys, for example, the CTRL or ENTER key.

Migrating from Legacy Versions of JBuilder

This CodeGear product provides a migration path from legacy versions of JBuilder (JBuilder 2006, or earlier) to your CodeGear product on Eclipse. This allows you to develop, test, and run your previously-created JBuilder projects in the Eclipse workspace.

In This Section

Legacy JBuilder Project Migration Overview
Describes the migration path from previous versions of JBuilder.

Legacy JBuilder 2006/JBuilder on Eclipse Differences
Summarizes differences between Legacy JBuilder 2006 and JBuilder on Eclipse development environments

Legacy JBuilder/Eclipse Dialog Box Equivalents
Summarizes some legacy JBuilder IDE and Eclipse dialog box equivalents.

Legacy JBuilder/Eclipse Menu Command and Keyboard Equivalents
Summarizes some legacy JBuilder and Eclipse (or JBuilder on Eclipse) menu commands and keyboard equivalents.

Project Properties
Describes project properties

Project Nodes
Describes project nodes

Run Configuration
Describes JBuilder 2007 run configurations

Source Control
Describes JBuilder 2007 source control options

Importing Legacy Projects
Describes various project import scenarios from a legacy version of JBuilder into your CodeGear product. on Eclipse.

Legacy JBuilder Project Migration Overview

You can import any type of Java project created in a legacy release of JBuilder (JBuilder 2006, or before) into your CodeGear product Eclipse workspace, including:

- Java SE (formerly J2SE) projects
- Java EE (formerly J2EE) projects
- VisiBroker projects
- RMI/JNI projects

The project import wizard does not copy the JBuilder source files and folders directly into the workspace; instead links are created using the Eclipse resource link capability. The Eclipse project file in the workspace maps a resource name, for example, `/src`, to an absolute path name, for example, `C:/MyProject/src/java`. New files that are added to the project are added to the original source folder. Projects under source control may be able to check it into the JBuilder on Eclipse workspace.

Created Workspace Files and Folders

The following files and folders are created in the workspace folder for a project imported from a legacy version of JBuilder:

- `.classpath`: The linked resources file (XML source).
- `.project`: The Java project file (XML source).
- `/bin`: The output folder.

Warning: If the **Enabled linked resources** option on the **Linked Resources** page of the **Preferences** dialog box (**Window ▸ Preferences ▸ General ▸ Workspace ▸ Linked Resources**) is not checked, the project import may fail. If this happens, the following message will be displayed in the **Import Status** dialog box: `Error creating source path link for <project name>. Linked resources are not supported by this application.`

The Eclipse build process uses the standard JDK compiler, not the legacy JBuilder compiler, Borland Make for Java. Before you build your imported project, you can check compiler options on the **Java Compiler** page of the **Properties** dialog box.

Using the Import Project Wizard

The import project wizard can translate Java EE and Java SE legacy JBuilder projects to JBuilder on Eclipse projects. Following are the notable differences in projects:

- Splitting of the legacy module into multiple project types
- Server configuration for compiling, running and deploying
- XML descriptors are converted to XDoclet annotations

When a project is converted, the Project import log displays conversion information including data regarding any artifacts that were not created in the conversion process. Not all Java EE artifacts are converted. Currently EJB Web EAR and EJB web services client projects are supported. Application client and JBoss service archives must be converted manually.

Projects with Generated Source

When a project, such as a VisiBroker or RMI project, has auto-generated sources that are output to the `/Generated Sources` folder in the `classes` folder, the `/Generated Sources` folder is not imported. However, when you build the project the source files are automatically generated and placed in a `/Generated Sources` folder in the Eclipse workspace. The **Derived** setting on the **Info** page of the **Properties** dialog box (**Properties** ▸ **Info** from the context menu in the **Package Explorer** with the folder selected) indicates that this folder is auto-generated.

Unsupported Properties

Some project properties are not supported in Eclipse or are translated to the Eclipse equivalent on import. The following table illustrates those items:

Project Item	Description
/Additional Settings Folder	Not imported; no equivalent.
/doc Folder	Not imported. Regenerate with **File** ▸ **Export** ▸ **Javadoc**.
/bak Folder	Not imported.
jbInit() Method	Left in code.
@todo Tags	Left in code.

Related Concepts

- Legacy JBuilder 2006/JBuilder on Eclipse Differences
- Legacy JBuilder/Eclipse Dialog Box Equivalents
- Legacy JBuilder/Eclipse Menu Command and Keyboard Equivalents
- Project Properties
- Project Nodes
- Run Configuration
- Source Control
- Importing Legacy Projects

Related Tasks

- Importing a Java EE Project From Legacy JBuilder
- Importing a Java SE Project From Legacy JBuilder

Importing Legacy Projects

You can import the following types of Java projects created with a legacy version of JBuilder (for example, JBuilder 2006) into your CodeGear product on Eclipse.

- Java EE (Note that Java EE projects cannot be imported through the JGear LiveSource product)
- Java SE
- VisiBroker
- RMI/JNI

Java EE Project

The project import wizard creates a project for each module from legacy JBuilder project modules, with shared source code. The modules table lists the Java EE modules found in the legacy project including Java versions and the corresponding JBuilder 2008 project created during the conversion process. The module table displays Java and Java EE versions (project facets) for JBuilder 2008 project.

Note: Java EE projects cannot be imported through the JGear LiveSource product.

EJB Projects

JBuilder 2008 supports **EJB 2.x** development using **XDoclet** annotations and **EJB 3.0** development using **Java EE 5.0** annotations. The project import wizard converts legacy JBuilder project **XML** descriptors to either **XDoclet** annotations (for **EJB 2.1**) or to **Java EE 5.0** annotations (for **EJB 3.0**).

Note: For EJB 2.1 interfaces that are not copied to the EJB project (or the utility project) from the legacy JBuilder project (interfaces are generated using XDoclet).

Java SE Project

Use the import wizard to import legacy JBuilder home directory files and libraries as required to properly import a Java SE project. The following legacy artifacts are imported in the project conversion:

- Libraries
- Runtime Configurations
- Javadoc Options
- Java Compiler Options default file encoding
- Java files

Java RMI/JNI Project

The `java.rmi` package provides classes for Java Remote Method Invocation (RMI). Using RMI enables the creation of distributed Java-to-Java applications where the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. RMI uses object serialization to marshal and un-marshal parameters and does not truncate types, supporting true object-oriented polymorphism.

Build the RMI/JNI project in JBuilder to expand any build macros that used as VM arguments. Expand the project in the **Package Explorer** and select an RMI or JNI file. Use the **Properties** dialog to display the **Properties for <filename>** dialog box. Use the **RMI/JNI Properties** page to view property settings imported from JBuilder.

VisiBroker Project

Open the **VisiBroker** page of the **Preferences** dialog box to verify the directory where VisiBroker tools are installed. Expand the project in the **Package Explorer** and select an IDL file or a Java interface file that will be translated from IDL, to IDL, or to IIOP. Use **Properties** to display the **Properties for <filename>** dialog box and confirm Property settings have been imported.

- For IDL to Java files, choose the **VisiBroker IDL Properties** page and verify options in the **IDL2Java Settings** area of the dialog box.
- For Java to IDL files, choose the **VisiBroker Java Properties** page and verify options in the **Java2IDL Settings** area of the dialog box.
- For Java to IIOP files, choose the **VisiBroker Java Properties** page and verify options in the **Java2IIOP Settings** area of the dialog box.

Related Concepts

Legacy JBuilder Project Migration Overview
Legacy JBuilder 2006/JBuilder on Eclipse Differences

Related Tasks

Importing a Java EE Project From Legacy JBuilder
Importing a Java SE Project From Legacy JBuilder
Creating a Java EE Project
JBuilder Project Migration

Legacy JBuilder 2006/JBuilder on Eclipse Differences

The Eclipse platform, modeled as a plug-in development environment, provides an end-to-end Java development platform. Plug-ins help create an adaptable and extensible system. The Eclipse environment provides perspectives, editors, and views that can be added to, configured, or replaced through the implementation of plug-ins.

The JBuilder plug-in for Eclipse adds views and editors to the existing Eclipse Java perspective, as well as providing a modeling perspective and an integrated set of development life-cycle management tools.

Perspectives

An Eclipse perspective provides a “flavor” for the Eclipse development environment and defines the initial set and layout of views in the Workbench. Each perspective provides a set of functionality aimed at accomplishing a specific type of task. As you work in the Workbench, you will probably switch perspectives frequently. Perspectives are available from the **Window** ▸ **Open Perspective** menu command. You can set perspective preferences with the **Window** ▸ **Preferences** ▸ **General** ▸ **Perspectives** command.

The Java perspective contains a code editor, a **Package Explorer** that is similar to the previous JBuilder **Project** pane, an **Outline** view that is similar to the previous JBuilder **Structure** pane, and a tabbed lower pane, for searching and error display, that is similar to the previous JBuilder **Message** pane. There is also a **Debug** perspective, a **Java Browsing** perspective, and a **Java Type Hierarchy** perspective that are similar to panes in the previous JBuilder IDEs.

JBuilder 2008 adds views and editors to the Java perspective that are specific for developer needs, such as tools for editing code, viewing and editing requirements and change requests, profiling, and creating unit tests. JBuilder 2008 also adds a Modeling perspective so that you can do most of these tasks while looking at a modeling view of your Java code. You can customize these perspectives with the **Window** ▸ **Customize Perspective** command.

Editors

Most Eclipse perspectives contain one or more editors for editing code. Eclipse editors include a Java source code editor, a text editor, and a GUI visual editor. JBuilder 2008 includes the modeling designer, requirements editor, and a change request editor. You can open as many editors as you wish, though only one editor is active at a time. The main menu and toolbar only contain items applicable to the active editor.

Views

Views provide alternative presentations of data. Views have unique context menus and may have unique toolbars. A view can be displayed on its own, or as a tabbed page in a multi-view presentation. JBuilder 2008 provides multiple views, including the **Modeling** Perspective, the **Requirements** view, and the **Profiling** view.

Note: Each view contains a toolbar with a drop-down menu icon.

Tips

There are many slight differences between the legacy JBuilder IDEs and the Eclipse user interface. The tips below can help you learn to navigate the Eclipse Java perspective quickly. Note that these tips are not extensive or exhaustive, and cover just some of the frequently used features.

Feature	Tip
Editor	If a Java file has errors, a red “X” icon is displayed in the left margin of the editor. Hovering the mouse over the icon displays the error as a tooltip.

Editor	When the editor cannot display tabs for all open files due to space constraints, the number of files not displayed is shown on a toolbar button. Click the button to see a file list.
Editor	When using Code Assist (code insight in JBuilder 2006), a tooltip with available Javadoc is displayed.
Editor	The **Navigate** ▶ **Open Type Hierarchy** command displays the type hierarchy of a specific source code element.
Editor	Hovering the mouse over a symbol displays Javadoc for that symbol, if available.
Editor	Clicking the mouse on an identifier marks all uses of that identifier in the current file. Locations where the identifier is used are marked in the gutter.
Editor	Typing a left-facing parenthesis, brace, or quote automatically adds the termination/closing mark.
Editor	Placing a caret in a symbol highlights all of its occurrences in the open file.
Editor	The gutter indicates lines of code that have changed.
Editor	Using the **Navigate** menu, you can search for references by a range of scopes, from the workspace to the current project to the current class hierarchy to just a selected group of files.
Editor	Previous searches are available from a drop-down menu in the **Search** view.
Editor	You can use the **Java Search** page of the **Search** dialog box (**Search** ▶ **Java**) to search for the particular usage of a symbol.
Editor	Use the **Change Method Signature** refactoring to modify the signature of a method.
Editor	The Javadoc author name field is automatically filled in when creating a new class.
Editor	Optimize imports and code formatting can be applied to a group of files.
Editor	You can search for references on a selected import statement (**Search** ▶ **References**).
Debugger	When a change is made, saved, and compiled during a debugging session, obsolete frames are automatically popped off the stack and the frame pointer is automatically set to the highest possible valid frame.
Debugger	To evaluate an expression, first execute the code (**Run** ▶ **Execute**), then display the results (**Run** ▶ **Display**).
Debugger	Only one Eclipse instance can be debugged at a time.
Debugger	Icons in the **Variables** view indicate the type of variable, for example, members or local variables.
Debugger	In the **Debug** perspective, right-click an application and choose **Terminate All** to remove all terminated launches.
Debugger	Breakpoints can be configured to stop when a condition changes, not just on a true/false condition. Breakpoints can also be configured to stop only in a particular thread.

Related Concepts

Legacy JBuilder Project Migration Overview
Legacy JBuilder/Eclipse Menu Command and Keyboard Equivalents
Legacy JBuilder/Eclipse Dialog Box Equivalents

Related Tasks

Importing Legacy Projects

Related Reference

Perspectives in the Eclipse Workbench User Guide
Editors in the Eclipse Workbench User Guide
Views in the Eclipse Workbench User Guide

Legacy JBuilder/Eclipse Dialog Box Equivalents

The following tables show the legacy JBuilder IDE (JBuilder 2006, or earlier) and Eclipse (or JBuilder on Eclipse) dialog box equivalents.

Project Properties dialog box

Legacy JBuilder	Eclipse
Paths page (Project ▸ Project Properties ▸ Paths)	**Java Build Path** page (Project ▸ Properties ▸ Java Build Path)
JDK option on the **Paths** page (Project ▸ Project Properties ▸ Paths)	**Libraries** tab on the **Java Build Path** page (Project ▸ Properties ▸ Java Build Path)
Output path on the **Paths** page (Project ▸ Project Properties ▸ Paths)	**Output Path** option on the **Java Build Path** page (Project ▸ Properties ▸ Java Build Path)
Source tab on the **Paths** page (Project ▸ Project Properties ▸ Paths)	**Source** tab on the **Java Build Path** page (Project ▸ Properties ▸ Java Build Path)
Documentation tab on the **Paths** page (Project ▸ Project Properties ▸ Paths)	**Javadoc Location** page (Project ▸ Properties ▸ Java Build Path)
Required Libraries tab on the **Paths** page (Project ▸ Project Properties ▸ Paths)	**Libraries** tab on the **Java Build Path** page (Project ▸ Properties ▸ Java Build Path)
Compiler options on the **Java** page (Project ▸ Project Properties ▸ Build ▸ Java)	**Compiler** page in the **Preferences** dialog box (Window ▸ Preferences ▸ Java ▸ Compiler)

Tools menu

Legacy JBuilder	Eclipse
Configure Libraries dialog box (Tools ▸ Configure ▸ Libraries)	**User Libraries** page in the **Preferences** dialog box (Window ▸ Preferences ▸ Java ▸ Build Path ▸ User Libraries)
Configure JDKs dialog box (Tools ▸ Configure ▸ JDKs)	**Installed JREs** page in the **Preferences** dialog box (Window ▸ Preferences ▸ Java ▸ Installed JREs)
Browser page in the **Preferences** dialog box (Tools ▸ Preferences ▸ Browser)	**Web Browser** page in the **Preferences** dialog box (Window ▸ Preferences ▸ General ▸ Web Browser)
Editor page in the **Preferences** dialog box (Tools ▸ Preferences ▸ Editor)	**Web Browser** page in the **Preferences** dialog box (Window ▸ Preferences ▸ Java ▸ Editor)
Documentation tab on the **Paths** page (Project ▸ Project Properties ▸ Paths)	**Javadoc Location** page (Project ▸ Properties ▸ Java Build Path)
Required Libraries tab on the **Paths** page (Project ▸ Project Properties ▸ Paths)	**Libraries** tab on the **Java Build Path** page (Project ▸ Properties ▸ Java Build Path)

Legacy JBuilder/Eclipse Menu Command and Keyboard Equivalents

The following tables show legacy JBuilder (JBuilder 2006, or earlier) and Eclipse (or JBuilder on Eclipse) menu command and keyboard equivalents.

Note: If no keyboard shortcut is listed, none is available.

File menu

Legacy JBuilder	Eclipse
File ▸ New (CTRL+N)	File ▸ New (CTRL+N)
File ▸ Open File (CTRL+O)	File ▸ Open File
File ▸ Close (CTRL+F4)	File ▸ Close (CTRL+F4)
File ▸ Close All (CTRL+SHIFT+F4)	File ▸ Close All (CTRL+SHIFT+F4)
File ▸ Save (CTRL+S)	File ▸ Save (CTRL+S)
File ▸ Save All (CTRL+SHIFT+S)	File ▸ Save All (CTRL+SHIFT+S)

Edit menu

Legacy JBuilder	Eclipse
Edit ▸ Undo (CTRL+Z)	Edit ▸ Undo (CTRL+Z)
Edit ▸ Redo (CTRL+SHIFT+Z)	Edit ▸ Redo (CTRL+Y)
Edit ▸ Cut (CTRL+X)	Edit ▸ Cut (CTRL+X)
Edit ▸ Copy (CTRL+C)	Edit ▸ Copy (CTRL+C)
Edit ▸ Paste (CTRL+V)	Edit ▸ Paste (CTRL+V)
Edit ▸ Format All (ALT+SHIFT+K)	Source ▸ Format (CTRL+SHIFT+F)
Edit ▸ Code Insight (CTRL+SPACE)	Edit ▸ Content Assist (CTRL+SPACE)
Edit ▸ Code Insight ▸ ParameterInsight (CTRL+SHIFT+SPACE)	Edit ▸ Parameter Hints (CTRL+SHIFT+SPACE)
Edit ▸ Code Insight ▸ Javadoc QuickHelp (CTRL+Q)	Navigate ▸ Open External Javadoc (SHIFT+F2)
Edit ▸ Select All (CTRL+A)	Edit ▸ Select All (CTRL+A)

Search menu

Legacy JBuilder	Eclipse
Search ▸ Find (CTRL+F)	Edit ▸ Find/Replace (CTRL+F)
Search ▸ Find In Path (CTRL+P)	Search ▸ Search ▸ Java Search (CTRL+H)
Search ▸ Replace (CTRL+R)	Edit ▸ Find/Replace (CTRL+F)
Search ▸ Search Again (F3)	Edit ▸ Find Next (CTRL+K)
Search ▸ Search Incremental (CTRL+E)	Edit ▸ Incremental Find (CTRL+J)
Search ▸ Go To Line (CTRL+G)	Navigate ▸ Go To Line (CTRL+L)
Search ▸ Go To Class Member (CTRL+SHIFT+G)	Select class member, then Navigate ▸ Go To ▸ Next Member (CTRL+SHIFT+UP)
Search ▸ Go To Previous Method	Select method, then Navigate ▸ Go To ▸ Previous Member (CTRL+SHIFT+DOWN)
Search ▸ Go To Next Method	Select method, then Navigate ▸ Go To ▸ Previous Member (CTRL+SHIFT+DOWN)
Search ▸ Find Classes (CTRL+MINUS)	Navigate ▸ Open Type (CTRL+SHIFT+T)
Search ▸ Find Definition (CTRL+ENTER)	Navigate ▸ Open Declaration (F3)
Search ▸ Find References ▸ Javadoc QuickHelp (CTRL+SHIFT+ENTER)	Search ▸ References ▸ Project

Search ▶ Find Referring Classes	Search ▶ References ▶ Hierarchy

Refactor menu

JBuilder	Eclipse
Refactor ▶ Optimize Imports (CTRL+I)	Source ▶ Organize Imports (CTRL+I)
Refactor ▶ Rename	Refactor ▶ Rename (ALT+SHIFT+R)
Refactor ▶ Move	Refactor ▶ Move (ALT+SHIFT+V)
Refactor ▶ Inline	Refactor ▶ Inline (CTRL+SHIFT+I)
Refactor ▶ Change Parameters (CTRL+S)	Refactor ▶ Change Method Signature (ALT+SHIFT+C)
Refactor ▶ Extract Interface From	Refactor ▶ Extract Interface
Refactor ▶ Extract Method (CTRL+SHIFT+E)	Refactor ▶ Extract Method (ALT+SHIFT+M)
Refactor ▶ Surround with Try/Catch (CTRL+SHIFT+C)	Source ▶ Surround with Try/Catch Block

Project menu

JBuilder	Eclipse
Project ▶ Make Project (CTRL+F9)	Project ▶ Build All (CTRL+B)
Project ▶ Rebuild Project	Project ▶ Build All (CTRL+B)
Project ▶ Make <File> (CTRL+SHIFT+F9)	Project ▶ Build All (CTRL+B)
Project ▶ Rebuild <File>	Project ▶ Build All (CTRL+B)
Project ▶ Make Project Group	Project ▶ Build Working Set

Project Properties

Once a project has been imported into the Eclipse workspace, you can right-click the project node and choose **Properties** to view project properties, including the build and output paths, library settings, and compiler options.

Paths

When a JBuilder Java project is imported, without doing a checkout, the project's source path remains in the `/src` folder in the project's original location. If you check out a project from version control, all source files are placed in the Eclipse workspace and the source path is relative to the workspace.

The JBuilder classpath is analogous to the Java build path in Eclipse. The build path is displayed on the **Java Build Path** page of the **Properties** dialog box. By default, the output path is the `/bin` folder in the Eclipse workspace, not the `/classes` folder, as in JBuilder You can change the path on the **Java Build Path** page of the **Properties** dialog box.

Note: The Eclipse **Package Explorer** does not display projects in their build order. To see the build order for multiple projects, open the **Build Order** page of the **Preferences** dialog box (**Window** ▶ **Preferences** ▶ **General** ▶ **Workspace** ▶ **Build Order**).

Libraries

Libraries are saved to the Eclipse workspace. Libraries that are required for the project are displayed on the **Libraries** tab of the **Java Build Path** page in the **Properties** dialog box.

The project import compares JDK version labels and translates the project JDK to the JRE in the `eclipse/jre` folder of your Eclipse installation. Subsequent imports of additional projects search for a JDK with the same version as an already-imported JDK. If one exists, that JDK is used, instead of creating multiple, identical JREs.

Note: The project import brings in both project libraries and libraries that those libraries require.

Compiler Options

Imported compiler options are display on the **Java Compiler** page of the **Properties** dialog box. If the compiler compliance level for the workspace is different from that of the imported project, the import makes a project-specific override. For VisiBroker projects, the **Compiler Compliance Level** on the **Java Compiler** page needs to be set to 1.4 or 1.3.

JBuilder-Specific Properties Pages

Property pages are supplied for JBuilder specific properties, such as VisiBroker or RMI/JNI projects. To view these property pages, right-click an IDL or Java interface file after the project import. Press F1 on these pages for more information.

Related Concepts

Legacy JBuilder Project Migration Overview
Source Control
Project Nodes
Run Configuration
Importing Legacy Projects

Related Tasks

Setting Import Properties
Building an Imported Project
Running an Imported Project

Project Nodes

A legacy JBuilder project can have multiple nodes, including Java EE nodes, archive nodes, Javadoc nodes, a Generated Source node, a build node, and so on. Not all nodes can be imported into the Eclipse workspace. The following nodes are not imported:

- Archive node
- Javadoc node
- Generated Source node

If your project has an Archive node, you can recreate the archive with the **File** ▸ **Export** ▸ **Archive File** command. You can regenerate Javadoc with the **File** ▸ **Export** ▸ **Javadoc** command.

On project import, auto-generated source files are not imported or automatically regenerated. However, when you build your imported project, the generated source files are created in the `/Generated Source` folder of the Eclipse workspace. The `/Generated Source` folder is added to the source path on the **Source** tab of the **Java Build Path** page of the **Properties** dialog box (**Project** ▸ **Properties** ▸ **Java Build Path**). In Eclipse, auto-generated files are referred to as *Derived* files. The **Derived** setting is on the **Info** page of the **Properties** dialog box (**Properties** ▸ **Info** from the selected folder's context menu in the **Package Explorer**).

Related Concepts

Legacy JBuilder Project Migration Overview
Source Control
Project Properties
Run Configuration
Importing Legacy Projects

Related Tasks

Setting Import Properties
Building an Imported Project
Running an Imported Project

Run Configuration

When a Java project from JBuilder is imported into the Eclipse workspace, the run configuration is also imported. This configuration includes run and debug settings. You can view the run configuration in the **Run** dialog box (**Run ▸ Run**). Configurations are sorted by type in the tree on left. In Eclipse, a run configuration is known as a launch configuration.

Related Concepts

Legacy JBuilder Project Migration Overview
Source Control
Project Properties
Project Nodes
Importing Legacy Projects

Related Tasks

Setting Import Properties
Building an Imported Project
Running an Imported Project

Source Control

JBuilder projects under source control can be checked out to the Eclipse workspace. When you check out directly from the repository into the workspace, an Eclipse project is created without any of the project elements that are not files. The check out pulls all source files into the workspace.

JBuilder projects can be checked out from the following source control systems:

- Subversion
- ClearCase
- CVS
- StarTeam
- Visual SourceSafe

Note: Subversion, StarTeam and CVS are the only source code control systems supported by the ProjectAssist/ TeamInsight features. StarTeam and CVS can be assimilated from an existing installation only.

Warning: CVS and Subversion projects that are checked into a local repository cannot be checked out.

If the project is under source control, the **Enable VCS Plugin** option on both **Import JBuilder Project** wizards is enabled. If the project is under source control and you do not select this option, the JBuilder project is imported from its original location.

Note: Before the check out, you may be required to log into the repository or synchronize the local version with the version in the repository.

Related Concepts

Legacy JBuilder Project Migration Overview
Project Properties
Project Nodes
Run Configuration
Importing Legacy Projects

Related Tasks

Setting Import Properties
Building an Imported Project
Running an Imported Project

Concepts

Concepts

This section lists the conceptual information provided with your CodeGear product. Refer to the Getting Started with the JBuilder or JGear Product for general information and details on how JBuilder and JGear products fit into the Eclipse-based environment.

In This Section

Developing Modeling Applications

Modeling applications are developed in the Together visual editor.

In This Section

Modeling Applications Overview

This section provides information on modeling applications. Modeling provides a visual approach to Java programming. The Modeling Perspective gives you an overview of your programming projects. In the model, you can browse class relationships and explore different aspects of your project. In the Modeling Perspective, you can select programming objects from the palette and drop them into your project. You can switch between the source code and model views.

JBuilder 2008 provides round-trip integration between the source code and modeling views. When you change the model, the source code changes to match. The model also reflects changes to the source code.

JBuilder 2008 provides modeling support for Java, Enterprise Java Bean (EJB), and Java Persistence API (JPA) projects. JBuilder 2008 includes Borland's Together modeling framework, as well as some of the other Eclipse modeling environments. JBuilder 2008 can import a model from an XML schema, an Xdoclet-annotated WTP project, a Java project, or an EJB project.

JBuilder 2008 also includes the InterBase and JDataStore database systems designed to develop and test database applications. You can create Java database applications using the Modeling Perspective, which includes the Together modeling framework. Together allows dragging and dropping of components to develop the database using a visual editor.

Modeling Perspective

The Modeling Perspective provides a graphical view of a Java project. In the Modeling Perspective, you can view packages, classes, interfaces, enumerations, Enterprise Java Beans (EJBs), and the links between them. You can make changes directly to the models and those changes are reflected in the underlying code.

Model Diagrams

You can view a model diagram of any portion of your project. The diagram shows the structure and relationships of the objects in your project.

Related Tasks

Creating a Modeling Project
Importing a Modeling Project

Related Reference

Together Product Page
InterBase Home Page

Java EE Applications Development

The topics in this section describe developing Java EE applications with JBuilder 2008.

In This Section

Java EE Applications Overview

JBuilder 2008 is a Java integrated development environment (IDE). Coupled with a supported application server, the JBuilder 2008 development platform ensures creation of distributed enterprise applications that are:

- Reliable and scalable to process business transactions quickly and accurately
- Secure to protect user privacy and the integrity of enterprise data
- Readily available to meet the increasing demands of the global business environment

Tip: Use the links below to discover these topics in detail.

Java EE with JBuilder

Use JBuilder 2008 to speed up and simplify development of client server application, web applications, and UML based diagramming with support for:

- Enterprise Java Beans (EJB)
- Web Tools Project (WTP) conversion to EJB modeling projects
- Web Services
- Java Persistence API (JPA)

Tip: Use the links below to discover these topics in detail.

Java EE applications

To efficiently create Java EE applications using JBuilder 2008 become familiar with the following topics:

- Creating EJB and web services projects
- Setting Up a Runtime Server
- Publishing a Java EE Application to a Server Runtime
- Running an Application on a Runtime Server

Tip: Use the links below to discover these topics in detail.

Supported Runtime Servers

For the runtime server versions supported by JBuilder 2008, see the concept topic Runtime Servers. Task topics on runtime servers can be linked to from Working with Runtime Servers.

Related Concepts

Creating a Java EE Project
Web Applications Overview
Runtime Servers
Web Services Overview
Modeling Applications Overview
Enterprise Java Bean (EJB) Applications Overview

Related Tasks

Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server

Related Reference

Eclipse help topic “J2EE architecture Web Application Developer's Guide”

The Web Tools Project (WTP) in JBuilder

The Eclipse Web Tools Platform (WTP) Project provides APIs for Java EE and Web-centric application development. It includes both source and graphical editors for a variety of languages, wizards and built in applications to simplify Web Service development, and tools and APIs to support deploying, running, and testing applications. The ultimate objective of the project is to provide highly reusable and extensible tooling for application production efficiency. WTP provides infrastructure for:

- Web Standard Tools
- Java EE Standard Tools

Tools provided will include editors, validators, and document generators for artifacts developed in a wide range of standard languages and a specialized workbench supporting actions such as publish, run, start, and stop of Web application code across target server environments.

The Web Standard Tools Project includes server tools which extend the Eclipse platform with servers as first-class execution environments. Server tools provide an extension point for generic servers to be added to the workspace, and to be configured and controlled.

JBuilder 2008 runtimes extend WTP runtimes in cases where the WTP runtime does not support a certain server version. Use these runtimes when working with application servers in JBuilder 2008 . The following runtimes are provided:

- Apache Geronimo 1.1.1
- Apache Tomcat 5.5
- JBoss 4.0.5 GA
- GlassFish V1 UR1

Note: Supported Runtimes are noted with “(CodeGear or Borland)”.

Related Concepts

Creating a Java EE Project
Runtime Servers
Web Services Overview
Enterprise Java Bean (EJB) Applications Overview

Related Tasks

Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server

Related Reference

Eclipse help topic “J2EE architecture Web Application Developer's Guide”

Using Runtime Servers

A server runtime environment is used to test, debug, and run a project. It provides the environment, libraries, and infrastructure that a “server” needs. A server is an instance of the server runtime used to host web applications and other server-side components.

In This Section

Runtime Servers
Concept topic providing an overview of runtime server support.

Runtime Servers

A server runtime environment is used to test, debug, and run a project. It provides the environment, libraries, and infrastructure that a “server” needs. A server is an instance of the server runtime used to host web applications and other server-side components.

The following runtimes are bundled with JBuilder 2008:

- Apache Geronimo 1.1.1 (not supported as a runtime server with the Application Factory functionality)
- Apache Tomcat 6.0
- JBoss 4.0.5 GA
- GlassFish V2

Tip: A best practice when using the runtime servers is to choose one of the versions that has (CodeGear or Borland) after it. These versions have been extended to support specific JBuilder 2008 features.

The following additional product is supported by Application Factory development with JBuilder 2008 but must be purchased separately to be used as runtime server:

- BEA WebLogic Application Server 10.0

The following products are supported by Java EE development with JBuilder 2008 but must be purchased separately to be used as runtime servers:

- Borland Application Server 6.6 or 6.7, with Tibco or OpenJMS
- IBM WebSphere 6.0
- IBM WebSphere 6.1
- BEA WebLogic Application Server 8.1
- BEA WebLogic Application Server 9.2
- BEA WebLogic Application Server 10.0
- Oracle Application Server 10.1.3.2
- Oracle Containers for Java (OC4J) 10.1.3.2
- GlassFish V1
- GlassFish V1.1
- JBoss 3.2.3
- JBoss 4.2.2
- Apache Geronimo 2.0

Related Concepts

Java EE Applications Overview
Web Applications Overview
Web Services Overview
Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java EE Project
Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server
Creating a Web Application Project

Related Reference

Borland Application Server Documentation
Eclipse help topic “Server targeting for Web applications”
Eclipse help topic “Web Projects”
Eclipse help topic “Web archive (WAR) files”
Eclipse help topic “Server targeting for web applications”

Developing EJB Applications

This section discusses developing applications with Enterprise Java Beans (EJB).

In This Section

Enterprise Java Beans (EJB) Overview

Enterprise Java Beans (EJBs) are server-side components for modular construction of enterprise applications. EJBs implement a component architecture for distributed transaction-oriented applications.

Entity beans provide access to your database tables to other Java beans on the application server. Session beans provide an external interface from the application server to your Java clients. Message beans respond to asynchronous communications from Java clients.

Accessing Data with Entity Beans

Entity beans represent business data and functionality. Entity beans provide a class interface to underlying relational database structure. Entity beans provide access to database fields, keys, and relationships. Entity beans handle communications with the database server. Entity beans provide data management functionality for session and message beans. For more information on entity beans, refer to the "Accessing Data with Entity Beans" link in the Related Information section at the bottom of this page.

Providing an Interface with Session Beans

Session beans represent business processes. A session bean is a short-lived bean that executes on behalf of a single client. Session beans provide an external interface for client applications. Public session bean methods handle business activities, accessing entity beans as needed and returning data to the client. For more information on session beans, refer to the "Providing an Interface with Session Beans"' link in the Related Information section at the bottom of this page.

Asynchronous Communications with Message Beans

A message bean implements business process logic in response to receipt of an asynchronous message through the Java Messaging Service (JMS). A client sends a message to a JMS destination that is associated with your EJB container. When the message arrives, the container passes it to the onMessage method of a message bean. The message bean then performs the requested service using the message as its input. For more information on session beans, refer to the "'Asynchronous Communications with Message Beans" link in the Related Information section at the bottom of this page.

Note: The EJB 3.0 specification is significantly different from the EJB 2.x specification. JBuilder 2008 supports both EJB 2.x and EJB 3.0 applications. Please make sure that you use the correct version of EJB for your application.

Related Concepts

Enterprise Java Bean (EJB) Applications Overview
Entity Bean Overview
Session Bean Overview
Message Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Modifying an Enterprise Java Bean (EJB)

Enterprise Java Bean (EJB) Applications Overview

Enterprise Java Beans are the server-side components for Java platform applications. EJBs provide database access to client-side Java applications. EJB applications move functionality into a thick server, allowing you to write the code once for the Java beans and use it everywhere in client applications.

Java beans are developed locally and deployed to a runtime server (also called a container or application server). The presentation client accesses EJBs on the runtime server.

Enterprise Java Beans (EJBs)

Java beans are developed to handle common data manipulation tasks, business processes, and asynchronous events. For more information on EJBs, refer to the “Enterprise Java Beans (EJBs)” link in the Related Information section at the bottom of this page.

Deploying EJBs to the Runtime Server

After EJBs have been developed and tested, they are deployed to a runtime server to be used by client applications. For more information on deploying EJBs, refer to the “Deploying EJBs to the Runtime Server” link in the Related Information section at the bottom of this page.

Note: The EJB 3.0 specification is significantly different from the EJB 2.x specification. JBuilder 2008 supports both EJB 2.x and EJB 3.0 applications. Please make sure that you use the correct version of EJB for your application.

Related Concepts

Enterprise Java Beans (EJB) Overview
Deploying Enterprise Java Beans (EJBs) Overview

Related Tasks

Creating an Enterprise Java Bean (EJB) Modeling Project
Importing an Enterprise Java Bean (EJB) Modeling Project

EJB Environment and Resources Overview

The enterprise bean's environment provides for customized bean data at runtime without the need to access or change the enterprise bean's source code. The environment provides beans with an object-independent way to refer to a value, a resource, or another component. The value of such environment references (or variables) is set at deployment time, based on the contents of the deployment descriptor.

EJB References

Each EJB reference describes the interface between the referencing enterprise bean and the referenced bean. You can define references between beans within the same JAR file or from an external enterprise bean (one that is outside the JAR file but in the same application).

Each EJB local reference describes the interface between the referencing enterprise bean and the local referenced bean.

For information on adding EJB References, refer to the “Creating an EJB Reference” link in the Related Information section at the bottom of this page.

Resource References

A resource reference identifies a resource factory reference of the enterprise bean. A set of resource references enables the application assembler or the bean deployer to locate all the references used by the bean. For information on adding a resource reference, refer to the “Creating a Resource Reference ” link in the Related Information section at the bottom of this page.

Environment Entries

The environment entry allows you to customize the bean's business logic when the bean is assembled or deployed, without the need to access or change the bean's source code directly. Each enterprise bean defines its own set of environment entries. All instances of an enterprise bean share the same environment entries. Enterprise bean instances aren't allowed to modify the bean's environment at runtime. For information on adding an environment entry, refer to the “Creating an Environment Entry ” link in the Related Information section at the bottom of this page.

Environment Resource References

A resource environment reference provides a logical name for a physical object. The client application uses the logical name to find the resource at runtime. For information on adding an environment resource reference, refer to the “Creating an Environment Resource Reference ” link in the Related Information section at the bottom of this page.

Related Concepts

Enterprise Java Bean (EJB) Applications Overview
Deploying Enterprise Java Beans (EJBs) Overview

Related Tasks

Creating an EJB Reference
Creating a Resource Reference
Creating an Environment Entry
Creating an Environment Resource Reference

Deploying Enterprise Java Beans (EJBs) Overview

Enterprise Java Beans (EJBs) are deployed to a runtime server (also known as a container or application server) for later use by client applications.

Security in an EJB environment is handled by the container, not the bean business methods. EJB security roles provide a bean-independent way to provide access to enterprise bean methods.

EJBs abstract away from the underlying server architecture. Environment and resource references provide a way for enterprise beans to refer to available resources without relying on a particular server configuration.

Using the Runtime Server

Bean suppliers deploy their provided EJBs to a runtime server, where they are available to client applications. For more information on using the runtime server, refer to the “Using the Runtime Server” link in the Related Information section at the bottom of this page.

EJB Security Roles

EJB security information is handled separately from bean business methods. This allows the bean deployer to configure security in the most appropriate way for the operational environment. EJB security roles provide a way for bean developers to permit access to bean methods to different categories of users. For more information on security roles, refer to the “EJB Security Roles” link in the Related Information section at the bottom of this page.

EJB Environment and Resources

For more information on EJB environment and resources, refer to the “EJB Environment and Resources” link in the Related Information section at the bottom of this page.

Related Concepts

Enterprise Java Bean (EJB) Applications Overview
EJB Security Roles Overview
EJB Environment and Resources Overview

Entity Bean Overview

This section describes EJB entity beans. Entity beans represent business data that is stored in a database. Each entity bean stands for an individual item instance in the underlying data store. Entity beans provide an object-oriented interface to the underlying relational database. Entity beans persist across client calls and can be shared by multiple clients.

Session beans access business data through entity beans. The entity beans implement common database functions, shielding the session beans from the underlying database schema.

Entity Bean Persistence

The state of an entity bean consists of its underlying data. The entity bean's state exists beyond the lifetime of the application. An entity bean is persistent because its state exists even after you shut down the database server or the applications it services. Entity beans manage their persistence by staying synchronized with the data store. Entity bean persistence can be managed either by the bean code itself (bean-managed persistence) or by the bean container (container-managed persistence). With bean-managed persistence, you write the SQL code to access the database. With container-managed persistence, the EJB container handles the database access for you.

Container-managed persistence is more portable than bean-managed persistence. The code for a CMP entity bean contains no SQL code and is thus independent of the underlying database. Container-managed persistence allows you to redeploy the entity bean on a different J2EE server without modifying or recompiling your code.

Entity Bean Relationships

Each entity bean has a unique object identifier. The unique identifier, or primary key, enables the client to locate a particular data item. An entity bean can be related to other entity beans in the same way that database rows and be related to rows in other tables. One entity bean contains the primary key for another entity bean as part of its data. The two entity beans can be joined by matching these fields and the resulting data used.

Note: The EJB 3.0 specification is significantly different from the EJB 2.x specification. JBuilder 2008 supports both EJB 2.x and EJB 3.0 applications. Please make sure that you use the correct version of EJB for your application.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Container-Managed-Persistence (CMP) Entity Bean
Creating a Bean-Managed-Persistence (BMP) Entity Bean
Importing Entity Beans from a Database
Creating a One-Way Relationship Between Entity Beans
Creating a Relationship Between Entity Beans
Adding a CMP Field to a CMP Entity Bean
Creating the Primary Key for an Entity Bean
Adding a Primary Key Join Field to an Entity Bean
Adding a New Named Query to an EJB 3.0 Entity Bean
Adding a New Named Native Query to an EJB 3.0 Entity Bean
Adding a Result Set Mapping to an EJB 3.0 Entity Bean
Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean
Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean
Adding a New Pre-Update Method to an EJB 3.0 Entity Bean
Adding a New Post-Load Method to an EJB 3.0 Entity Bean
Adding a New Post-Persist Method to an EJB 3.0 Entity Bean
Adding a New Post-Remove Method to an EJB 3.0 Entity Bean
Adding a New Post-Update Method to an EJB 3.0 Entity Bean
Adding a Home Method to an EJB 2.x Entity Bean
Adding a Find Method to an EJB 2.x Entity Bean
Adding a Select Method to an EJB 2.x Entity Bean

Message Bean Overview

This section describes message beans. A message bean provides asynchrony to EJB applications by acting as a JMS (Java Messenging Service) message consumer. A message bean is associated with a JMS topic or queue and receives JMS messages sent by EJB clients or other beans. Like stateless session beans, message beans maintain no client-specific state. Clients send JMS messages to message beans. A message bean listens for messages, using a single onMessage method to process received messages.When a message arrives, the container ensures that a message bean corresponding to the message topic/queue exists , and calls its onMessage method with the client's message as the single argument.

A message bean retains no data or state for a specific client. A single message-driven bean can process messages from multiple clients.

Message Destinations

The message destination indicates the JMS destination type (topic or queue) to which the message bean will bind. The message bean functions as a full-fledged JMS client, indistinguishable from any other JMS client. In addition to functioning as asynchronous JMS clients, message beans also support message concurrency. Since message beans are stateless and managed by the container, they can both send and receive messages concurrently (the container simply grabs another bean out of the pool).

The OnMessage Method

When a message arrives, the container calls the message-driven bean's onMessage method to process the message. The onMessage method normally casts the message to one of the five JMS message types and handles it in accordance with the application's business logic. The onMessage method can call helper methods, or it can invoke a session or entity bean to process the information in the message or to store it in a database.

Note: The EJB 3.0 specification is significantly different from the EJB 2.x specification. JBuilder 2008 supports both EJB 2.x and EJB 3.0 applications. Please make sure that you use the correct version of EJB for your application.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Message Bean
Creating a Message Destination for a Message Bean
Creating a Message Destination Link for a Message Bean

EJB Security Roles Overview

Security roles define the permission required to run EJB methods. A security role is a set of logically related method permissions. The application assembler defines the security roles and method permissions for each set of deployed EJBs. A user must have at least one security role associated with a method in order to invoke the method.

Defining Security Roles

The application assembler specifies the methods of the remote and home interface that each security role is allowed to invoke. The assembler defines the method permissions relation in the deployment descriptor. Each method-permission element includes a list security roles and a list of methods. The listed security roles may invoke the listed methods. A security role or a method can appear in multiple method-permission elements.

Running As a Security Role

An EJB, Java control, or web service method can run under the security role of the invoking user, or it can run under a different security role. This might be necessary when the EJB uses resources that have strict security requirements.

Related Concepts

Enterprise Java Bean (EJB) Applications Overview
EJB Security Roles Overview

Related Tasks

Creating a Security Role
Creating a Security Role Reference
Creating a Run-As-Security Link

Session Bean Overview

This section describes session beans. Session beans provide an interface from the EJB container to client applications. Session beans implement business processes through entity beans.

Session beans represent business tasks, not persistent data. For example, a Session bean might performs a database search for a user and return the results to the user. Session beans can communicate with all other types of beans, and can thus be used for many tasks other than database transactions.

A session bean is composed of a component interface, a home interface, a bean implementation class, and a deployment descriptor. The component interface contains the client business methods of the bean. The home interface contains methods for the bean life cycle. The bean implementation class implements all the methods that allow the bean to be managed in the container. The deployment descriptor contains bean properties that can be edited at assembly or deployment time.

Session Bean States

A session bean is a short-lived bean that executes on behalf of a single client. Stateless session beans do not preserve state across method calls. Each call to a session bean invokes a standard stateless session bean with no memory of previous calls to the same session bean.

Stateful session beans preserve their states within and between transactions. If the client invokes method calls against the same bean stub, the calls are sent to the same bean instance in the container. Field variables in the bean instance retain their values as long as the client application retains the bean reference.

Component Interface

Business methods provide the component interface of the session bean to the client. Each business task provided by the session bean is represented in a business method. Clients call the business methods of the session bean to manage business tasks for them. For more information on adding a component method, consult the “Adding a Business Method” link in the Related Information section at the bottom of this page.

Note: The EJB 3.0 specification is significantly different from the EJB 2.x specification. JBuilder 2008 supports both EJB 2.x and EJB 3.0 applications. Please make sure that you use the correct version of EJB for your application.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a New Session Bean
Adding a Business Method to an EJB
Adding an Interceptor Method to an EJB 3.0 Session Bean
Adding a Post-Construct Method to an EJB 3.0 Session Bean
Adding a Pre-Destroy Method to an EJB 3.0 Session Bean
Adding a Timeout Method to an EJB 3.0 Session Bean

Developing Web Services

The web services features in the JBuilder or JGear products allow you to quickly design, deploy, run, and test a web service.

In This Section

Web Services Overview

You can create software that performs a set of tasks, and then make it available to others by running it on a web server over a network, such as the Internet or a local area network. A web service's public interfaces and bindings are defined and described in a service description language. Developers can then discover these service descriptions and use them to write a client application to invoke and access your services.

JBuilder 2008 helps you develop web services quickly. You can develop bottom-up web services by exporting a Java class or bean to a web service. Once you open your Java class as a web service, the service is immediately runnable.

Supported Runtime Servers

For the runtime server versions supported by JBuilder 2008, see the concept topic Runtime Servers. Task topics on runtime servers can be linked to from Working with Runtime Servers.

Web Services Standards

The standards on which web services development is based are evolving technologies. The primary standards include:

- EJB (Enterprise JavaBean Technology))
- SOAP (Simple Object Access Protocol)
- WSDL (Web Services Description Language)
- UDDI (Universal Description, Discovery and Integration)
- WSIL (Web Services Inspection Language)
- JAX-RPC (Java API for XML-based Remote Procedure Call)
- WS-I (Web Services Interoperability)
- SAAJ (SOAP with Attachments API for Java)

See the Eclipse *Web Application Development Guide* for information on web services standards.

Web Service Properties

When the Web Services Designer is open, you can set service and WSDL, WSIL, and UDDI properties. Service properties control the web service; WSDL, WSIL, and UDDI properties control the connection to the web service. Defaults are provided for the targeted server runtime and toolkit so that the service is immediately runnable.

Building and Running Web Services

When you export a Java class to the Web Services Designer, JBuilder 2008 builds the dynamic web project hosting the web service or web services. Most of the web services artifacts are regenerated. Files in the `/GeneratedSource/` folder are deleted and recreated. They are read-only.

You cannot edit the `JUnit` test case in the `/GeneratedSource/` folder. Instead, you can edit the `JUnit` subclass in the `/src/` folder. This file will not be overwritten at build time.

Before you can run the web service, you have to configure the runtime for your web server or container.

When you run the web service, the run configuration for the target web or application server is used. To view the run configuration, open the **Run** dialog box (**Run ▸ Run**). The run configurations for the server are in the **Web Service** node. The configurations for the client are in the **Web Client** node.

When you run the web service, the WSDL, WSIL, or UDDI file is validated. If the file is not valid, the web service won't run.

JBuilder 2008 Web Services Tools

End-to-end web services generation in JBuilder 2008 uses both existing WTP features, as well as JBuilder 2008-only features like the Web Services Designer. Much of the work, such as file generation, is done behind-the-scenes. JBuilder 2008 provides:

- **Web Services Explorer**— A design surface for visually creating and implementing web services.
- **Properties pane**— A pane for setting service and WSDL properties.
- **Add Web Service From URL** wizard—A wizard for creating a web service from a WSDL located at a URL.
- **Convert into Web Services Client Project** wizard—A wizard to convert an existing Java project with a WSDL into a web services client project.

Apache Axis Toolkit

JBuilder 2008 supports the Apache Axis web services toolkit.

The Apache Axis toolkit is an open source implementation of Simple Object Access Protocol (SOAP), an XML-based protocol for exchanging information. When you target your project for Apache Axis, the appropriate web services files, including a WSDL document, are generated for you. Methods of the selected class are exposed as a web service. Server-side classes, including as an implementation for the server, are created automatically.

Related Concepts

Apache Axis Toolkit
Web Services Designer Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime
Working in the Web Services Designer

Web Services Designer Overview

The Web Services Designer provides a design surface for quickly creating, implementing, and validating web services. Once you open the Web Services Designer, the service representation for the exported JavaBean is displayed. In the Web Services Designer, you set service and WSDL options with inspectors.

All activity in the Web Services Designer is persisted on disk using a Web Services Deployment Unit (WSDU) file, named `axis.wsdu`. A toolkit-specific build file, named `build_axis.xml`, is created in the same folder. Any activity in the Web Services Designer is then entered as an Ant task in this build file. The Ant build file invokes appropriate build tasks and other build functions.

Service Representation

When you export a JavaBean to a web service, a service representation is created in the Web Services Designer. The representation contains fields. Each field has an associated inspector. For example, when you design a Java web service, the service representation contains Service, Methods, Server, and Settings fields.

To access an inspector, click a field in the service representation. Values in the inspectors are automatically filled in for you when you export a Java class to a web service. The default toolkit values are also filled in for you, although you can change them.

Note: When the top field of a service is checked, web services generation is enabled. If it is unchecked, the service is disabled, the inspectors are not available, and the service is not built.

You use the inspectors to set service and WSDL options. Changes are applied to the WSDL file at the next build. Service options include:

- Service style
- Use
- Type mapping version
- SOAP action
- Location URL
- Service name
- Binding name
- Port type name
- Deploy scope

WSDL options include:

- Include WSDL file
- Import schema
- Target namespace
- Output/Interface WSDL file
- Location import URL
- Implementation WSDL file
- Implementation namespace

Web Services Designer UI

After you have created a dynamic web project and exported a JavaBean to a web service, a Web Services Designer node appears in the **Project Explorer**. An Axis module is displayed as a child node of the Web Services Designer node.

Web services are displayed as service representations in the Web Services Designer. The service representation contains fields, such as Server, Service, and WSDL. Each of these fields has an associated inspector for setting service options. Click a field to open its inspector.

The Web Services Designer has a toolbar for creating and deleting services and for viewing source code. The context menu contains commands for viewing the implementation source.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer

Apache Axis Toolkit

Apache Axis is an open source implementation of Simple Object Access Protocol (SOAP), an XML-based protocol for exchanging information. Axis uses the Simple API for XML (SAX) instead of the Document Object Model (DOM). It is a modular, flexible, and a high performance implementation of SOAP. The Apache Axis toolkit is JAX-RPC (Java API for XML-based Remote Procedure Call) compliant and supports WSDL 1.1. It also provides document/literal support for the WS-I Basic Profile 1.0 and JAX-RPC 1.1 specifications.

When you target your web service for Apache Axis, the toolkit generates the required web services files, including a WSDL document. Methods are exposed as a web service. Server-side classes are created automatically. You can choose to create a client project to test your web service.

Exporting a Web Service

The Axis toolkit supports exporting Java classes and stateless session beans as web services. When working in the Web Services Designer on an Axis web service, you can set properties for the service. When you export a Java class or EJB, a WSDL is automatically generated according to the options you set. The WSDL is an XML file that describes the service.

Note: The WSDL is automatically added to the root of the `/WebContent/` folder in your project.

When you export a class to a web service, the dynamic web project is enabled for Axis. The following occurs:

- The AXIS `JAR` files are copied to the `/WebContent/WEB-INF/lib` folder.
- Entries are added to the `web.xml` file for the Axis servlets.
- The `Java2WSDL` builder is added to the project.
- The `WSDD` builder is added to the project.
- The server-side Axis informational pages are added to the `/WebContent/` folder.
- A run configuration is created to start the web or application server.

Importing a Web Service

You can import a WSDL from a URL with the **Add Web Service from URL** wizard (**File ▸ New ▸ Other ▸ Web Services ▸ Web Service Client from URL**). The Axis toolkit generates client-side classes for consuming a service and a `JUnit` test case to test interaction with the web service. The files generated by the toolkit, which are saved in a package name based on the WSDL target namespace or one that you specify, are dependent upon the properties you set for the service in the Web Services Designer.

Building a Web Service

If **Project ▸ Build Automatically** is enabled, the web service is built automatically when you modify the web service in the Web Services Designer. If this option is off, you need to build the project with build commands.

As you work in the Web Services Designer, an Ant build file, `build-wsdl2java.xml` is generated and written to the `/WebContent/` folder in your project. The appropriate Ant tasks are generated and saved to this Ant build file. At build time, the Ant build file is passed to the toolkit to generate the appropriate web services files.

The generated source is read-only and should not be modified, with these exceptions:

- The client-side `JUnit` test case subclass is not overwritten when you regenerate web services. Any modifications you make to this file won't be overwritten. This file is in the `/src/` folder of the Client project.

- Bean types are not overwritten when you regenerate web services.

Axis Java2WSDL Parameters

The Axis java2wsdl Ant task generates a WSDL from Java classes. Java classes form a WSDL. Mappings from namespaces to packages are provided as nested mapping elements. These parameters are the same settings that you make in the Web Services Designer. This task doesn't do any dependency checking.

Axis WSDL2Java Parameters

The Axis `wsdl2java` Ant task generates Java classes from a WSDL. Mappings from namespaces to packages are provided as nested mapping elements. These parameters are the same settings that you make in the Web Services Designer. This task doesn't do any dependency checking.

Web Services Files

When you use the Web Services Designer to create web services, files are generated by the Axis toolkit. These web services can include server-side classes, deployment files for the server, and a WSDL document to describe an exported service. At build time, these files are passed to the Apache Axis toolkit to generate the appropriate web services files and deployment information. The web services files generated by the Axis toolkit are accessible in the `/WebContent/` folder for the server and the `/WebContent/`, `/Generated_Source/` and `/src/` folders for the client.

Web Services Explorer Files

build_java2wsdl.xml	Server- and client-side. The Ant build file for a bottom-up web service.
build_wsdl2java..xml	Server- and client-side. The Ant build file for a top-down web service.
server-config.wsdd	Server-side. XML file created by the Axis toolkit at build time. Provides deployment information to the server.

Client-side Files

<service name>Proxy.java	The client proxy. (Client `/Generated_Source/` folder)
<service name>Service.java	A service interface that defines a get method for each port listed in the service element of the WSDL. This service interface defines a factory class to get a stub instance. (Client `/Generated_Source/` folder)
<service name>ServiceLocator.java	A locator class that is the client-side implementation of the service interface. (Client `/Generated_Source/` folder)
<service name>TestCase.java	A JUnit test case for testing the web service. (Client `/Generated_Source/` folder)
<service name>TestCaseImpl.java	A subclass of the JUnit test case. Add tests to this subclass instead of the generated JUnit test case class. (Client `/Generated_Source/` folder)
<PortType name>.java	An interface for each `portType` in the WSDL. The implementation of this interfaces calls remote methods. (Client `/Generated_Source/` folder)
<binding name>Stub.java	A client-side stub class that acts as a proxy for a remote web service. Allows you to call the web service as if it were a local object. This class implements the `<PortType name>.java` interface. (Client `/Generated_Source/` folder)
data types	Java files for all other types. Holders needed for the web service. (Client `/Generated_Source/` folder)

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Developing Web Applications

The JBuilder 2008 web development environment provides the tools needed to develop simple (consisting of only static Web pages) or more advanced dynamic web application based on the Java EE specification. JBuilder 2008.

In This Section

Web Applications Overview

This topic describes web application development in JBuilder 2007.

Web Applications Overview

Static or detailed dynamic web applications are quickly developed with JBuilder 2008. Static web projects include images, HTML files and cascading style sheets. Dynamic web projects contain dynamic Java EE resources. Web application are deployed within a web project to the server in the form of a Web archive (WAR) file. The web application is viewed as a web site from a web browser.

Static Web Applications

For web applications that require only basic content use the static web project type. Static web applications can be converted to dynamic web projects.

Dynamic Web Applications

Dynamic web projects may include Java Server Pages and servlets and are based on the Java EE model which defines a web application directory structure.

Related Concepts

Runtime Servers

Related Tasks

Creating a Web Application Project
Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server

Related Reference

Eclipse help topic “Server targeting for Web applications”
Eclipse help topic “Web Projects”
Eclipse help topic “Web archive (WAR) files”
Eclipse help topic “Server targeting for web applications”
Eclipse help topic “Web Projects'
Eclipse help topic “Creating a static web project”
Eclipse help topic “Dynamic web projects and applications”
Eclipse help topic “Web page design”

Java Persistence API (JPA) Applications Development

The topics in this section describe developing JPA applications with JBuilder or JGear.

In This Section

Java Persistence API Applications Overview

Your JBuilder or JGear product is a Java integrated development environment (IDE). Coupled with a supported application runtime server, the JBuilder or JGear development platform allows the creation of distributed enterprise applications that are:

- Reliable and scalable to process business transactions quickly and accurately
- Secure to protect user privacy and the integrity of enterprise data
- Readily available to meet the increasing demands of the global business environment

Java Persistence API (JPA) was included in your JBuilder or JGear product to simplify the development of Java EE and Java SE applications using data persistence.

Java Persistence API with JBuilder or JGear

JBuilder 2008 can speed up and simplify development of Java applications. With JPA and modeling features, it allows you to create a Java modeling project with JPA support. You can based your JPA project on standard persistence technologies, such as:

- Hibernate
- TopLink
- Others, of your choosing

JPA is a POJO persistence API of object/relational mapping.

JPA Applications

To efficiently create JPA applications using JBuilder 2008 become familiar with the following task and dialog reference topics:

- Creating a Dynamic Web Java Persistence API (JPA) Modeling Project
- Creating a Java Persistence API (JPA) Modeling Project
- Dynamic Web JPA Modeling Dialogs Reference
- JPA Modeling Dialogs Reference
- Running an Application on a Runtime Server

Supported Runtime Servers

For the runtime server versions supported by JBuilder 2008, see the concept topic Runtime Servers. Task topics on runtime servers can be linked to from Working with Runtime Servers.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Runtime Servers

Related Tasks

Creating a Java Persistence API (JPA) Modeling Project
Working with Runtime Servers

Related Reference

New JPA Modeling Project: Persistence unit settings page
New JPA Modeling Project: Java Settings
Hibernate Documentation
TopLink Resources

Application Factory Concepts

This section contains conceptual information regarding the many aspects of the JBuilder or JGear Application Factory functionality.

In This Section

Application Factory Overview

The Application Factory functionality introduces an application-driven development paradigm, where the structure, evolution, and logic behind the development of the application are checked into version control along with the source code for the application itself.

Major software development processes use a bewildering array of frameworks and technologies to build even the simplest of applications. In addition, many typical applications are built over and over again from the ground up, resulting in duplicated efforts. Such methods involve a steep learning curve for developers, requiring knowledge of the frameworks and technologies along with application-specific models and behaviors. Due to the scope of this effort, it is very common to have teams narrowly focused on the many aspects of the application. However, it is absolutely required that each feature fit into the overall architecture. The problem multiplies over the age of the project as more features get added and the teams are in flux. Application Factory fundamentally addresses this issue by attaching application-specific metadata throughout the life of the application's development. Application Factory provides developers with the ability to attach actionable behavior to application modules. These behaviors go all the way from complete code generation to laying bread crumbs for newer developers to follow and implement.

Application Factory records a developer's intent when a particular piece of code was written. This allows the workings of the code to be easily ascertained at a later date. Application Factory facilitates the process of capturing the original developer's intent and context by providing in-IDE tools for making notes about code, as well as creating bread crumb trails to project- and file-specific developer knowledge. This metadata stays attached to the code and can be opened by any subsequent developer to understand the context and purpose of code snippets, methods, and classes.

Using Application Factory, system architects can develop templates that include Application Factory pointers. These pointers help developers understand the rationale and correct technique for implementing specific features. The system architect includes code and Application Factory tags that explain to the developer what custom code needs to be written and how to configure the application. With these templates in place, sites can use application templates to quickly customize, build, and deploy applications.

The Application Factory functionality provides tools to:

- Organize code visually.
- Track changes.
- Associate changes to actions.
- Data mine actions from the past.
- Associate all project artifacts in the context of the desired user story or task.

Application Factory Fundamentals

Application Factory allows you to store application-specific information along with the application in an Application Module. An Application Module is a set of Application Projects associated with an application, in combination with the metadata project for the application. Each workspace can contain only one Application Factory project. The Application Factory project that accompanies an application module includes:

- Tags—keywords associated with a piece of information
- Application Diagram—visual representation of application architecture and functionality
- Scripts—code generating/templating mechanism providing a way to generate template code
- Readme —overview of application functionality
- Cheat Sheet—cheat sheet providing important steps for using the application and scripts

Refer to the subsections and links below for more information on Modules, Projects and metadata types.

Application Factory Modules

An Application Module is a complete application available as a set of JBuilder projects. Attached with an Application Module is Application Factory metadata in the form of an Application Factory project. The metadata in the Application Factory project enables an application-driven development model.

Pre-packaged Applications Modules ship with your JBuilder or JGear product and may include data-aware web applications, shopping carts, E-commerce systems, or Eclipse Monkey DOM project.

Refer to Application Factory Modules

Application Factory Projects

Each application can become an Application Factory Module (also known as an application module) by creating the Application Factory project for an application. Currently each workspace supports a single application (which could include multiple projects) and a single Application Factory project.

A new project can be created using the new Application Factory wizard (**File** ▸ **New** ▸ **Project or Other** ▸ **Application Factory** ▸ **Application Factory Project**).This wizard creates a template readme and cheatsheet along with an empty tag repository and application diagram. The project structure is created and global scripts and templates can be pulled into the new project as well.

The Application Factory project includes the following artifacts.

Tags

A tag is a keyword associated with a piece of information. Tags are typically used to group related resources. The application module project contains a set of tags that provide an organizational and navigational mechanism for the application. This tag repository for an application is stored in the Application Factory project.

Each tag is associated with multiple resources in the application (defined as a set of projects in a workspace). Tags can have parent-child relationships and can be related to each other. Each tag can have a description and associated notes.

The **Tags View** is used to create and manage tags for an application. The **Tags View** provides the ability to focus the workspace on the file set associated with each tag. This provides an easy way to navigate through the application.

Tags and selected resources can be marked as **Application Diagram** candidates. This allows the tags and associated relationships to be exposed in the **Application Diagram** associated with the module. For example, a set of high-level parent tags can be exposed via the application diagram to describe application functionality.

Application Diagram

The **Application Diagram** describes application architecture and functionality. This diagram can include application architecture, employed technologies, third-party dependencies, and so forth. The diagram is useful as a tool to describe how the internals of the application work to a new user. The **Application Diagram** is stored in the Application Factory project.

The diagram surfaces information from the tags, mainly tags marked as diagram candidates. The diagram also displays description/notes for tags. It also represents parent-child relationships and related tags. The diagram provides a high-level summary of the application.

The **Application Diagram** can be displayed by using a context-menu item (right-clicking) on the Application Factory project file. You then choose **Application Factory** ▸ **Application Diagram** to open the Application Diagram, or you can double-click on the Application Factory project in the **Model Navigator** view to open the **Application Diagram**.

Application Module Scripts

Application module scripts can dynamically:

- Generate wizard-like dialogs.

- Invoke Java and Eclipse API methods.
- Invoke generate or modify files.

Script generation through a wizard, through the Recipe Editor, or through context actions can be a convenient starting point for new users. The scripts are written using JavaScript. The scripts can use FreeMarker templates to help perform the tasks of file and code generation and modification of already existing resources. Scripts and FreeMarker templates are stored in the Application Factory project under the Scripts folder.

The scripting mechanism (that is, running scripts and so forth) is based on Eclipse Monkey, a project within Eclipse which surfaces the Java and Eclipse APIs for scripting. JBuilder extends this with additional APIs to help you in file and code generation. The APIs available to a script are defined in the script metadata, a Javadoc-like comment block at the top of the script file.

JBuilder provides wizards to generate template scripts and script recipes that can be used to generate complex scripts/templates. The generated scripts use standard UI widgets to prompt for parameters (for example, project names, package names, search/replace patterns, and so forth). Scripts can also be generated by mining VCS commit history for commonly used patterns. Script run history is available for the Application Factory project, any file in the workspace, and any script using the **Archeology** view.

Supported Runtime Servers

For the runtime server versions supported by Application Factory and JBuilder 2008, see the concept topic Runtime Servers. Task topics on runtime servers can be linked to from Working with Runtime Servers.

Related Concepts

Workbench Features of Application Factory
Application Factory Projects
Application Factory Modules
Application Factory Users
Runtime Servers

Related Tasks

Using Application Factory
Working with Runtime Servers

Related Reference

FreeMarker Template Engine Overview
Eclipse Monkey Help Front Page

Workbench Features of Application Factory

The Application Factory features in the workbench allow you to enhance your development process by providing the following functionality:

- Wizard to create an Application Factory Project
- Tag Cloud Viewer and Editor, with the following capabilities:
- Creating Tags
- Adding parent-child relationships to tags
- Associating project resources to tags
- Focusing IDE views based on tag

- **Application Diagram** view, with the following capabilities:
- Abstracting of high-level tags and associated relationships to a UML-based diagram that describes the application architecture
- Adding notes to a diagram
- Adding tags and associate resources directly to the diagram

- Code generation abstraction using scripts, with the following capabilities:
- Generating a template script through a wizard
- Using the **Script Recipe for Application Factory** for script generation
- Managing scripts through the **Scripts—Application Factory** view
- Accepting and resolving script change and set phases
- Using the Script run **Archeology** view
- Focusing IDE-views based on script run archeology

- **Application Factory Explorer** and Module Editor, with the following capabilities:
- Displaying available application modules through the **Explorer** view
- Filtering on the **Explorer** view by type of application and frameworks used
- Providing preview, screenshots, tag and application diagram snapshots, and licensing information through a Module Editor/viewer for an application module
- Importing (consuming) and exporting (publishing) application modules

- Pre-built Application Modules, that include data-aware web application modules based on AppFuse

Wizards

The Application Factory functionality implements several new wizards in the workbench, including:

- Application Factory Project—creates the Application Factory project templates and metadata in the workspace.
- Script for Application Factory—creates a template script from selection from a list of pre-defined DOMs (APIs) and addition of standard user-interface widgets for resource location (project, file, class, entity) and based on a Freemarker template (
FreeMarker Template Engine Overview
).
- Script Recipe for Application Factory—creates a new script recipe that permits complex script generation.

For more details on scripting, the Script Recipe for Application Factory and Recipe Editor, refer to the Application Factory Modules concept topic.

Perspectives

In an Eclipse-based IDE, a perspective determines visible action and views within a window. Each perspective can contain a number of views, explorers, and editors.

The **Application Factory Producer** perspective is the default perspective that users are placed in after the user creates an Application Factory module project or an Application Factory project. The perspectives associated with the Application Factory functionality are:

- Application Factory Producer Perspective
- Application Factory Repository Exploring Perspective
- Application Factory Java Perspective
- Application Factory Modeling Perspective

Application Factory Producer Perspective

This is the default perspective after completing the newApplication Factory wizard.

When the user completes a new Application Factory wizard, the Application Factory Producer Perspective includes the following views:

- Tags View
- Scripts—Application Factory View
- Package Explorer View
- Navigator View
- Commit History View

When the user creates a new Application Factory project, the following views/editors are displayed:

- Application Diagram
- Application Module Editor (in read-write mode)
- Template Readme in the HTML Editor
- Template cheat sheet in the Cheat Sheet Editor

Application Factory Repository Exploring Perspective

The Application Factory Repository Exploring Perspective is the default perspective when launching the product for the first time or after importing an application module using the Application Factory Explorer. The Application Factory Repository Exploring Perspective includes the following views/editors:

- Application Factory Explorer View—similar to the Package Explorer, this view allows users to browse the library of available application modules and consume (import) or publish (export) applications modules
- Tags View
- Scripts—Application Factory View

When the user imports an application module into the workspace, the following files/views are opened in the Application Factory Repository Exploring Perspective:

- Application Module Editor (read-only)
- Application Diagram
- Application Cheat Sheet
- Readme

Application Factory Java Perspective

Application Factory functionality adds the following views to the base Java Perspective, including:

- Tags View
- Scripts—Application Factory View

When the user imports a data-aware application module into the workspace, the following files/views are opened in the Application Factory Java Perspective:

- Application Module Editor
- Application Diagram
- Application Cheat Sheet
- Readme

Application Factory Modeling Perspective

Application Factory includes a unique Application Factory Modeling perspective using the Together LiveSource™ technology that allows a visual representation of your project. All Java code is live and editable in graphical form as UML (Unified Modeling Language) 2.0 sequence diagrams and class diagrams.

Application Factory functionality adds the following aspects to the Application Factory Modeling Perspective, including:

- Tags View
- Scripts—Application Factory View

When the user imports a data-aware application module into the workspace, the following files/views are opened in the Application Factory Modeling Perspective:

- Application Module Editor
- Application Diagram
- Application Cheat Sheet
- Readme

Views

The Application Factory functionality implements several new views to the workbench, including:

- Tags
- Scripts — Application Factory
- Script Learn/Resolve/Commit
- Commit History

- Archeology
- Application Factory Explorer

At any time, the user can open any of these views by following the IDE path of **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** and choosing the appropriate view name.

Tags

The Application Factory project contains a set of tags that provide an organizational and navigational mechanism for the application. This tag repository for an application is stored in the Application Factory project.

Each tag can be associated with multiple resources in the application (defined as a set of projects in a workspace). Tags can have parent-child relationships and can be related to each other. Each tag can have a description and notes.

The **Tags** view is used to create and manage tags and resource associations for an application. The **Tags** view provides the ability to focus the workspace on the file set associated with each tag. This provides an easy way to navigate through the application.

Tags and selected resources can be marked as **Application Diagram** candidates. This allows the tags and associated relationships to be exposed in the **Application Diagram** associated with the module. In this manner, a set of high-level parent tags can be exposed via the **Application Diagram** to describe application functionality.

Scripts — Application Factory

The **Scripts — Application Factory** displays scripts in the Application Factory project. The view automatically filters out scripts without a main method. (In other words, scripts that cannot be executed directly. A dropdown list in the toolbar displays all scripts for an Application Factory project.)

Double-clicking on a script name in the **Scripts — Application Factory** view executes the script. Right-clicking on the script and selecting **Edit** opens the script in the editor.

Toolbar options and equivalent context menu options in the **Scripts — Application Factory** view allow the following actions:

- Open script run archeology for a selected script
- Focus IDE views on the script run for a selected script
- Focus the **Scripts — Application Factory** view on the scripts associated with the active (currently selected) tag

The **Scripts — Application Factory** also provides an option to create a batch script that executes a series of scripts. To generate a connecting script, multi-select the scripts that you want to execute in a batch, right-click and select **Create Connecting Script**.

Scripts Learn/Resolve/Commit

The **Scripts Learn/Resolve/Commit** view is a tree-view list of files, associated script(s) and snippet(s) that need to be resolved. This view is similar to the VCS synchronization view in Eclipse. Along with the entry in the **Scripts Learn/Resolve/Commit**, a description for each change (file/snippet) is displayed in a wrapped-text format. The file list pane has a toolbar for various resolution options (commit all changes, discard all changes and so forth). The file list uses different markers to indicate missing resources and the level of confidence for the change.

Clicking on each file change displays the associated change, including the description for the change, in the upper-right pane. The script scrolled to the appropriate line is displayed in the lower-right pane. Right-clicking on each file change allows you to perform the following actions:

- Provide a new location for the target file
- Change the insert location for the snippet in the target file

- Open the compare view displaying the difference in the target file with and without the change
- Discard changes for a file

Commit History

Data mining of information in your version control system can be done using the **Commit History** view in Application Factory. The **Commit History** view displays information about all commits into the Subversion Version Control System (VCS) repository. The **Commit History** view pulls in the VCS information from projects in the workspace that are under source control. It then aggregates them by date. The resulting data can be filtered and searched by date, author and commit comment text.

Open the **Commit History** view by selecting the menu path **Window** ▶ **Show View** ▶ **Other** ▶ **Application Factory** ▶ **Commit History**. This view can only be used if a repository has already been configured in the workspace.

The **Commit History** view allows users to enter search parameters based on author name, date range and check-in comment. Entries from the search results in the **Commit History** view can be used in a script recipe to generate a script and template for code generation based on the VCS entries. Associated actions are also available as a context menu option for each entry or a multiple selection of entries in the commit history results. This automatically generates code in the script to create the same resource and the code snippet is associated with the entry from the commit history viewer. The snippet (VCS entry) is converted to a Freemarker template (
FreeMarker Template Engine Overview
).

Archeology

The **Archeology** view displays script runs for a file, runs for a single script and script runs for the Application Factory project.

The **Archeology** view allows for navigation (browser style to navigate between filtering contexts). The archeology viewer functionality allows you to:

- Display script runs for a file, runs for a single script and script runs for the project. The script runs can be filtered by date, author and script name.
- Display details for each script run. Details include the list of files affected by a selected script run, and the script associated with each file change (with the script scrolled to the appropriate line which changed the file).
- Use context menu options to open a compare view for the file, to open source for the file and to show archeology for the file (that is, script runs for the selected file in the list).
- Focus action via a toolbar button. Focus action allows you to focus the IDE on all resources affected by the script run to see all runs that modified a file.

There are three panes in this view:

- Script Run pane—displays script runs as per context (file, project, script) grouped in tasks. This pane is a tree table with script run information in columns that can be used to filter the runs. The column information includes script name, date/time, and author.
- File List pane—displays a list of files affected by a selected script run. This is the left-hand pane below the Script Run pane, There are context menu options on the file list to open a compare view for the file, to open the source for the file, and to show archeology for the file (that is, script runs for the selected file in the list).
- Script Change pane—displays the script scrolled to the appropriate line that changed the file. This is the right-hand pane below the Script Run pane.

The Archeology Viewer can be accessed using the IDE path **Window** ▶ **Show View** ▶ **Other** ▶ **Application Factory** ▶ **Archeology**. The **Archeology** view comes up empty when launched in this manner.

The **Archeology** view can also be accessed using context menu option **Application Factory** ▸ **Open Script Run Archeology** at the project, file and script level. This redirects to the Archeology View with the appropriate filter applied.

Application Factory Explorer and Application Module Editor

The **Application Factory Explorer** view is a browser for application modules. Application modules are exported (published) by default to the modules directory under the root of the JBuilder install directory. Multiple locations for application modules can be provided at **Windows** ▸ **Preferences** ▸ **Application Factory** page.

The **Application Factory Explorer** view is included in the **Application Factory Repository Exploring** perspective. The **Application Factory Explorer** filters available application modules by the type of application or the type of framework(s) or license used in the application module.

The **Application Module Editor** is anchored on a module or a module archive file. The editor can be opened from within the **Application Factory Explorer**. The editor is in read-only mode when opened from the **Application Factory Explorer** and displays information about the module based on the module archive. The **Application Module Editor** is also opened by default when the user creates a new Application Factory project. In this case, the editor is in read-write mode and allows user to edit Application Module properties.

Refer to the Application Factory Modules concept topic for additional information on the **Application Factory Explorer** and Application Module Editor.

Application Diagram

The **Application Diagram** view can be displayed through the **Application Factory Modeling Perspective** or through any of the **Explorer** views by right-clicking on the Application Factory Project. The diagram displays tags (and relationships) that are exposed as diagram candidates. It also displays the description and notes associated with the tag and any resources associated with the tag that are exposed in the diagram.

Refer to the Application Factory Modules concept topic for additional information on Application Diagrams.

Related Concepts

Tour of the User Interface (UI)
Application Factory Overview
Application Factory Projects
Application Factory Modules
Application Factory Users

Related Tasks

Using Application Factory

Related Reference

FreeMarker Template Engine Overview
Eclipse Monkey Overview Page

Application Factory Users

An Application Factory user may fall into several categories or perform several user functions as application development with Application Factory progresses:

- Ad-hoc Developer uses application-specific metadata capture to track the structure and evolution of any stand-alone project (independent of Modules).
- Module Producer generates new modules that can be repeatedly employed for rapid application development.
- Module Consumer employs new modules for rapid application development.

Ad-hoc Developer Role

In the Ad-hoc Developer function, the user start to tag and track application evolution allowing the customization of existing scripts and creation of new scripts. In this role, the Ad-hoc Developer may:

- Work with a set of pre-existing or new projects (no Modules necessary).
- Use basic Application Factory capabilities to create an Application Factory project.
- Optionally, import some global scripts.
- Use the tag features to tag resources.
- Create an **Application Diagram** based on tags.
- Check the new Application Factory project into a version-control system (VCS) similar to any other application projects.

The Developer user can also use advanced Application Factory capabilities for:

- Parameterizing and templating capabilities with script creation wizards.
- Creating recipes for scripts from workspace resources, VCS history and action records using the Application Factory Script Recipes.

Module Producer Role

A Producer can be defined as an architect who is new to Application Factory. In the Producer role, the user has previously captured usable program functionality as an Ad-hoc Developer. A Producer starts with a set of created projects to generate an Application Factory Module.

The Producer then minimally performs these steps to export the Module for general use:

- Uses the tag feature to tag resources.
- Uses the Tag Cloud to navigate around a complex application without a having to go back and forth from a diagram.
- Uses the ability to focus only on resources pertaining to a selected tag and filter out all other resources from the view.
- Creates an application diagram based on tags.
- Adds a Readme, Cheat Sheet and Application Module information through an application module editor.
- Exports and publishes the Application Module.

Most Application Modules take advantage of the behavior creation features in Application Factory, so the Producer can quickly prepare the Module for export. To prepare to export the Module, the Producer:

- Uses Script creation wizards with parameterizing and templating capabilities.
- Uses Script DOM creation modules to push most of the “behavior” into Java with a thin script layer for the front-end.
- Uses the **Recipe Editor** to create templates and scripts from workspace resources. The scripts are capable of producing large amounts of customized source code via inflating templates.
- Uses the Recipe Editor to create recipes for scripts through VCS mining for change snippets.
- Uses the Recipe Editor along with watch-mode to create recipes for scripts based on macro-style actions.

Consumer Role

The Application Factory Module Consumer uses previously-created Application Factory Modules to rapidly build other applications. In this role, the Consumer would work with the application projects as any developer normally does. In addition, the Consumer makes use of Application Factory scripts as needed.

A Consumer can be defined as a team member who is new to Application Factory and who is mandated to work with Application Modules. This user starts by checking out or installing an Application Module. An Application Module could be a single project or multiple projects along with the Application Factory project for the application. The Consumer then proceeds to learn about the application using the application diagram, tags and scripts. The Consumer can then proceed with assigned tasks. This might include adding tags/scripts in this process, which they can then re-export as a new Application Module to share with the rest of the team. The Application Factory Consumer would:

- Launch the **Application Factory Explorer** and choose an application to import from the gallery. A set of Application Modules ship with JBuilder or JGear products, such as data-aware web applications, shopping carts and e-commerce systems. The **Application Factory Explorer** allows a preview of functionality as well as providing information about the application. The gallery is also populated with the users' Application Modules.
- Application projects get created in the workspace along with an Application Factory project.
- The Application Factory project includes a Framework Diagram, Tag Diagram, Readme and Cheat Sheet guide to understanding and using the application, and application-specific code generators as scripts.

The Consumer works with the application projects in the normal manner and makes use of Application Factory scripts as needed. Using Application Factory, the Consumer has a deployable application from the very beginning with an automatically-created action trail. The Consumer learns while using Application Factory tools on action trails and tags and can choose to modify/add tags. This knowledge, along with the project itself, can then be shared with the rest of the development team.

When comfortable with the Application Factory functionality, the Consumer can customize and create application-specific behavior. The Consumer can use the extensive application creation tools (Producer version only) or the Consumer can use the script wizard to create a rich skeleton and works directly with scripts.

After the Consumer enhances an Application Module, it can be shared for reuse by exporting as an archive and publishing the Application Module (Producer version only) or by exporting the enhanced module through version control.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Projects
Application Factory Modules

Related Tasks

Using Application Factory

Application Factory Projects

An Application Factory project is an Eclipse project that resides in the same workspace as the application projects.

A new project can be created using the new Application Factory wizard (**File** ▶ **New** ▶ **Project or Other** ▶ **Application Factory** ▶ **Application Factory Project**). This wizard creates the project structure along with a template readme, template cheat sheet, an empty tag repository, and an application diagram. Global scripts and templates can be pulled into the new project as well.

The Application Factory project contains the following components:

- **Scripts Folder:** Contains scripts and templates used in the application.
- **Tags Folder:** Contains the tag repository for the application. There can be only one tag repository per application module.
- **Preview Folder:** Contains screenshots of the application, snapshots of the tag/application diagram and any other resources required to describe the application.
- **readme.html File:** Contains a readme file describing the application and the concepts of Application Factory. There can be only one readme per application module.
- **cheatsheet.xml File:** Contains a cheat sheet with steps to execute the most common actions in an application module. Some of the steps are generic to all application modules (for example, how to use the tag view). There can be only one cheat sheet per application module.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Users

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
FreeMarker Template Engine Overview

Application Factory Modules

An Application Factory Module is a complete application available as a set of JBuilder projects. An Application Factory Module consists of the application project(s) and a companion project (known as the Application Factory project) that contains information about the application. This metadata enables an application-driven development model.

An Application Module is stored as a module archive file (.mar). The archive file contains the Application Factory project and all application projects. JBuilder ships with pre-packaged Application Modules. Users can browse through and install these Modules using the **Application Factory Explorer** view in JBuilder.

Modules can be published to the default directory (*JB_HOME*/modules) through the **Application Module Editor (in read/write mode)** or by using the **Export Application Module** wizard (**Export** ▶ **Application Factory** ▶ **Export Application Module**).

A custom directory, or a list of search directories for Application Modules, can be specified through the **Application Factory Preferences** page (**Window** ▶ **Preferences** ▶ **Application Factory** ▶ **Module Search/Export Directories**.)

Any module created with Application Factory can be re-exported. If a module is designated as an add-on modules when published (exported), this module can be imported later by a consumer and the consumer can add that module's content to an Application Module that already exists in the workspace.

Module Application Types

Modules include the following application types:

- Data-Aware Application (web applications based on AppFuse)
- Template Applications (such as Pet Store, Book Store or Eclipse Monkey DOM template applications)
- Top-Down Application (E-commerce application based on Apache OfBiz)

In addition, any module can be re-exported and even configured to allow it to be imported as an add-on module. RSS/Atom feeds can also be designated as an import location for application modules.

Data-Aware Application Modules

The Data-Aware Applications modules are based on AppFuse (AppFuse Home Page), which is an open-source project based on popular Java and web application frameworks. AppFuse enables users to build web applications quickly and efficiently and the data-aware application modules in JBuilder make web application even faster.

The Data-Aware Application modules include the web application project and the Application Factory project. The Application Factory project contains the application diagram, tags and scripts that help users learn about the application. The Application Factory project functionality is surfaced in the **Application Factory Repository Exploring** and the **Application Factory Modeling** perspectives.

The three Data-Aware Applications supported as Application Modules are as follows:

- JSF Web Application
- Spring MVC Web Application
- Struts 2 Web Application

All three applications use JPA/Hibernate as the backend layer and are currently deployable using Apache Tomcat 6.0 and MySQL 5.x . All three applications are packaged as modules (that is, they include complete application metadata such as tags, diagrams, and so forth).

All of the applications include the following custom script functionality:

- CRUD—Scripts allow generation of CRUD (Create, Replace, Update, Delete) functionality based on an entity. The scripts handle all types of relationships and generate user interface code to display master-detail relationships. Field display and master-detail display options are available in the entity/field properties in the JPA diagram for the project. CRUD functionality supports all types of primary keys and database relationships. Current application server and database support includes Apache Tomcat 6.0 and MySQL 5.x.
- Changing styles—Scripts allow changing the CSS style for the application.
- Changing company name—Scripts allow changing the company name for all pages.
- Create Tables from entities—Scripts allow the generation of database tables based on JPA/Hibernate entities.

Top-Down Application Modules

A top-down application module in Application Factory is a complete application that can be instantly deployed without needing any changes. Of course, a top-down application can be changed and modified at any level to fit more specific needs.

Application Factory provides a top-down application module for creating E-commerce applications using Apache OFBiz technology. The open-source Apache OfBiz project provides a framework for building business-class applications. For more information on OFBiz, refer to the Apache OFBiz website at
The Apache Open for Business Project
. For Application Factory tasks to creating an E-commerce application using the OFBiz framework, refer to Creating E-commerce Applications with Application Factory.

Template Application Modules

Template applications provide a useful blueprint for future applications. All are registered Eclipse plugins and create an application quickly and easily. Template applications currently available with your JBuilder or JGear product include:

- Pet Store—provides a blueprint application from the conversion of the Java EE 5.0 Blueprints Pet Store sample into an application module for JBuilder and JGear.
- Book Store—provides a blueprint application with bookstore functionality.
- Eclipse Monkey DOM Plugin—.creates the skeleton of an Eclipse plugin, which registers an Eclipse Monkey DOM (Domain Object Model). This surfaces an API that can be called by the script when the DOM is installed and the DOM identifier is part of the script metadata. (This script metadata is located in the comment block at the top of the script file).

Add-on Modules

Add-on Modules can be created when an Application Module is exported from Application Factory. Any add-on module can be imported later by a consumer into the IDE and that consumer can add that module's content to an Application Module that already exists in the workspace.

Note: If a module to be imported was not marked as an add-on module when it was published (exported), attempting to import it when an Application Module already exists in the workspace generates a warning about only one Application Module allowed in the workspace.

When an add-on module is consumed (imported), it is created in a subdirectory of the Application Module in the current workspace. The subdirectory is located under the current Application Module's workspace add-on module directory. It has the add-on module's name as the parent directory name. For example, if you import a module named `Test` as an add-on module to your existing Application Module named `FirstModule`, the add-on module is created under the`FirstModule/Add-on/Test` directory. Once imported as an add-on module, the module is not available as a separate module but instead as files in the current workspace Application Module.

After a module is exported as an add-on module for later import, it appears in the **Application Factory Explorer** view with the other template projects. The Add-on filter is activated in the **Application Factory Explorer** for the project so you can filter the right-hand column view for **Add-on Modules**, **Not Add-on Modules**, or both.

RSS/Atom Feeds

When an Application Module is exported, the producer of the module can designate an RSS/Atom feed file to accompany the module for later deployment. Application Factory supports generating and reading both RSS and Atom feed type files.

Both the created RSS/Atom feed file and the associated module archive (.mar) file have to be physically deployed to the location specified in the **Export Application Module** wizard. Once deployed, an RSS feed URL can be added to the **Module Search/Export Directories Preference** page. The specified RSS feed can then be read as a location for importing an Application Module from either the **Application Factory Explorer** view, or from the **Import Application Module** wizard.

The remotely deployed Application Module is not loaded until the consumer selects to create or add an application from the **Application Module Editor** but all information about the remotely deployed Application Module is available in the RSS/Atom feed file.

Until it is removed from the Feeds list, the RSS/Atom feed file location is read when the list of importable Application Modules is shown to a consumer in the **Application Factory Explorer** or the **Import Application Module** wizard. In the **Application Factory Explorer**, there is an **Import Location** filter field in the left-side pane. This filter allows you to see and filter the right-side view importable Application Modules by their import location. **RSS Modules** is one of the filter options. Selecting this filter options shows only the modules that can be imported from RSS/Atom feeds.

Exploring Modules

The **Application Factory Explorer** view allows users to browse a library of available Application Modules. Application Modules that ship with JBuilder are located in the modules directory under the root of the JBuilder install directory. Multiple locations for Application Modules can be provided in the **Window** ▸ **Preferences** ▸ **Application Factory** Preferences page. The **Application Factory Explorer** displays application modules (.mar files) from the specified list of directories.

Double-clicking on a module in the **Application Factory Explorer** opens the **Application Module Editor** on the right-side of the workspace.

Users can create applications based on modules (that is, install application modules into the workspace). They can also choose to produce application modules and publish (export) them so they are listed in the **Application Factory Explorer** view.

Application Module Editor

The **Application Module Editor** is anchored on a module or a module archive file. The editor can be opened from within the **Application Factory Explorer**. The editor is in read-only mode when opened from the **Application Factory Explorer** and displays information about the module based on the module archive. The **Application Module Editor** is also opened by default when the user creates a new Application Factory project. In this case, the editor is in read-write mode and allows user to edit Application Module properties.

Each module supports the following elements within the Application Module Editor:

- Application Name
- Application Description.
- Frameworks Used
- Screenshots

- Tag Diagram Snapshot
- Application Diagram Snapshot
- License

The user interface for the **Application Module Editor** include tabs for the following:

- Preview—Screenshots for the application. In read-write mode, drag and drop image files onto the thumbnail icon in the Preview pane.
- Tag Diagram Snapshot—A snapshot of the Tag Diagram for the Application Module. This is automatically generated when an Application Module is exported (published).
- Application Diagram Snapshot—A snapshot of the Application Diagram for the Application Module. This is automatically generated when an Application Module is exported (published).
- License—License information for the module. The UI provides standard licensing choices from which to select.

Scripting

Code generation capabilities for an application can be abstracted using scripts included with Application Factorymodules. Available for these scripts is an extensive Java API from installed plugins that make up the IDE. These APIs include standard code generation/file manipulation functions (for example, locating resources, searching for patterns, generating new files, modifying existing files, and so forth)

JBuilder application modules include a set of global scripts and examples. The scripts easily make use of FreeMarker templates for code generation. Most JBuilder scripts use Freemarker templates (
FreeMarker Template Engine Overview
).

Wizards available to help write new scripts include:

- **Script for Application Factory** —This wizard creates scripts using existing project and/or files as templates.
- **Script Recipe for Application Factory**—This wizard creates a single script including a complex user interface if needed and helps you produce code to create, modify, or delete workspace files.

Script for Application Factory

Creating a template—based script, this wizard offers selection from a list of pre-defined DOMs (APIs), addition of a user-interface, and generation of script code to modify or create files. The wizard permits users to select a FreeMarker template (
FreeMarker Template Engine Overview
), which is parsed for simple FreeMarker values that need value assignments. The wizard automatically generates UI to prompt for template variable parameters.

Script Recipe for Application Factory and Recipe Editor

The wizards create a new script recipe file. The recipe opens in a the **Recipe Editor** after the wizard completes. Script recipes are stored in the Application Factory project. Scripts and associated templates are generated automatically based on files and projects that have added to the recipe. The **Recipe Editor** automatically generates templates and scripts to create projects, files and apply snippets to modify or create files.

Users can add tasks to the recipe and populate the tasks in the following manner:

- Create recipes to generate a project:
- Create a new task in the script **Recipe Editor** using the **Add Task** toolbar button.

- Drag and drop any existing open projects from the workspace onto the upper pane. A dialog appears prompting the user to select resources that should be included in project creation.
- Right-click on the task in the **Recipe Editor** and select the **Generate script** option.
- The template and script appear beneath the task.
- Click on a resource file or template or the script to open in the **Recipe Editor** lower pane. The script and template can be modified and tested using the **Test** button in the toolbar.

- Create recipes to generate file content:
- Create a new task in the script **Recipe Editor** using the **Add Task** toolbar button.
- Drag and drop any existing files from the workspace onto the task.
- Right-click on the task in the **Recipe Editor**and select the **Generate script** option.
- The template and script appear beneath the task.
- Click on the template or the script to open in the **Recipe Editor**. The script and template can be modified and tested using the **Test** button in the toolbar.

- Create recipes to generate snippets based on Version Control System (VCS) VCS history—Use the **Commit History** view to mine data on various code commits.

Commit History View

The Commit History View displays information about all commits into the Subversion VCS repository.

Open the **Commit History** view by selecting the menu path **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Commit History**. This view can only be used if a repository has already been configured in the workspace. The Commit History View allows users to enter search parameters based on author name, date range and check-in comment. Entries from the search results in the **Commit History** view can be dragged and dropped directly onto a task. Associated actions are also available as a context menu option for each entry or a multiple selection of entries in the commit history results. This automatically generates code in the script to create the same resource and the code snippet is associated with the entry from the commit history viewer. The snippet is converted to a FreeMarker template (
FreeMarker Template Engine Overview
).

Existing Resources

Use existing resources in the workspace as templates for code generation by dragging and dropping files from the workspace onto the task. These actions automatically generates code in the script thats create the same resource with the same content. A Freemarker template (
FreeMarker Template Engine Overview
) is created for the content.

Script Runs

Script run reliability is an important part of scripting. Application Factory functionality ensure that script runs provide the user with control over the code generation process. The user can:

- Accept all code generation changes as is.
- Choose to apply all code generation changes to different locations in a resource.
- Choose to discard individual changes.

The user can use the scripts with modified application modules and choose to fix any resource location problems (due to modifications) by hand.

When a script is run, code snippets are created in a snippet directory under the Application Factory project. If there are any problems with the script run, the **Script Learn/Resolve/Commit** view is displayed at the end of the script run. If there are no problems with the script run, a dialog pops up asking if user wants to see a detailed list of changes. The **Script Learn/Resolve/Commit** view is displayed if user chooses the option.

Script Resolution

Once a script has created and run, the **Script Learn/Resolve/Commit** view is displayed at the end of the run if there are any problems. If there are no problems with the script run, a dialog pops up asking if user wants to see a detailed list of changes. The purpose of the **Script Learn/Resolve/Commit** view is to notify the user of any merge conflicts or of any resources that could not be located during the script run.

The Script Resolve View includes:

- A list of resources that are created/changed.
- A compare view showing before and after commit changes.
- A Detail View showing the line of script that caused the change.
- An Description View displaying the description associated with the change.
- A Problems View if there are any pending merges or unresolved entries in the Script Resolve View.

The **Script Learn/Resolve/Commit**view is a tree-view list of files, associated script(s) and snippet(s) that need to be resolved. Along with entry, a description for each change is displayed.. The file list pane has a toolbar for various resolution options (commit all changes, discard all changes and so forth.) The file list uses different markers to indicate missing resources and the level of confidence for the change.

Double-clicking on an entry in the file list tree view opens the **Compare** view in an editor mode. The Compare view can also be opened by using the context menu and selecting **Open Compare Viewer**. This view displays the difference between the original file with and without the change introduced by the snippet(s). Right-click context menu options permit changing the insert location in the target file and also discarding changes for the file. Toolbar options in the file list pane and the script pane allow the opening of the file or the opening of the script in the associated editor.

Changes are based on a snippet or multiple snippets per file. Code snippets are stored in a snippets directory with the metadata for the Application Factory Project. When reviewing snippets, the user is able to accept or discard multiple changes or all changes. User-modified files are detected and marked to indicate that the user should not rely on the proposed location for snippet merging. The user can choose a location for each snippet. A change log stores script notes and snippet-specific notes.

Each entry in the **Script Learn/Resolve/Commit** view for which the resource is successfully located shows a marker for the level of confidence of the code generation change. Missing resources are flagged visually. The user can provide a new location for missing resources.

The Compare view displays the exact location where the generated code is to be placed. The marker is located at the top of the file in cases where exact location cannot be determined. The user can change the insertion point for the generated snippet in the target resource by right-clicking on the resource in the file list and selecting **Change Insert Location**.

All snippets in the Application Factory project are deleted after the commit process is complete. A warning is displayed if unresolved snippets exist in the Application Factory project when a script run is launched.

Consuming Modules

Users who consume modules bring up the **Application Factory Explorer** view, browse through available applications, and press the **Create Application** button to install the Application Module into the current workspace. This process involves unzipping the Application Module archive (.mar) file into the current workspace. This includes the Application Factory project and all associated application projects, and any add-on projects or associated RSS/ Atom feeds.

Note: Each workspace can only contain a single Application Factory project.

Publishing Modules

Users can publish modules using the **Export Application Module** wizard (**File** ▸ **Export** ▸ **Application Factory** ▸ **Export Application Module**). This wizard creates a module archive (.mar) file that includes all projects in the current workspace. Application Modules are exported (published) by default to the modules directory under the root of the JBuilder install directory. Alternate locations can be provided in the **Window** ▸ **Preferences** ▸ **Application Factory** ▸ **Modules Search/Export Directories** page. When publishing a module, it can be designated as an add-on module, An RSS/Atom feed file to accompany the module for later deployment can also be designated.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Projects
Application Factory Users

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
AppFuse Home Page
FreeMarker Template Engine Overview
Eclipse Monkey Help Front Page

ProjectAssist and TeamInsight Concepts

This section describes the concepts of the ProjectAssist and TeamInsight installed developer tools. These tools are installed on (or assimilated through) a ProjectAssist server and development team members can access the tools to manage and coordinate their work through TeamInsight portlets (Bugzilla, Continuum/Maven, CVS, Liferay, StarTeam, Subversion, and XPlanner).

In This Section

ProjectAssist and TeamInsight Overview

ProjectAssist and TeamInsight are JBuilder 2008 features that install and facilitate the use of a suite of development tools. The TeamInsight tools enhance the performance of your software development team. These tools help coordinate teamwork and thereby optimize your team's efforts. The tools are installed and configured on a ProjectAssist server by the ProjectAssist Administrator.

Only the JBuilder 2008 Enterprise editions support the ProjectAssist installation features. The JBuilder 2008 Enterprise edition supports the ProjectAssist installation or assimilation features for the following software products: Version Control (CVS, StarTeam or Subversion), Build System (Continuum), Defect Tracking (Bugzilla or StarTeam) or Task Provider (StarTeam or XPlanner).

As part of the ProjectAssist install, the Administrator defines projects and team members for the projects. The team members can then coordinate their efforts through the use of the various TeamInsight tools. The TeamInsight tool selection is determined by the JBuilder 2008 product edition (Enterprise). The following tools can be available through a ProjectAssist installation or assimilation:

- Liferay to open the team's web portal, which summarizes the current status of the project and provides access to several TeamInsight components.
- CVS, StarTeam or Subversion for version control, which allows team members to check source files in and out, and to synchronize the repository files.
- Subversion Viewer (Sventon) to browse the Subversion source repository.
- Continuum/Maven to establish an automatic build environment linked with the repository and to monitor build and quality status.
- Bugzilla or StarTeam to record and track defects and change requests in the source code.
- StarTeam or XPlanner to monitor development progress by creating and tracking projects, iterations of projects, user stories, and individual tasks.
- Burn down chart and current iteration details from XPlanner.

Defining Users and User Roles

As part of the ProjectAssist server install and configuration, the ProjectAssist Administrator defines projects and adds users for those projects. For each ProjectAssist component, the ProjectAssist Administrator categorizes users into one of two user roles (administrator or developer) or no access for that component. When adding a user, all components default to the Developer role for all users. The exception is the shared MySQL component, which defaults to No Access for everyone but the ProjectAssist Administrator.

Administrator	If you are the ProjectAssist server Administrator, you install ProjectAssist on a server or assimilate existing project components over single/multiple machines. The ProjectAssist Administrator adds projects, users, assigns user access rights, and sends user notifications regarding the server install and the TeamInsight client installation. The ProjectAssist Administrator sample default administrator, Joe Bloggs, is assigned Administrator privileges for all TeamInsight components by default. He can be cloned to establish a Administrator identity for a new user. As part of the server configuration of users, the ProjectAssist Administrator assigns the user unique roles for each TeamInsight tool by right-clicking on the user name. For example, a user can be assigned XPlanner administrative privileges yet retain Developer privileges on other TeamInsight tools.
Developer	Users are granted access to repositories based on assigned project rights. Users receive access to the ProjectAssist components from the ProjectAssist server Administrator. Configuration information is downloaded from either the attachment to the ProjectAssist user notification message or from the Liferay portal.

No Access	For each TeamInsight tool, a user can be assigned a No Access role by the ProjectAssist Administrator. No access means the user has no read, write nor execute access to the component. The ProjectAssist Administrator can assign a No Access role by right clicking on the user name.

Liferay Portal Summarizing Project Status

The Liferay portal provides reports from several TeamInsight tools, including details about the Subversion repository, progress in the current XPlanner project, bug tracking status, project build status, and QA Lab summary information.

The Liferay portal contains the following portlets:

- Subversion repository status (JBoss Labs Kosmos Subversion)
- CVS repository information for project repositories
- Current iteration details (XPlanner)
- Burn down chart and current iteration details from XPlanner
- Build status (Continuum)
- Bug/defect status (Bugzilla)
- QALab summary and QALab classes (QALab)
- StarTeam Task, Change Requests, and/or StarTeam version control repository information

To match your project needs, the Liferay portal can be customized to display other portlets.

Subversion, CVS, or StarTeam Repository for Version Control

TeamInsight can include an assimilated CVS repository or a local or remote Subversion repository for version control in the JBuilder 2008 Enterprise Edition.

Version control systems manages project files and directories in the repository over time. Team members can simultaneously check out the projects stored in a shared repository. When team members check in their changes, the version control system synchronizes the repository using an edit-update-commit paradigm and manages branching in the source repository.

If Subversion is chosen as the version control system, TeamInsight gives each team member access to the Subversion Viewer, a read-only browser for Subversion repositories (Sventon). Team members can access the Subversion Viewer through the TeamInsight Viewer. The Subversion Viewer enables users to browse, download, view logs and locks, and diff the files in the repository.

If CVS is assimilated as the version control system, TeamInsight can display a CVS web view of the repositories. The CVS web view URL is specified through the **Stacks** tab in the ProjectAssist Designer. The designated URL should be a pre-installed web view of the repository. Team members can access the specified CVS web view URL through a tab at the bottom of the TeamInsight Viewer.

Continuum/Maven Continuous Build System

Continuum is a continuous integration build system that builds and tests code on a regular basis. Continuum monitors the source repository on a specified schedule and triggers a build if any changes have been made in the repository. As developers check in code throughout the day, builds are triggered. Many builds can occur daily. A build can be defined as anything from testing and compiling a single project to the assembly and testing of a deliverable from multiple projects. Continuum allows for rapid turnaround of integrated builds, thus supporting quicker identification of critical build issues.

The Maven tool can build and manage Java-based projects. Maven is a software project management and comprehension tool that is based on the concept of a project object model (POM). Maven allows a project to build using the POM and a set of plugins that are shared by all projects using Maven, providing a uniform build system. Maven can manage a project's build, reporting and documentation from this central piece of information. Maven creates a way to build the projects, a clear definition of project content, an easy way to publish project information, and a way to share Java Archives (JARs) across several projects. The result is a tool that can be used for building and managing Java-based projects.

Bugzilla for Bug Tracking

Bugzilla is a bug tracking system that allows development teams to report, track and repair defects in their products while remaining team connected. It is a powerful tool that allows effective team organization and communication. Bugzilla also expands the use of the term "bug" to track other things such as feature requests so it can also be used for some general project tracking.

Borland ALM StarTeam for Change Requests, Task Planning or Version Control

The JBuilder 2008 Enterprise Edition supports the ProjectAssist installation for the assimilation of an existing StarTeam installation

During the ProjectAssist server installation with JBuilder 2008 Enterprise Edition, the user can choose to include StarTeam, or alternate installations or assimilations, for version control, change requests or tasks:

- Version Control System (CVS, StarTeam, or Subversion)
- Continuous Build system (Continuum)
- Defect or Change Request system (Bugzilla or StarTeam)
- Task Provider (StarTeam or XPlanner)

You can also open a StarTeam task or bug repository using Mylyn, a task-focused interface integrated into the JBuilder 2008 product.

XPlanner for Project Planning

XPlanner is an open-source Web-based planning and monitoring tool designed for projects using eXtreme programming (XP) or agile project management. XPlanner provides:

- Project iterations (sprints)
- User stories (product feature requirements)
- User tasks (work assignments)
- Metrics for tracking progress

You can also open an XPlanner task repository using Mylyn, a task-focused interface integrated into JBuilder 2008 or ProjectAssist 2008.

Mylyn for Bugzilla, StarTeam and XPlanner

JBuilder 2008 enables you to include the Bugzilla or assimilated Borland ALM StarTeam repository bugs and StarTeam or XPlanner repository tasks in the Eclipse **Task List** view. You can then use Mylyn to define queries against those repositories, such as a "My Bugs" query. The Mylyn plugin automates task-focused user capabilities for Bugzilla, StarTeam and XPlanner. After you activate a task, Mylyn remembers the context of your subsequent work, such as the files associated with the active task. Later when you return to the task, the preserved context enables you to work more efficiently.

Eclipse Wiki

As part of the team collaboration features of JBuilder 2008, a Wiki for team members is now available inside the IDE. A project Wiki can be a powerful project documentation tool for team members to coordinate efforts, disperse unified project information, and take full advantage of the open-source community. Editing Wiki files inside the IDE provides automatic page creation and dynamic linking (to other pages and Java source in the workspace). Checking the .wiki files into the version control system permits team members that checkout the project to view and edit the wiki. Wiki content can also be exported into a set of static web pages, including coversion of Java files into HTML.

The Eclipse Wiki plug-in is provided with JBuilder 2008. However, it must be enabled and configured according to documentation provided on the web or in the Eclipse Wiki Help files included with JBuilder 2008.

Note: Refer to the Eclipse Wiki Help files and the following "Related Reference" link for more information on enabling and configuring your Wiki plugin.

Related Concepts

Liferay: The TeamInsight Project Portal
Subversion: Source Code Repository
CVS: Source Code Repository
Bugzilla: Defect Tracking System
Continuum/Maven: Continuous Build System
Mylyn Concepts
StarTeam: Source Code Repository, Change Request Tracking, and Task Provider
XPlanner: Project and Team Management

Related Reference

Eclipse Wiki Editor Plugin Web Information

Liferay: The TeamInsight Project Portal

The TeamInsight tools support a Liferay web portal accessible from client workstations. Team members typically access the Liferay portal to check project status.

The Liferay portal contains portlets or windows that display status reports from several of the TeamInsight tools. Several of the portlets also contain a link to the TeamInsight tool that generated the information on the portlet. For example, the XPlanner windows in the portal both contain a link to XPlanner.

Liferay Portal Gathers Reports From TeamInsight Tools

By default, the Liferay portal contains the following reports from the TeamInsight tools (if installed) and can be accessed through a tab on the TeamInsight Viewer:

- Status report for the Subversion repository, compiled by the Kosmos monitoring plugin
- CVS repository information for project repositories
- Burn down chart and current iteration details from XPlanner
- Build status from Continuum
- Bug tracking status from Bugzilla
- QALab Summary and QALab Classes
- StarTeam Task, Change Requests and/or StarTeam version control repository information

Default Contents of the Liferay Portal

The Liferay project portal contains status reports from the TeamInsight tools (if installed) as follows:

- **JBoss Labs Subversion**: Gives the location of project repositories and, for each repository, the current revision number, the total number of committers, activity in the last 7 days, and the age of the latest touch (change). Each field, except for the Latest touch age, contains a link to details or expanded information. For example, the link for the project repository field opens a pair of charts (Repository entry history and Files by file type). The link for the current revision number displays revision details (Most active files). This portlet's information comes from the Kosmos Subversion monitoring tool from JBoss Labs.
- **CVS Repository Information**: Displays data on the project repository, commit log, active files, developer details, and setup information.
- **Burn Down Chart**: Displays a graph representing the burn down rate (total remaining hours of work over time) for the current iteration. Tabs display similar XPlanner information from other iterations. Users can link directly to XPlanner.
- **Current Iteration Details**: Summarizes the hours completed on the stories in the current project iteration. Tabs display similar XPlanner information from other iterations. Users can link directly to XPlanner.
- **Build Status**: Displays a summary of the most recent project builds. Includes links to the Results for each build, and a link to Continuum. The **Project Health** link opens the project web page for Continuum.
- **Bugzilla Status**: Lists bugs reported for the project, as well as statistics about bug reports. The Bugzilla Status portlet has six tabs and a link to go to the Bugzilla bug management tool. The Important tab lists the most important bugs, while the Newest tab lists the most recently reported bugs. The Severity and Priority tabs display graphs. The Assignee tab displays a pie chart showing the relative number of bugs assigned to each team member. The Trends tab displays statistics about bugs over time.
- **QALab Summary and QALab Classes:** Summarizes statistics about recent QA results obtained from the Cobertura and PMD open-source plugins. QALab works with Continuum/Maven to compile results in a qalab.xml file for each run. QALab Classes lists all the classes that are responsible for the results.

- **StarTeam Repository Information:** Displays data on the StarTeam project repository.
- **StarTeam Tasks**: Lists open tasks for the project, as well as setup information, including Project Name, Login Name, StarTeam view, folder, host and endpoint data.
- **StarTeam Change Requests**: Lists change requests reported for the project, as well as assignee and setup information. The StarTeam Change Request Status portlet has three tabs. The Newest tab lists the most recently reported bugs. The Assignee tab lists bugs assigned to each team member. The Setup tab displays setup information, including Project Name, Login Name, StarTeam view, folder, host and endpoint data.

Related Concepts

XPlanner: Project and Team Management
Subversion: Source Code Repository
CVS: Source Code Repository
Bugzilla: Defect Tracking System
StarTeam: Source Code Repository, Change Request Tracking, and Task Provider

Related Tasks

Administering the Liferay Portal
Opening the TeamInsight Viewer and the Liferay Portal
Changing Your Passwords for the TeamInsight Tools
Adding Mylyn Repositories for Bugzilla and XPlanner

Related Reference

The Bugzilla Guide
QALab Introduction
JBoss Kosmos
What is Cobertura?
PMD

Subversion: Source Code Repository

Subversion (as **Subclipse**) is the source code version control manager. TeamInsight also provides a read-only browser for the Subversion repository.

The Subversion Repository Resides on the Server

During installation by ProjectAssist, the Administrator creates the Subversion repository for the development project. All team members can access the source code repository simultaneously. Subversion maintains full copies of all previous versions of the project.

The top level in the project directory typically contains subdirectories named **trunk**, **branches**, and **tags**. Subversion manages branching in the repository.

Team Members Use Edit-Update-Commit Work Pattern

Subversion, as **Subclipse**, is the version control tool in Eclipse and JBuilder 2008. Subversion allows users to check out copies of the same files and projects. Team members use an Edit-Update-Commit work pattern.

To check out a project, team members click **Project** ▶ **Checkout Project**. Checking out a project creates a private local copy for the team member. After making changes to the source, the team member uses the right-click menu command **Team** ▶ **Update** to synchronize the private copies with the repository. Finally, the team member checks in the files to the repository using **Team** ▶ **Commit**, and Subversion synchronizes the repository.

Tip: For online help on **Subclipse**, click **Help** ▶ **Help Contents** ▶ **Subclipse**.

Subversion Viewer Provides Read-Only Access to the Repository

TeamInsight provides an open-source, read-only browser for Subversion repositories. You access the Subversion Viewer through the Liferay web portal.

The Subversion Viewer enables you to:

- Browse and download files for specific revisions
- Search the current directory and below, including CamelCaseSearch
- View the log of changes
- View the current file locks
- Diff files between revisions or directories
- View the directory flattened into one level

For easy viewing, the viewer highlights source code using **JHighlight**. You can also customize the style sheets that the viewer uses.

Related Concepts

Continuum/Maven: Continuous Build System

Related Tasks

Using the Subversion Viewer for Browsing the Project Repository
Checking Out a Project, Making Changes, and Checking Your Changes Into the Repository

Related Reference

Subclipse Online Help

CVS: Source Code Repository

CVS (Concurrent Versions System) is an open-source code version control manager. An existing CVS installation can be assimilated during the ProjectAssist server installation of the TeamInsight components.

CVS permits team members access to the source code repository simultaneously. CVS maintains copies of all previous versions of a project.

Assimilation of an existing CVS installation into the ProjectAssist component structure allows:

- Addition of Mavenized projects to existing CVS repository.
- Integration with Continuum for continuous builds and metrics (with a cvs.exe, or equivalent executable file, on the path of the machine running Continuum).
- Accessibility to users of new projects added via the TeamInsight .ticx file.
- Integration of CVS repository statistics in TeamInsight Viewer portal (if CVS repository history is enabled).

The CVS Repository Resides on a Server

During installation by ProjectAssist, the Administrator can assimilate an existing CVS server repository for the project. The ProjectAssist Administrator must have a user account on the assimilated CVS with read and write access in order to add projects. The ProjectAssist Administrator cannot add or modify users on the CVS installation. Users should have their own CVS for repository access.

Using an assimilated CVS installation also requires a secure CVS PServer port for anonymous CVS access.

Team Members Use Edit-Update-Commit Work Pattern

CVS is one of the version control options available in Eclipse and JBuilder 2008. CVS allows users to check out copies of the same files and projects. Team members use an Edit-Update-Commit work pattern.

Individuals without CVS accounts may be able to check out projects by pulling them anonymously but they cannot check in any changes. (Refer to the following **Importing the TeamInsight .ticx file** subtopic.

To check out a project, team members click **Project** ▸ **Checkout Project**. Checking out a project creates a private local copy for the team member. After making changes to the source, the team member uses the right-click menu command **Team** ▸ **Update** to synchronize the private copies with the CVS server repository. Finally, the team member checks in the files to the repository using **Team** ▸ **Commit**, and CVS synchronizes the repository.

CVS Viewer Specified by URL

The **TeamInsight** Viewer allows you to specify a URL of a CVS viewer. Specify a URL for a pre-installed web view of the CVS repository. The URL is used as the CVS view in the TeamInsight viewer. The URL address may be left blank and no CVS view will be available through the TeamInsight Viewer.

Importing the TeamInsight .ticx file

Assimilating an existing CVS installation into the JBuilder 2008 ProjectAssist component stack and Importing the TeamInsight .ticx file to the team members local machine adds:

- TeamInsight viewer portal if there was a URL specified in the component configuration.
- Project check out from CVS repository. Because user names are embedded in PSERVER locations, an attempt is made to match an existing CVS repository location (as seen in the CVS Repository Exploring perspective).

Otherwise, if the project is checked out as anonymous, the user cannot check in any changes. Locations are matched by host in the following manner:

- **No Matching Locations:** The user is prompted for a user name and password. The user should enter existing CVS credentials to check out with read/write access (if this is the access of the user's existing credentials). Otherwise, a user may pull files as an anonymous user. In this case, a new CVS location is created.
- **Non-anonymous Match:** The user's existing name is used. A password prompt is issued if that location does not have its password saved, and has not been accessed during the current session.
- **Anonymous Match**: If there is only an anonymous match, this match is used.

You can change the user name of any CVS location saved by the IDE, and any projects that used that connection are changed to match. This allows user checkout as anonymous, and then a name change of the CVS location so the user can check in changes later.

For any project, under the project's CVS properties, you can "Change Sharing" to any compatible location at any time.

Related Concepts

Continuum/Maven: Continuous Build System

Related Tasks

Checking Out a Project, Making Changes, and Checking Your Changes Into the Repository

Related Reference

CVS Online Documentation
Installing CVS Secure PServer for Anonymous CVS Access

StarTeam: Source Code Repository, Change Request Tracking, and Task Provider

(Enterprise Edition) During installation of the ProjectAssist server and TeamInsight components, Borland's ALM **StarTeam** product can be assimilated as the source code version control manager, change request tracker, and/or task provider system.

Borland® StarTeam® is a configuration management tool, designed for coordinating and managing the entire software delivery process. StarTeam promotes team communication and collaboration through centralized control of project activities. It provides integrated requirements management, change and configuration management, project and task management, defect (referred to as change requests in StarTeam terms), and file versioning. The assimilation of a StarTeam installation into your TeamInsight component stack adds to the team-centric features of TeamInsight and ProjectAssist by further unifying teams within the centralized environment and allowing you to leverage your current StarTeam software.

StarTeam Assimilation through ProjectAssist

During installation by ProjectAssist, the Administrator can assimilate an existing StarTeam installation into the ProjectAssist component stack for use by all TeamInsight members.

The ProjectAssist Administrator selects the stack components to install during the ProjectAssist installation through the **New ProjectAssist File: Select Stack Components** dialog. With ProjectAssist 2008 and JBuilder 2008 Enterprise Edition, the TeamInsight tool choices are:

- Version Control System (CVS, StarTeam, or Subversion)
- Defect or Change Request Tracking system (Bugzilla or StarTeam)
- Continuous Build system (Continuum)
- Task Provider (StarTeam or XPlanner)

StarTeam or CVS assimilation through ProjectAssist requires pre-existing StarTeam or CVS servers.

Note: The StarTeam Cross Platform Client needs to be installed on the machine where the Continuum installation resides in order for Continuum builds to work with StarTeam.

Mylynized Views for StarTeam Tasks and Change Request Tracking

With assimilation of an existing StarTeam installation, TeamInsight tools users can add a Mylyn Connector for StarTeam tasks and change requests. TeamInsight members should go to the **Window** ▶ **Configure Mylyn** ▶ **projectname** menu path to create a Connector for StarTeam Change Requests and StarTeam Tasks, and to see queries of their StarTeam Change Requests and Tasks.

Related Concepts

ProjectAssist and TeamInsight Overview
Liferay: The TeamInsight Project Portal
Mylyn Concepts
Continuum/Maven: Continuous Build System

Related Tasks

Adding Mylyn Repositories for StarTeam Change Requests or Task Planning
Checking Out a Project, Making Changes, and Checking Your Changes Into the Repository

Related Reference

StarTeam Product Page

Continuum/Maven: Continuous Build System

ProjectAssist provides continuous integration of source code. Continuum, a continuous integration tool, builds and tests code on a regular basis. The continuous integration system monitors a source control repository (such as Subversion, CVS, or StarTeam) at regular intervals and can trigger a build when changes are made to the repository. A build can include anything from compiling and testing a single project to assembling and testing of a deliverable from multiple dependent projects. The Continuum tool ensures that a project build succeeds at any point in the development cycle by allowing immediate identification of defects.

The Continuum component of the ProjectAssist install works with Maven 2 projects. A Maven 2 project typically consists of multiple modules. Each module has its own pom.xml file. ProjectAssist determines whether a project is a Maven 2 project by detecting the existence of a pom.xml file in the root of the project. Maven projects can also be quickly created using the Maven archetype wizard.

Related Concepts

ProjectAssist and TeamInsight Overview
Subversion: Source Code Repository

Related Tasks

Using the Subversion Viewer for Browsing the Project Repository
Using Continuum/Maven for Continuous Integration Builds
Checking Out a Project, Making Changes, and Checking Your Changes Into the Repository

Related Reference

Maven Project from Archetype
Continuum Online Resources and Documents
Maven Online Resources and Documents

Mylyn Concepts

Mylyn is a task-focused user interface available with Eclipse. Mylyn makes multi-tasking easier thus reducing information overload. Task and bug information is integrated into repositories that offer offline editing for ease of use and increased productivity.

Once tasks and bugs are integrated into the Mylyn view, Mylyn monitors work activity to identify information relevant to the tasks or bugs. Mylyn uses this context to filter information, providing only the related and useful information in the user interface. The information you need to do your job efficiently is right at your fingertips through the Mylyn view of the Task List.

The Mylyn view of the Task List allows the definition of queries against task or bug repositories that use a Mylyn connector. Mylyn provides task-focused user capabilities for JIRA, Bugzilla, Trac, and generic Web repository. JBuilder 2008 adds Mylyn support for StarTeam and XPlanner, in addition to the generic Mylyn plug-ins. After you activate a task, Mylyn remembers the context of your subsequent work, such as the files associated with the active task. Later when you return to the task, the preserved context enables you to work more efficiently.

JBuilder 2008 enables you to automatically add Bugzilla and XPlanner repositories (and also your own bug/task queries) to the Eclipse **Task List** view with Mylyn. With assimilation of an existing StarTeam installation, you can add a Mylyn repository to the **Task List** view for StarTeam tasks and bugs, and define queries against those tasks and bugs.

Using Mylyn, you can:

- Connect to task- or bug-tracking repository
- Define a query against the repository so that bugs or tasks are represented as Mylyn tasks in the development environment
- Define tasks related to the repository
- View task or bug reports locally or in an embedded browser
- Activate tasks and focus on the active task
- Save task context, including files and file hierarchy
- Work with tasks offline and synchronizes with the repository at a later time

Related Concepts

ProjectAssist and TeamInsight Overview
Bugzilla: Defect Tracking System
StarTeam: Source Code Repository, Change Request Tracking, and Task Provider
XPlanner: Project and Team Management

Related Tasks

Configuring Your TeamInsight Client
Adding Mylyn Repositories for Bugzilla and XPlanner
Adding Mylyn Repositories for StarTeam Change Requests or Task Planning

Related Reference

External Documentation for Mylyn from Eclipse.org
External Documentation about Mylyn Connectors to Repositories
External Article: Task-Focused Programming with Mylyn

Bugzilla: Defect Tracking System

Bugzilla is a bug tracking system that allows development teams to report, track and repair defects in their products while remaining team connected. It is a powerful tool that allows effective team organization and communication. Bugzilla also expands the use of the term "bug" to track other things such as feature requests so it can also be used for some general project tracking.

Bugzilla allows team members to:

- Track bugs and code changes
- Communicate easily with teammates regarding defects and bug status
- Submit and review patches to the product code
- Manage quality assurance (QA) in a coordinated fashion

Bugzilla can be opened in either the **Bugzilla**TeamInsight Viewer or through a web browser. Bugzilla Status, a portlet on the Liferay project portal, lists several categories of bugs (important, severity, and so forth) and includes a link to the Bugzilla server component.

The Bugzilla Repository Resides on the Server

During ProjectAssist installation and definition of TeamInsight projects, the Administrator creates the Bugzilla repository for the development project. All team members can access the bug tracking repository simultaneously when they have installed the TeamInsight client on their machines.

Bugzilla is web-based and relies on an installed web server (Apache) and a database (MySQL) . These required services are also installed during the ProjectAssist server install.

Note: Initially, all defined users are assigned the same password by the ProjectAssist Administrator. Each user must login to Bugzilla and change this initial password to a more secure password.

Bugzilla and Mylar

JBuilder 2008 enables you to add the Bugzilla repository bugs a to the Eclipse **Task List** view, and to use Mylar to define queries against those bugs. Refer to the Mylar concepts and tasks for more information.

Related Concepts

ProjectAssist and TeamInsight Overview
Liferay: The TeamInsight Project Portal
Bugzilla: Defect Tracking System
Mylyn Concepts

Related Tasks

Adding Mylyn Repositories for Bugzilla and XPlanner
Logging in to TeamInsight Bugzilla
Creating or Generating Bug Reports in Bugzilla
Querying Bugzilla for Bug Reports
Managing Bug Reports in Bugzilla

Related Reference

Bugzilla Resources and Documents
The Bugzilla Guide

XPlanner: Project and Team Management

XPlanner is an open-source Web-based planning and monitoring tool designed for projects that use eXtreme programming (XP) or **agile project management**.

Agile Projects Work Well in XPlanner

The organization of XPlanner is similar to the organization of agile projects. XPlanner contains pages for Projects, Iterations, Stories, and Tasks.

In Agile project management, the development cycle is divided into short **iterations** or **sprints**. During each iteration, team members complete a coherent increment of the final project. Each iteration is typically one to four weeks long. The team plans each product feature by creating one or more **user stories** that describe the feature. Within each story, the team then creates tasks that describe the work required to complete the feature.

Team members commit to completing a specific set of tasks, which are defined in each iteration. Team members meet on a frequent, often daily, basis in a **scrum** meeting to report progress they've made and to discuss obstacles blocking their way. Team leaders, known as scrum masters, aim to ensure that the team succeeds in completing the assigned tasks.

Projects Contain Iterations

The **Top** page in XPlanner lists the projects defined in XPlanner, their selectable IDs, and the name of the initial iteration in the project. Only the Administrator for ProjectAssist can create projects that are linked using the TeamInsight tools.

On a **Project** page in XPlanner, you see a list of the iterations defined in the project, along with their scheduled start and stop dates, and a list of the stories defined for each iteration.

Iterations Contain User Stories

For each project in XPlanner, you can create any number of iterations or sprints.

Iterations or **sprints** are coherent increments of the total work in a project. Iterations can include backlog, future plans, current sprints, and past or future sprints. Each sprint lasts a scheduled amount of time in which the team completes tasks that have been assigned to them in XPlanner. You can define an iteration without starting it and then return to start the iteration later.

On the **Iteration** page in XPlanner, you see a list of the defined user stories associated with that iteration.

User Stories Contain Tasks

The team divides their work on a project work into user stories (descriptions of features in the final product) and tasks (the steps required to create a feature). On the **Story** page in XPlanner, you see a list of the tasks defined for that story.

User Stories Describe Project Features

For each planned feature, the team creates one or more user stories in XPlanner.

Tasks Describe the Steps in Creating the Feature

For each user story, the team creates one or more tasks in XPlanner and estimates the time required to complete the task. During each iteration, the team tracks the time they spend on each task and marks tasks they have completed.

Tree Structure Aids Navigation in XPlanner

The **tree structure** located at the top of XPlanner windows displays your location in the XPlanner database. For example, the **Top** page does not display a tree structure at all, because you have not opened the database yet. However, after you open a project, an iteration, and a story, the tree structure on the **Story** page displays the names of the project, iteration, and story.

Related Concepts

ProjectAssist and TeamInsight Overview
Mylyn Concepts

Related Tasks

Adding Mylyn Repositories for Bugzilla and XPlanner
Adding Team Members in XPlanner (Administrator Task)
Creating and Starting Project Iterations in XPlanner (Administrator Task)
Planning a Product Feature: Creating a User Story in XPlanner
Planning Your Work: Creating Tasks in XPlanner
Tracking Your Time and Completing Tasks in XPlanner
Moving or Continuing a Story or Task in XPlanner
Monitoring Iteration Metrics in XPlanner

Related Reference

XPlanner Documentation Available from XPlanner.org

Working with Peers

The peer to peer feature in JBuilder allows you to chat with peers and share data. You can share projects through a repository. You can also set up contact groups to effectively collaborate with a group of peers.

In This Section

Peer to Peer Collaboration

Peer to peer collaboration features allow two or more users to collaborate across a local area network (LAN) and send data. Peers are discovered automatically when they are on the same LAN. You can collaborate with peers who are using JBuilder 2008, as well as with peers using JBuilder 2006 or any later version.

You enable the peer to peer subsystem on the **Peer to Peer** page of the **Preferences** dialog box (**Windows ▸ Preferences ▸ Peer to Peer**).

You open the Peers view through the menu path (**Window ▸ Show View ▸ Other ▸ Peer to Peer ▸ Peers**). The Peers View shows you peers who are currently online. You can change your user status in the Peers view by clicking on your name and using the dropdown status menu. Peers are displayed in the Peers pane, on the left side of the Peers view.

Chatting with Peers

You use the peer to peer feature to chat with peers on your LAN. You chat with peers in the panes on the right side of the Peers pane after opening a session. A record of each chat, the chat log, is maintained and written to a file. The default location for the chat log directory is set on the **Peer to Peer** page of the **Preferences** dialog box. One chat log, with all sessions recorded, is maintained for each peer. You can view or delete the log at any time.

Note: The messages are recorded in the chat log of each individual member of the collaboration session.

The Collaboration pane displays the running chat, as well as links to files, stack traces, web pages, or version control system (VCS) links that have been sent to you by a peer.

Open the chat log in the Peers View by right-clicking and selecting the View Chat Log context menu, The chat log is UTF-8 encoded. If a peer is using an international locale and fonts, and you are using the US locale and fonts, the peer may not display correctly in the Peers pane. However, this information is still saved correctly in the log file. The filename for the chat log may be corrupted as well, since the peer name is part of the filename. If you view the chat log file on a machine with the appropriate fonts, the filename and contents of the file will display correctly.

Collaborating with Contact Groups

You can use contact groups to organize your peer list. For example, you could create a group of people working on specific product features. Then, instead of selecting each member individually, you can select the group to open a session. You can add and remove groups and peers within groups. One peer can appear in multiple groups.

Sharing Projects with Peers

You share projects through a repository. To share a project, you can send the VCS link to a peer. The VCS link contains an identifier for the VCS plug-in, a reference to the VCS location for the project, and the name of the project. Your peer opens the VCS link to automatically check out the project locally.

Peers View

The Peers view consists of the Peers pane and the Collaboration pane. The Peers pane, on the left side of the view, shows your status (Available, Away, or Offline), an informational node with your IP address, the available peers, as well as any contact groups you have created.

Individual tabbed pages in the Collaboration pane, on the right side of the view, show the peer(s) you are chatting with and the chat. Data that you have sent is displayed in this pane, as well as links that you have received (to files, web pages, stack traces, or VCS links to a project). Tooltips in the Peers pane display the peer's user name, associated icon, IP address, status, and description.

Note: If a peer is using an international locale and fonts, and you are using the US locale and fonts, the peer may not display correctly in the Peers pane.

Eclipse Wiki

As part of the team collaboration features of JBuilder 2008, a Wiki for team members is now available inside the IDE. A project Wiki can be a powerful project documentation tool for team members to coordinate efforts, disperse unified project information, and take full advantage of the open-source community. Files with a .wiki extension become part of the project. Editing inside the IDE provides automatic page creation and dynamic linking (to other pages and Java source in the workspace). Checking the .wiki files into the version control system permits team members that checkout the project to view and edit the wiki. Wiki content can also be exported into a set of static web pages, including coversion of Java files into HTML.

The Eclipse Wiki plug-in is provided with JBuilder 2008. However, it must be enabled and configured according to documentation provided on the web or in the Eclipse Wiki Help files included with JBuilder 2008.

Note: Refer to the Eclipse Wiki Help files and the following "Related Reference" link for more information on enabling and configuring your Wiki plugin.

Related Tasks

Enabling Peer to Peer Collaboration
Opening a Peer to Peer Session
Managing Contact Groups
Chatting with Peers
Sharing Team-Enabled Projects with Peers
Sending Data To Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link
Eclipse Wiki Editor Plugin Web Information

Procedures

Tasks

This section lists all of the task-oriented help topics included with this CodeGear product.

In This Section

JBuilder Project Migration

The migration path from a legacy JBuilder to Eclipse allows you to import every type of JBuilder project into the Eclipse workspace, including Java SE projects, Java EE projects, VisiBroker projects, RMI/JNI projects, and project groups, as well as projects that are under source control. The migration allows you to develop, test, and run your project in Eclipse.

In This Section

Building an Imported Project

JBuilder 2008 builds using a standard JDK compiler. This topic describes the auto-build feature, manually building an imported project, change build order and changing the output path.

To configure compiler options:

1. Right click the project in the **Navigator** ▸ **View** and select **Properties** ▸ **Java Compiler**.
2. Configure the compiler and the project preferences. Click **Apply** and **OK** to exit project properties.

To deactivate auto-build and enable manual build:

1. From the workbench click **Project**. If a checkmark is visible next to **Build Automatically**, click **Build Automatically** once to deactivate.
2. If no checkmark is visible it is already deactivated and manual builds may occur.

Note: The auto-build feature is on by default for new or imported projects. When auto-build is on, builds occur after every set of resource changes, to keep the `.class` file updated. When auto-build is off, builds must be invoked manually.

To build an imported project:

1. From the workbench select **Project** ▸ **Build Project** to perform an incremental build on the selected project.
2. Choose **Project** ▸ **Build All** to incrementally build all open projects.
3. Choose **Project** ▸ **Clean** to delete all previous build output for the selected project. If auto-build is on, a full build is invoked.

To change the build order:

1. Open the **Build Order** page of the **Preferences** dialog box by clicking **Window** ▸ **Preferences** ▸ **General** ▸ **Workspace** ▸ **Build Order**.
2. Deactivate the **Use Default Build Order** option.
3. Select the desired project and use the **Up** and **Down** buttons to rearrange the build order.
4. Click **OK** to close the dialog box and save the new build order.

To change the output path:

1. Select **Project** ▸ **Properties** ▸ **Java Build Path** to open the **Java Build Path** page of the **Properties** dialog box. .
2. Change the folder in the **Default Output Folder** field.
3. Click **OK** to close the dialog box and save the output path.

Related Concepts

Legacy JBuilder Project Migration Overview
Importing Legacy Projects

Related Tasks

Importing a Java EE Project From Legacy JBuilder
Importing a Java SE Project From Legacy JBuilder
Setting Import Properties
Building an Imported Project
Running an Imported Project

Related Reference

Eclipse Help Topic “Java builder”

Importing a Java EE Project From Legacy JBuilder

Java EE is supported in legacy JBuilder versions via the creation of Java EE modules in a single JBuilder project with shared source code. JBuilder and JGear products are based on the Eclipse framework that supports the WTP model. The WTP model requires the creation of a project for each module. The modules table in the second page of the wizard lists the Java EE modules found in legacy JBuilder (JBuilder 2006 and before) projects including Java EE, Java versions, and the corresponding JBuilder or JGear project created during the conversion process. Click on each row in the module table to display the Java and Java EE versions (project facets) for a JBuilder or JGear project.

To set up the runtime server:

1 From the workbench select **Window** ▸ **Preferences** ▸ **Server** ▸ **Installed Runtimes** and click **Add**.

2 Select a runtime to correspond with the **Server** set up in the `.jpx` project to be imported.

> **Note:** To learn how to add a runtime server see "Related Procedures."

3 Set the **Application Server Home Directory**, select the **JRE**, and click **OK**.

To activate the Java EE Project Import Wizard:

1 From the workbench select **File** ▸ **New** ▸ **Project** ▸ **Legacy JBuilder** ▸ **Java EE Project from Existing JBuilder .jpx Project**.

> **Note:** This wizard can also be accessed from **File** ▸ **Import**.

2 Select **Browse** to locate the `.jpx` file.

To import libraries:

1 The legacy JBuilder home directory contains file and libraries needed to properly import the project. If the default entry does not point to the correct `.jbuilder` directory, click **Browse** to locate it.

2 Review the **Library Status** table. If each library required for the import has a **green checkmark** next to it, click **Next** and continue to step 4.

If any of the libraries has a **red X** next to it the library with the required directory references could not be located. In this case, go to step 3.

3 Add additional directories to be searched by selecting **Add** . Browse to the desired resource and click **Next**.

4 Click **Browse** to locate and select the runtime server.

To set project settings:

1 Review the **Modules** table to see each imported module is related to a project in the imported project.

2 Accept the default **Project Settings** to create a new utility module. Select the **Java Version** and go to step 3.

> **Note:** The option to create a utility project is automatically selected when a legacy JBuilder project containing more than one Java EE module is selected. A utility project is a Java project containing source code for all Java EE projects in a the workspace. This is the recommended conversion option when importing a legacy JBuilder project containing multiple modules to prevent the duplication of source code in the Java EE projects. Creating a utility project also

allows the creation of a EAR project if the legacy project does not contain a EAR module. The EAR project is automatically include all Java EE projects, including the utility project. The utility project is included as a classpath dependency in EJB projects via the J2EE Module Dependencies properties for the EJB project. It is also included as a J2EE dependency for web projects resulting in the JAR created by the utility project being bundled into the resultant web archive's lib directory.

3 Select the EJB project in the modules table and select**EJB 3.0** from the drop down options at the bottom of the wizard and click **Finish**.

Note: JBuilder and JGear products support **EJB 2.x** development using XDoclet annotations and **EJB 3.0** development using **Java EE 5.0** annotations. Legacy JBuilder projects containing XML descriptors are converted to either XDoclet annotations (for **EJB 2.1**) or to **Java EE 5.0** annotations (for **EJB 3.0**). The import wizard allows conversions from **EJB 2.1** to **EJB 3.0** using existing XML descriptors.**EJB 2.1** interfaces are not to be copied over to the EJB project (or the utility project) from the legacy JBuilder project since interfaces are generated using XDoclet.

The project is now converted and project files are available in the **Navigator** view.

Warning: Migrating large projects can be time and memory intensive. Close all unnecessary applications before migrating a large project.

Related Concepts

Legacy JBuilder Project Migration Overview

Related Tasks

JBuilder Project Migration
Setting Import Properties
Setting Up a Runtime Server
Building an Imported Project
Running an Imported Project
Importing a Java SE Project From Legacy JBuilder

Importing a Java SE Project From Legacy JBuilder

These tasks describe the steps to import a Legacy JBuilder project to a Java SE project.

To activate theJavaSE Project Import Wizard:

1 From the workbench select **File** ▸ **New** ▸ **Project** ▸ **Legacy JBuilder** ▸ **Java Project from Existing JBuilder .jpx Project**.

Tip: This wizard can also be accessed from **File** ▸ **Import**.

2 Select **Browse** to locate the `.jpx` file.

To import projects:

1 The Legacy JBuilder home directory contains file and libraries needed to properly import the project. If the default entry does not point to the correct `.jbuilder` directory, click **Browse** to locate it.

2 Review the **Library Status** table. If each library required for the import has a **green checkmark** next to it, click **Next** and continue to step 4.

If any of the libraries has a **red X** next to it the directories could not be located. In this case, go to step 3.

3 Add additional directories to be searched by selecting **Add**. Browse to the desired resource and click **Next**.

4 Accept the default **Project Settings** and click **Finish**.

The following legacy artifacts are imported:

- Libraries
- Runtime Configurations
- Javadoc Options
- Java Compiler Options default file encoding
- Java files

Related Concepts

Legacy JBuilder Project Migration Overview

Related Tasks

JBuilder Project Migration
Setting Import Properties
Setting Up a Runtime Server
Building an Imported Project
Running an Imported Project
Importing a Java EE Project From Legacy JBuilder

Importing a Legacy Java RMI/JNI Project

This task describes the steps to import a legacy RMI/JNI project.

To import a Java RMI/JNI project from legacy JBuilder:

1. Use the required steps to import the project (see Related Procedures).
2. Expand the project in the **Package Explorer** and select an **RMI** or **JNI** file.
3. Right-click the file and choose **Properties** to display the **Properties for <filename>** dialog box.
4. Choose the **RMI/JNI Properties** page to view imported property settings.

- **RMI** options are set in the **RMI Compiler Settings** area of the dialog box.
- **JNI** options are set in the **JNI Compiler Settings** area of the dialog box.

Related Concepts

Legacy JBuilder Project Migration Overview

Related Tasks

Importing a Java SE Project From Legacy JBuilder
Importing a Java EE Project From Legacy JBuilder
Importing a Legacy Java VisiBroker Project
Setting Import Properties
JBuilder Project Migration

Importing a Legacy Java VisiBroker Project

This task describes how to import a legacy Java VisiBroker project.

To import a Legacy Java VisiBroker project:

1 Use the required steps to import the project (see Related Procedures).

2 Select **Window** ▸ **Preferences** ▸ **VisiBroker** to open the **VisiBroker** page of the **Preferences** dialog box.

3 Verify the directory where VisiBroker tools are installed.

4 Click **Apply** and **OK** to save the settings.

5 Expand the project in the **Package Explorer** and select an **IDL** file or a Java interface file to translate from **IDL**, to **IDL**, or **IIOP**.

6 Right-click the file and choose **Properties** to display the **Properties for <filename>** dialog box.

- For IDL to Java files, choose the **VisiBroker IDL Properties** page and verify options in the **IDL2Java Settings** area of the dialog box.
- For Java to IDL files, choose the **VisiBroker Java Properties** page and verify options in the **Java2IDL Settings** area of the dialog box.
- For Java to IIOP files, choose the **VisiBroker Java Properties** page and verify options in the **Java2IIOP Settings** area of the dialog box.

Related Concepts

Legacy JBuilder Project Migration Overview

Related Tasks

JBuilder Project Migration
Importing a Java SE Project From Legacy JBuilder
Importing a Java EE Project From Legacy JBuilder
Importing a Legacy Java RMI/JNI Project
Setting Import Properties

Importing a Source Controlled Project from a Legacy Version of JBuilder

Use the following steps to import a source controlled Legacy JBuilder project to JBuilder 2008.

To import a Java project from legacy JBuilder using source control:

1. Follow the steps to use the project import wizard for a Java EE, Java SE, Java RMI/JNI, or Java VisiBroker project (see **Related Procedures**), and add the following step for a source controlled project.
2. Click the **Enable VCS Plugin For This Project** option. Log onto the server to check out the project. The project is checked out into the Eclipse workspace.

   Warning: CVS and Subversion projects that are checked into a local repository cannot be checked out.

3. Click **Finish** to import or check out the project.

Related Concepts

Legacy JBuilder Project Migration Overview

Related Tasks

JBuilder Project Migration
Importing a Java SE Project From Legacy JBuilder
Importing a Java EE Project From Legacy JBuilder
Importing a Legacy Java VisiBroker Project
Importing a Legacy Java RMI/JNI Project
Setting Import Properties

Running an Imported Project

The run configuration is automatically imported when you import a project from a legacy JBuilder product version.

To run an imported project:

1 Select **Run** ▸ **Run** to open the **Run** dialog box.

2 Expand the node that matches the type of imported project in the **Configurations** list and choose the name of the configuration. Typically, the configuration name is the same as the project name.

3 Click the **Run** button to run the project.

Note: If the project uses macros in the run VM arguments, compile the project before importing it. Compilation expands the macros. If the project is not compiled, it will not run.

Related Concepts

Legacy JBuilder Project Migration Overview
Importing Legacy Projects

Related Tasks

JBuilder Project Migration
Importing a Java SE Project From Legacy JBuilder
Importing a Java EE Project From Legacy JBuilder
Setting Import Properties
Building an Imported Project
Running an Imported Project

Setting Import Properties

Before importing a Java EE or VisiBroker project, configuring the application server and VisiBroker locations, you need to set properties.

To set properties for importing Java EE projects:

1. From the workbench click **File Import**.
2. Click the J2 EE node and select from the following import file options:
   - App Client JAR file
   - EAR File
   - J2EE Utility JAR
   - RAR file
3. After selecting the import file click **Next**, and follow the prompts to complete the import properties configuration.

To set properties for VisiBroker project imports:

1. Select **Window** ▸ **Preferences** ▸ **VisiBroker** to open the **VisiBroker** page of the **Preferences** dialog box.
2. Enter the directory where VisiBroker tools are installed in the **VisiBroker Tools Directory** field. Typically, this is `bin` folder of the Borland Application Server installation.
3. Click **Apply** and **OK** to save project settings.
4. Select **Project** ▸ **Properties** ▸ **Builders** to open the **Builders** page of the **Properties** dialog box. Make sure the **VisiBroker Builder** option in the **Configure The Builders For This Project** list is selected.
5. Click **OK** to save project settings.

Related Concepts

Legacy JBuilder Project Migration Overview
Importing Legacy Projects

Related Tasks

JBuilder Project Migration
Importing a Java SE Project From Legacy JBuilder
Importing a Java EE Project From Legacy JBuilder
Setting Import Properties
Running an Imported Project
Building an Imported Project

Modeling Applications

Borland's Together modeling system allows you to create a visual model as you develop Java database applications. InterBase and JDataStore database systems are included as part of the development environment.

In This Section

Creating a Java Modeling Project

This section describes how to create a Java Modeling project.

To create a new Java modeling project in JBuilder or JGear:

1. Select **File** ▸ **New** ▸ **Project**.
2. Select **Modeling** ▸ **Java Modeling Project** from the list.
3. Enter a name for your new project.
4. Specify the Java build settings.
5. Click the **Finish** button.

Related Concepts

Modeling Applications Overview

Related Tasks

Creating a Modeling Project
Importing a Modeling Project
Importing a Java Project as a Java Modeling Project

Creating a Modeling Project

This section describes how to create a modeling project.

To create an empty modeling project in JBuilder or JGear:

1. Select **File** ▶ **New** ▶ **Project**.
2. Select the appropriate type of modeling project from the list.
3. Enter a name for your new project.
4. Click the **Finish** button.

Related Concepts

Modeling Applications Overview

Related Tasks

Creating a Java Modeling Project
Creating an Enterprise Java Bean (EJB) Modeling Project
Creating a Java Persistence API (JPA) Modeling Project
Importing a Modeling Project

Importing a Java Project as a Java Modeling Project

Use these steps to import a Java project as a Java Modeling project.

To import a new Java modeling project from a Java project:

1 Select **File** ▸ **New** ▸ **Project**.

2 Select **Modeling** ▸ **Java Modeling Projects from Java Projects** from the list.

3 Specify the Java project to import.

4 Click the **Finish** button.

Related Concepts

Modeling Applications Overview

Related Tasks

Creating a Modeling Project
Importing a Modeling Project
Creating a Java Modeling Project

Importing a Modeling Project

This section describes how to import a modeling project.

To import a modeling project into JBuilder or JGear:

1. Select **File** ▸ **New** ▸ **Project**.
2. Select the appropriate type of modeling project from the list.
3. Enter a name for the imported project.
4. Click the **Finish** button.

Related Concepts

Modeling Applications Overview

Related Tasks

Importing a Java Project as a Java Modeling Project
Importing an Enterprise Java Bean (EJB) Modeling Project
Creating a Modeling Project

Java EE Applications

Java EE components are assembled into an application and are deployed to production, to be run and managed by the Java EE server. Use the following links to discover detailed information about creating Java EE applications using JBuilder 2008.

In This Section

Developing Enterprise Java Bean (EJB) Applications

The tasks in this area provide information on how to work with Enterprise Java Bean (EJB) 2.x and 3.0 applications in JBuilder or JGear development environments.

In This Section

EJB Modeling Applications

Borland's Together modeling system allows you to create a visual model as you develop EJB applications. InterBase and JDataStore database systems are included as part of the development environment.

In This Section

Creating an EJB Modeling Project based on WTP XDoclet Project
Create an EJB modeling based on WTP XDoclet project..

Creating an Enterprise Java Bean (EJB) Modeling Project
Describes how to create a new Enterprise Java Bean (EJB) Modeling project.

Importing an EJB Modeling Project from a Java Project
Steps to import an EJB modeling project from an existing Java modeling project.

Importing an Enterprise Java Bean (EJB) Modeling Project
Describes how to import an Enterprise Java Bean (EJB) Modeling project.

Creating an EJB Modeling Project based on WTP XDoclet Project

The **Create an EJB Modeling Project with XDoclet Annotations** wizard converts an existing **Web Tools Platform (WTP) EJB** project to an **EJB** modeling project using **XDoclet** annotations.

Warning: The **WTP EJB** project must exist in the current Workspace and **XDoclet** annotation support must be installed and configured to work with the Workbench.

To create an EJB modeling project with XDoclet Annotations

1. Select **File** ▸ **New** ▸ **Project** to invoke the **New Project** wizard.
2. In the **Select a Wizard** window navigate to the **EJB** folder and select **EJB Modeling Project from an XDoclet Annotated WTP Project**, and click **Next**.
3. A list of **WTP EJB** projects in the current Workspace is displayed.

   Note: Only **WTP EJB** projects (not **EJB** modeling projects) are displayed.

4. Activate the checkbox next to the desired **EJB** project and click **Finish**.

The **WTP EJB** project is converted to an **EJB** modeling project and **EJB** diagrams are created based on **EJB** source and **XDoclet** annotations in the **WTP EJB** project.

Related Concepts

Legacy JBuilder Project Migration Overview

Related Tasks

Setting Import Properties
Building an Imported Project
Enabling XDoclet

Related Reference

Creating Enterprise Beans with XDoclet Annotation Support

Creating an Enterprise Java Bean (EJB) Modeling Project

This section describes how to create a new Enterprise Java Bean (EJB) Modeling project.

To create a new EJB modeling project in JBuilder or JGears:

1. Select **File** ▸ **New** ▸ **Project**.
2. Select **EJB** ▸ **EJB Modeling Project** from the list.
3. Enter a name for the new project.
4. Select a target runtime and project configuration for the project.
5. Click the **Finish** button.

Related Concepts

Modeling Applications Overview

Related Tasks

Creating a Modeling Project
Importing a Modeling Project
Importing an Enterprise Java Bean (EJB) Modeling Project

Importing an EJB Modeling Project from a Java Project

Use the **EJB Modeling Project from a Java Project** wizard to import existing **EJB** sources and **XML** descriptors from an Eclipse Java project. The **XML** descriptors can be converted to **EJB 2.x XDoclet** annotations or to **EJB 3.0** annotations.

Note: The following steps assume a correctly configured web application server. For steps to install the **JBoss** web application server see **Related Procedures**.

Tip: A new **EJB** modeling project is created based on the source and descriptors from the Java project. The Java project can exist anywhere on the hard disk.

To import an EJB modeling project from an existing Java project:

1. Place the **XML** descriptors in a folder named **META-INF** and make sure the folder is located in the project source directory.
2. Select **File** ▶ **New** ▶ **Project** to invoke the **New Project** wizard.
3. Navigate to the **EJB** folder, select **EJB Modeling Project from Java Project**, and click **Next**.
4. Select the desired Java project and click **Next**.
5. Name the new **EJB** modeling project.
6. Set the **Target Runtime** and click **Next**.

   Warning: Create a new runtime where an existing runtime is not already installed.

7. Set the **EJB** and Java versions for the converted project.

   XDoclet annotations based on **XML** descriptors are generated for **EJB 2.1**.

   Java EE 5.0 annotations based on **XML** descriptors are generated for **EJB 3.0**.

   Note: The XML descriptors must be located in a folder named **META-INF**.

8. Click **Next**.
9. Accept or customize the remaining configuration settings, and click **Finish**.

Related Concepts

Legacy JBuilder Project Migration Overview

Related Tasks

Setting Import Properties
Building an Imported Project
Creating an EJB Modeling Project based on WTP XDoclet Project
Setting Up a Runtime Server

Importing an Enterprise Java Bean (EJB) Modeling Project

This section describes how to import an EJB modeling project from a Java project or from an Xdoclet-annotated WTP project.

To import an EJB modeling project from a Java project:

1. Select **File** ▸ **New** ▸ **Project**.
2. Select **EJB** ▸ **EJB Modeling Project from Java Project** from the list.
3. Enter a name for your new project.
4. Select the project to import.
5. Select a target runtime and project configuration for the project.
6. Click the **Finish** button.

To import an EJB modeling project from an Xdoclet-annotated WTP project:

1. Select **File** ▸ **New** ▸ **Project**.
2. Select **EJB** ▸ **EJB Modeling Project from Xdoclet annotated WTP project** from the list.
3. Enter a name for your new project.
4. Select the project to import.
5. Select a target runtime and project configuration for the project.
6. Click the **Finish** button.

Related Concepts

Modeling Applications Overview

Related Tasks

Creating a Modeling Project
Importing a Modeling Project

Enterprise Java Bean (EJB) 2x-Specific Tasks

This section provides information on specific tasks for working with Enterprise Java Bean (EJB) 2.x applications in JBuilder or JGear development environments.

In This Section

Adding a Create Method to an EJB 2.x Entity Bean

This section describes how to add a create method to an EJB 2.x entity bean.

Note: The EJB 3.0 specification has eliminated this interface for entity beans.

To add a create method to an entity bean in the Modeling Perspective:

1. Open the class diagram for the entity bean.
2. Right click on the entity bean.
3. Select **New** ▸ **Create Method**.
4. Click twice on the new method to open the in-place editor.
5. Enter the name and return type of the new create method.

To add a create method to an entity bean using the Code Editor:

1. Open source code for the entity bean.
2. Add the new create method directly to the source code.
3. Add annotations.
4. For EJB 2.x, add code to expose the create method in the home interface.
5. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Find Method to an EJB 2.x Entity Bean
Adding a Home Method to an EJB 2.x Entity Bean
Adding a Select Method to an EJB 2.x Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a Find Method to an EJB 2.x Entity Bean

This section describes how to add a find method to an entity bean.

Note: The EJB 3.0 specification has replaced this method with named queries.

To add a find method to an entity bean in the Modeling Perspective:

1. Open the class diagram for the entity bean.
2. Right click on the entity bean.
3. Select **New** ▸ **Find Method**.
4. Click twice on the new method to open the in-place editor.
5. Enter the name, query, and return type of the new find method.

To add a find method to an entity bean using the Code Editor:

1. Open source code for the entity bean.
2. Add the new find method directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Named Query to an EJB 3.0 Entity Bean
Adding a New Method to an EJB
Adding a Create Method to an EJB 2.x Entity Bean
Adding a Home Method to an EJB 2.x Entity Bean
Adding a Select Method to an EJB 2.x Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a Home Method to an EJB 2.x Entity Bean

This section describes how to add a home method to an EJB 2.x entity bean.

Note: The EJB 3.0 specification has eliminated this interface for Entity beans.

To add a home method to an EJB 2.x entity bean in the Modeling Perspective:

1. Open the class diagram for the session bean.
2. Right click on the entity bean.
3. Select **New** ▸ **Home Method**.
4. Click twice on the new method to open the in-place editor.
5. Enter the name and return type of the new home method.

To add a home method to an EJB 2.x entity bean using the Code Editor:

1. Open source code for the entity bean.
2. Add the new home method directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Session Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Home Method to an EJB 2.x Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a Select Method to an EJB 2.x Entity Bean

This section describes how to add a select method to an EJB 2.x entity bean.

Note: The EJB 3.0 specification has eliminated this method for entity beans.

To add a select method to an entity bean in the Modeling Perspective:

1. Open the class diagram for the entity bean.
2. Right click on the entity bean.
3. Select **New** ▸ **Select Method**.
4. Click twice on the new method to open the in-place editor.
5. Enter the name, query, and return type of the new select method.

To add a select method to an entity bean using the Code Editor:

1. Open source code for the entity bean.
2. Add the new select method directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Create Method to an EJB 2.x Entity Bean
Adding a Find Method to an EJB 2.x Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Creating a Bean-Managed-Persistence (BMP) Entity Bean

This section describes how to create a new BMP entity bean.

To create a new BMP entity bean in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Select the **BMP entity bean** tool.
4. Place the BMP entity bean in the model.

To create a new BMP entity bean in the Code Editor:

1. Create a new Java file for your BMP entity bean.
2. Code the BMP entity bean by hand.
3. Add annotations.
4. Add the new BMP entity bean source file to your project.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Importing Entity Beans from a Database

Creating a Container-Managed-Persistence (CMP) Entity Bean

This section describes how to create a new container-managed-persistence (CMP) entity bean.

To create a new CMP entity bean in the Modeling Perspective:

1. Open the modeling perspective.
2. Bring up the model for your EJB project.
3. Select the **CMP entity bean** tool.
4. Place the CMP entity bean in the model.

To create a new CMP entity bean in the Code Editor:

1. Create a new Java file for your CMP entity bean.
2. Code the CMP entity bean by hand.
3. Add annotations.
4. Add the new CMP entity bean source file to your project.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Importing Entity Beans from a Database

Enterprise Java Bean (EJB) Generic Tasks—2.x or 3.0

This section provides information on specific tasks for working with Enterprise Java Bean (EJB) 2.*x* or 3.0 applications in JBuilder or JGear development environments.

In This Section

Importing Entity Beans from a Database
Describes how to import database tables into an EJB project as entity beans.

Modifying an Enterprise Java Bean (EJB)
Describes how to modify an enterprise java bean (EJB).

Removing an Enterprise Java Bean (EJB)
Describes how to remove an Enterprise Java Bean (EJB).

Viewing the Source Code of an Enterprise Java Bean (EJB)
Describes how to view the source code of an Enterprise Java Bean (EJB).

Adding a Business Method to an EJB

This section describes how to add a business method to an entity bean or a session bean.

To add a business method to an EJB in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the session bean.
3. Select **New** ▸ **Business Method**.
4. Click on the new method to view its properties.
5. Select properties for your new business method.

To add a business method to an EJB using the Code Editor:

1. Open the source code for the EJB.
2. Add the new business method directly to the source code.
3. Add annotations.
4. For EJB 2.x projects, add local and remote setting interfaces.
5. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a CMP Field to a CMP Entity Bean

This section describes how to add a new Container-Managed Persistence (CMP) field to a CMP entity bean using either the Modeling Perspective or the Code Editor.

A CMP field is a virtual field in an entity bean. A CMP field refers to a column in a database table, and the entity bean implements getters and setter methods for the field.

To add a new CMP field to a CMP entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **CMP Field**.
5. Enter the name of the new field.
6. Click in the new field to view and set its properties.

To add a CMP field to a CMP entity bean to an EJB using the Code Editor:

1. Open the source code for the EJB.
2. Add the new field directly to the source code.
3. Add annotations to the source code
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Field from an Enterprise Java Bean (EJB)

Adding a New Method to an EJB

This section describes how to add a new method to an EJB using either the Modeling Perspective or the Code Editor.

Refer to the links at the bottom of this page for information on how to add a business, pre-persist, pre-remove, pre-update, post-persist, post-remove, post-update, or post-load method to a 3.0 EJB. Refer to the links at the bottom of this page for information on how to add a business, create, home, find, or select method to a 2.x EJB.

Note: The EJB 3.0 specification is quite different from the EJB 2.x specification. JBuilder 2008 provides support for both EJB 2.x and EJB 3.0 methods. Make sure that you are using the correct methods for your version of EJB.

To add a new method to an EJB in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Operation** or the type of method to be added.
5. Click twice on the new method in the diagram.
6. Enter the name and return type of the new method.

To add a new method to an EJB using the Code Editor:

1. Open source code for the EJB.
2. Add the new method and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a Business Method to an EJB
Adding an Interceptor Method to an EJB 3.0 Session Bean
Adding a Post-Construct Method to an EJB 3.0 Session Bean
Adding a Pre-Destroy Method to an EJB 3.0 Session Bean
Adding a Timeout Method to an EJB 3.0 Session Bean
Adding a New Pre-Update Method to an EJB 3.0 Entity Bean
Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean
Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean
Adding a New Post-Update Method to an EJB 3.0 Entity Bean
Adding a New Post-Load Method to an EJB 3.0 Entity Bean
Adding a New Post-Persist Method to an EJB 3.0 Entity Bean
Adding a New Post-Remove Method to an EJB 3.0 Entity Bean
Adding a Create Method to an EJB 2.x Entity Bean
Adding a Find Method to an EJB 2.x Entity Bean
Adding a Home Method to an EJB 2.x Entity Bean
Adding a Select Method to an EJB 2.x Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Creating a Message Bean

This section describes how to create a new message bean using either the Modeling Perspective or the Code Editor.

To create a new message bean in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Select the **Message Bean** tool.
4. Place the message bean in the model.
5. Define the message destination and message destination link for the message bean.

To create a new message bean in the Code Editor:

1. Create a new Java file for your message bean.
2. Code the message bean by hand.
3. Add annotations.
4. Add the new message bean source file to your project.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Message Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Creating a Message Destination for a Message Bean
Creating a Message Destination Link for a Message Bean

Creating a Message Destination for a Message Bean

This section describes how to create a message destination for a message bean.

To create a new message bean destination in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Select the **Message Bean Destination** tool.
4. Place the message bean destination in the model.

Related Concepts

Enterprise Java Beans (EJB) Overview
Message Bean Overview

Related Tasks

Creating a Message Bean
Creating a Message Destination Link for a Message Bean

Creating a Message Destination Link for a Message Bean

This section describes how to create a message destination link for a message bean.

To create a new message bean destination link in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Verify the existence of the message bean and message bean destination.
4. Select the **Message Bean Destination Link** tool.
5. Link the message bean to the message bean destination.

Related Concepts

Enterprise Java Beans (EJB) Overview
Message Bean Overview

Related Tasks

Creating a Message Bean
Creating a Message Destination for a Message Bean

Creating a New Session Bean

This section describes how to create a new session bean using either the Modeling Perspective or the Code Editor.

To create a new session bean in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Select the **Session Bean** tool.
4. Place the session bean in the model.

To create a new session bean in the Code Editor:

1. Create a new Java file for your session bean.
2. Code the session bean by hand.
3. Add annotations.
4. Add the new session bean source file to your project.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Session Bean Overview

Related Tasks

Creating a Java Class for a Web Service

Creating a One-Way Relationship Between Entity Beans

This section describes how to create a one-way relationship between entity beans. The relationship needs to match the relationship between tables in the underlying database.

To create a one-way relationship between entity beans in the Modeling Perspective:

1. Open the class diagram for the entity beans.
2. Select the **EJB Relationship (Unidirectional)** tool from the palette.
3. Select the source entity bean.
4. Select the target entity bean.

To create a one-way relationship between entity beans using the Code Editor:

1. Open the source code for the entity beans.
2. Add the new relationship directly to the source code.
3. Save your changes.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Creating a Relationship Between Entity Beans
Creating the Primary Key for an Entity Bean
Adding a Primary Key Join Field to an Entity Bean

Creating a Relationship Between Entity Beans

This section describes how to create a relationship between entity beans. The relationship needs to match the relationship between tables in the underlying database.

To create a relationship between entity beans in the Modeling Perspective:

1. Open the class diagram for the entity beans.
2. Select the **EJB Relationship** tool from the palette.
3. Select the source entity bean.
4. Select the target entity bean.

To create a relationship between entity beans using the Code Editor:

1. Open the source code for the entity bean.
2. Add the new relationship directly to the source code.
3. Save your changes.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Creating a One-Way Relationship Between Entity Beans
Creating a Relationship With Primary Key Mapping Between Entity Beans
Creating the Primary Key for an Entity Bean
Adding a Primary Key Join Field to an Entity Bean

Creating a Resource Reference

This section describes how to create a resource reference for an entity bean.

Note: For information on creating an injected resource reference, refer to the “Creating an Injected Resource Reference” link in the Related Information list at the bottom of this page.

To create a resource reference in the Modeling Perspective:

1. Open the class diagram for the entity bean.
2. Right click on the entity bean.
3. Select **New** ▸ **Resource Reference**.
4. Place the new resource reference in the diagram.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Environment and Resources Overview

Related Tasks

Creating an Environment Entry
Creating an Environment Resource Reference
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating a Run-As-Security Link

This section describes how to create a run-as-security link in an EJB project.

To create a run-as-security link in an EJB project in the Modeling Perspective:

1. Open the class diagram for the EJB project.
2. Verify the existence of the EJB and the security role that you want to connect.
3. Select the **Run-As-Security Link** tool from the palette.
4. Click on the EJB that needs a run-as security link.
5. Click on a security role to link the EJB to the security role.

To create a run-as-security link in an EJB project using the Code Editor:

1. Open the source code for project.
2. Add the run-as security link directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Security Roles Overview

Related Tasks

Creating a Security Role
Creating a Security Role Reference
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating a Security Role

This section describes how to create a security role in an EJB project.

To create a security role in an EJB project in the Modeling Perspective:

1. Open the class diagram for the EJB project.
2. Select the **Security Role** tool from the palette.
3. Click twice on the new security role.
4. Enter the name of the security role.

To create a security role in an EJB project using the Code Editor:

1. Open source code for the project.
2. Add the security role directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Security Roles Overview

Related Tasks

Creating a Run-As-Security Link
Creating a Security Role Reference
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating a Security Role Reference

This section describes how to create a security role reference in an EJB project.

To create a security role reference in an EJB project in the Modeling Perspective:

1. Open the class diagram for the EJB project.
2. Verify the existence of the source EJB and the target security role.
3. Select the **Security Role Reference** tool from the palette.
4. Click on the EJB that needs a security role reference.
5. Click on a security role to link the EJB to the security role.

To create a security role reference in an EJB project using the Code Editor:

1. Open source code for the project.
2. Add the security role reference directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Security Roles Overview

Related Tasks

Creating a Security Role
Creating a Run-As-Security Link
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating an EJB Reference

This section describes how to create an EJB reference.

Note: For information on how to create an injected EJB reference, refer to the “Creating an Injected EJB Reference” link in the Related Information section at the end of this page.

To create an EJB reference in the Modeling Perspective:

1. Open the class diagrams for the EJBs.
2. Select the **EJB Reference** tool from the palette.
3. Click on the source EJB.
4. Click on the target EJB.

Note: You can create an EJB reference from an EJB in one EJB package to an EJB in a different package.

To create an EJB reference using the Code Editor:

1. Open the source code for the EJBs.
2. Add the new reference and Xdoclet annotation directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Environment and Resources Overview

Related Tasks

Creating an Injected EJB Reference
Creating an Environment Entry
Creating an Environment Resource Reference
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating an Environment Entry

This section describes how to create an environment entry for an entity bean.

To create an environment entry in the Modeling Perspective:

1. Open the class diagram for the entity bean.
2. Right click on the entity bean.
3. Select **New** ▸ **Environment Entry**.
4. Place the new environment entry in the diagram.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Environment and Resources Overview

Related Tasks

Creating an EJB Reference
Creating an Environment Resource Reference
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating an Environment Resource Reference

This section describes how to create an environment resource reference for an entity bean.

To create an environment resource reference in the Modeling Perspective:

1. Open the class diagram for the entity bean.
2. Right click on the entity bean.
3. Select **New** ▸ **Environment Resource Reference**.
4. Place the new environment resource reference in the diagram.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Environment and Resources Overview

Related Tasks

Creating an EJB Reference
Creating an Environment Entry
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating the Primary Key for an Entity Bean

This section describes how to create the primary key for an entity bean. The primary key needs to match the primary key in the underlying database table.

To create a simple primary key field for an entity bean in the Modeling Perspective:

1. Open the class diagram for the entity bean.
2. Right click on the entity bean.
3. Select **New**.
4. Select **Simple PK Field**.

To create a compound primary key for an entity bean in the Modeling Perspective:

1. Click on the EJB in the Property Editor.
2. Click on **CMP Field**.
3. Click on the **Standard EJB Properties** tab in the Property Editor.
4. Select the fields used in the compound primary key.

To create the primary key field for an entity bean using the Code Editor:

1. Open the source code for the Entity bean.
2. Add the new field directly to the source code.
3. Save your changes.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Creating an Enterprise Java Bean (EJB) Modeling Project
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Adding a Primary Key Join Field to an Entity Bean
Creating a One-Way Relationship Between Entity Beans
Creating a Relationship Between Entity Beans

Deleting a Field from an Enterprise Java Bean (EJB)

This section describes how to delete a field from an Enterprise Java Bean (EJB).

To delete a field from an EJB in the Modeling Perspective:

1. Open the diagram for the EJB.
2. Right click on the field to be deleted.
3. Select delete.
4. Confirm the deletion of the field.

To delete a field from an EJB using the Code Editor:

1. Open source code for the EJB.
2. Delete the field directly from the source code.
3. Save your changes.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Adding a New Method to an EJB

Deleting a Method from an EJB

This section describes how to delete a method from an EJB.

To delete a method from an EJB in the Modeling Perspective:

1. Open the diagram for the EJB.
2. Right click on the method to be deleted.
3. Select delete.
4. Confirm the deletion of the method.

To delete a method from an EJB using the Code Editor:

1. Open source code for the EJB.
2. Delete the method from the source code.
3. Save your changes.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Adding a New Method to an EJB

Importing Entity Beans from a Database

This section describes how to import database tables into an EJB 2.x or EJB 3.0 project as entity beans.

To import entity beans from a database server in an EJB 2.x modeling project:

1. Right click on the EJB modeling project in the Model Navigator.
2. Select **Import Entity Beans from Database...**
3. Select the database connection from the drop-down list. If your database connection is missing from the list, click **Add connections . . .** to add the database connection to the list.
4. Select the database schema for importation.
5. Specify the source folder and package into which to import the entity beans.

   Note: You can specify a new package into which to import the data.

6. Select the tables to be imported.
7. Click **Finish** to import the entity beans.

To import entity beans from a database server in an EJB 3.0 modeling project:

1. Right click on the EJB modeling project in the Model Navigator.
2. Select **Import Entities from Database...**
3. Select the database connection from the drop-down list. If your database connection is missing from the list, click **Add connections. . .** to add the database connection to the list.
4. Select the database schema for importation.
5. Specify the source folder and package into which to import the entity beans.

   Note: You can specify a new package into which to import the data.

6. Select the tables to be imported.
7. Click **Finish** to import the entity beans.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Modifying an Enterprise Java Bean (EJB)

This section describes how to modify an enterprise java bean (EJB) using the Code Editor.

To modify an EJB:

1. Open the EJB's source code.
2. Make your changes directly to the bean's source code.
3. Save your changes.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Viewing the Source Code of an Enterprise Java Bean (EJB)
Deleting a Field from an Enterprise Java Bean (EJB)
Adding a New Method to an EJB
Deleting a Method from an EJB

Removing an Enterprise Java Bean (EJB)

This section describes how to remove an Enterprise Java Bean (EJB).

To remove an EJB using the Package Explorer:

1. Open the package containing the EJB.
2. Click on the Java file containing the bean.
3. Press the DELETE key on your keyboard.
4. Confirm the deletion.

To remove an EJB in the Modeling Perspective:

1. Open the diagram containing the ERJB.
2. Click on the EJB.
3. Press the Delete key on your keyboard.
4. Confirm the deletion.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating an Enterprise Java Bean (EJB) Modeling Project
Creating a Java Class for a Web Service
Modifying an Enterprise Java Bean (EJB)

Viewing the Source Code of an Enterprise Java Bean (EJB)

This section describes how to view the source code of an Enterprise Java Bean (EJB).

To view the source code of an EJB from the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Right click on the EJB.
4. Select Open to view the source code for the EJB.

To view the source code of an EJB from the Package Explorer:

1. Select the module in which the EJB resides.
2. Click on the package containing the EJB.
3. Double click on the EJB source file.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating an Enterprise Java Bean (EJB) Modeling Project
Modifying an Enterprise Java Bean (EJB)

Enterprise Java Beans (EJB) 3.0–Specific Tasks

This section provides information on specific tasks for working with Enterprise Java Bean (EJB) 3.0 applications in JBuilder or JGear development environments.

In This Section

Creating an EJB 3.0 Embeddable ID Class Reference
Describes how to create an EJB 3.0 embeddable ID class reference.

Creating an EJB 3.0 Entity Listener Reference
Describes how to create an EJB 3.0 entity listener reference.

Creating an EJB 3.0 Interceptor Reference
Describes how to create an EJB 3.0 interceptor reference.

Creating an EJB 3.0 Mapped Superclass
Describes how to create a new EJB 3.0 mapped superclass.

Creating an Injected EJB Reference
Describes how to create an injected EJB reference for a session bean.

Adding a New Named Native Query to an EJB 3.0 Entity Bean

This section describes how to add a named native query to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new named native query to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Named Native Query**.
5. Enter the name of the new named query.
6. Click on the named native query to view its properties.
7. Enter the query text in the **Query...Value** box.
8. Enter the result set in the **Resultset Mapping...Value** box.

To add a new named native query to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new named native query and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a New Named Query to an EJB 3.0 Entity Bean
Adding a Result Set Mapping to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a New Named Query to an EJB 3.0 Entity Bean

This section describes how to add a named query to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new named query to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Named Query**.
5. Enter the name of the new named query.
6. Click on the named query to view its properties.
7. Enter the query text in the **Query...Value** box.

To add a new named query to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new named query and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a New Named Native Query to an EJB 3.0 Entity Bean
Adding a Result Set Mapping to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a New Post-Load Method to an EJB 3.0 Entity Bean

This section describes how to add a new post-load method to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new post-load method to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Post-Load Method**.
5. Enter the name of the new method.
6. Click on the method to view and set its properties.

To add a new post-load method to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new method and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding a New Pre-Update Method to an EJB 3.0 Entity Bean
Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean
Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean
Adding a New Post-Update Method to an EJB 3.0 Entity Bean
Adding a New Post-Persist Method to an EJB 3.0 Entity Bean
Adding a New Post-Remove Method to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a New Post-Persist Method to an EJB 3.0 Entity Bean

This section describes how to add a new post-persist method to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new post-persist method to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Post-Persist Method**.
5. Enter the name of the new method.
6. Click on the method to view and set its properties.

To add a new post-persist method to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new method and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding a New Pre-Update Method to an EJB 3.0 Entity Bean
Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean
Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean
Adding a New Post-Update Method to an EJB 3.0 Entity Bean
Adding a New Post-Load Method to an EJB 3.0 Entity Bean
Adding a New Post-Remove Method to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a New Post-Remove Method to an EJB 3.0 Entity Bean

This section describes how to add a new post-remove method to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new post-remove method to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Post-Remove Method**.
5. Enter the name of the new method.
6. Click on the method to view and set its properties.

To add a new post-remove method to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new method and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding a New Pre-Update Method to an EJB 3.0 Entity Bean
Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean
Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean
Adding a New Post-Update Method to an EJB 3.0 Entity Bean
Adding a New Post-Load Method to an EJB 3.0 Entity Bean
Adding a New Post-Persist Method to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a New Post-Update Method to an EJB 3.0 Entity Bean

This section describes how to add a new post-update method to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new post-update method to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Post-Update Method**.
5. Enter the name of the new method.
6. Click on the method to view and set its properties.

To add a new post-update method to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new method and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding a New Pre-Update Method to an EJB 3.0 Entity Bean
Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean
Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean
Adding a New Post-Load Method to an EJB 3.0 Entity Bean
Adding a New Post-Persist Method to an EJB 3.0 Entity Bean
Adding a New Post-Remove Method to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean

This section describes how to add a new pre-persist method to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new pre-persist method to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Pre-Persist Method**.
5. Enter the name of the new method.
6. Click on the method to view and set its properties.

To add a new pre-persist method to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new method and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding a New Pre-Update Method to an EJB 3.0 Entity Bean
Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean
Adding a New Post-Update Method to an EJB 3.0 Entity Bean
Adding a New Post-Load Method to an EJB 3.0 Entity Bean
Adding a New Post-Persist Method to an EJB 3.0 Entity Bean
Adding a New Post-Remove Method to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean

This section describes how to add a new pre-remove method to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new pre-remove method to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Pre-Remove Method**.
5. Enter the name of the new method.
6. Click on the method to view and set its properties.

To add a new pre-remove method to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new method and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding a New Pre-Update Method to an EJB 3.0 Entity Bean
Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean
Adding a New Post-Update Method to an EJB 3.0 Entity Bean
Adding a New Post-Load Method to an EJB 3.0 Entity Bean
Adding a New Post-Persist Method to an EJB 3.0 Entity Bean
Adding a New Post-Remove Method to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a New Pre-Update Method to an EJB 3.0 Entity Bean

This section describes how to add a new pre-update method to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This method is only applicable to EJB 3.0 entity beans.

To add a new pre-update method to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Pre-Update Method**.
5. Enter the name of the new method.
6. Click on the method to view and set its properties.

To add a new pre-update method to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new method and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding a New Pre-Persist Method to an EJB 3.0 Entity Bean
Adding a New Pre-Remove Method to an EJB 3.0 Entity Bean
Adding a New Post-Update Method to an EJB 3.0 Entity Bean
Adding a New Post-Load Method to an EJB 3.0 Entity Bean
Adding a New Post-Persist Method to an EJB 3.0 Entity Bean
Adding a New Post-Remove Method to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a Post-Construct Method to an EJB 3.0 Session Bean

This section describes how to add a post-construct method to an EJB 3.0 session bean.

Note: This method is only available for EJB 3.0 session beans.

To add a post-construct method to an EJB 3.0 session bean in the Modeling Perspective:

1. Open the class diagram for the session bean.
2. Right click on the session bean.
3. Select **New** ▸ **Post-Construct Method**.
4. Click on the new method to view its properties.
5. Select properties for your new post-construct method.

To add a post-construct method to an EJB 3.0 session bean using the Code Editor:

1. Open the source code for the session bean.
2. Add the new post-construct method directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Session Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Creating a New Session Bean
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding an Interceptor Method to an EJB 3.0 Session Bean
Adding a Pre-Destroy Method to an EJB 3.0 Session Bean
Adding a Timeout Method to an EJB 3.0 Session Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a Pre-Destroy Method to an EJB 3.0 Session Bean

This section describes how to add a pre-destroy method to an EJB 3.0 session bean.

Note: This method type is only available for EJB 3.0 session beans.

To add a pre-destroy method to an EJB 3.0 session bean in the Modeling Perspective:

1. Open the class diagram for the session bean.
2. Right click on the session bean.
3. Select **New** ▸ **Pre-Destroy Method**.
4. Click on the new method to view its properties.
5. Select properties for your new pre-destroy method.

To add a pre-destroy method to an EJB 3.0 session bean using the Code Editor:

1. Open the source code for the session bean.
2. Add the new pre-destroy method directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Session Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Creating a New Session Bean
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding an Interceptor Method to an EJB 3.0 Session Bean
Adding a Post-Construct Method to an EJB 3.0 Session Bean
Adding a Timeout Method to an EJB 3.0 Session Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a Primary Key Join Field to an Entity Bean

This section describes how to add a primary key join column to an entity bean using either the Modeling Perspective or the Code Editor.

To add a primary key join column to an entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Primary Key Join Column**.
5. Enter the name of the new field.
6. Click in the new field to view its properties.
7. Enter the join definition in the **Definition...Value** box.
8. Enter the referenced column name in the **Referenced Column Name...Value** box.

To add a primary key join column to an entity bean to an EJB using the Code Editor:

1. Open the source code for the EJB.
2. Add the new field directly to the source code.
3. Add annotations to the source code.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Creating the Primary Key for an Entity Bean
Creating a One-Way Relationship Between Entity Beans
Creating a Relationship Between Entity Beans
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Field from an Enterprise Java Bean (EJB)

Adding a Result Set Mapping to an EJB 3.0 Entity Bean

This section describes how to add a result set mapping to an EJB 3.0 entity bean using either the Modeling Perspective or the Code Editor.

Note: This capability is only applicable to EJB 3.0 entity beans.

To add a new result set mapping to an EJB 3.0 entity bean in the Modeling Perspective:

1. Open the class diagram for the EJB.
2. Right click on the EJB.
3. Select **New**.
4. Select **Resultset Mapping**.
5. Enter the name of the new result set mapping.
6. Click on the result set mapping to view its properties.
7. Enter the column results in the **Column Results...Value** box.
8. Enter the entity results in the **Entity Results...Value** box.

To add a new result set mapping to an EJB 3.0 entity bean using the Code Editor:

1. Open the source code for the EJB.
2. Add the new result set mapping and annotations directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Adding a New Method to an EJB
Adding a New Named Query to an EJB 3.0 Entity Bean
Adding a New Named Native Query to an EJB 3.0 Entity Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding a Timeout Method to an EJB 3.0 Session Bean

This section describes how to add a timeout method to an EJB 3.0 session bean.

Note: This method type is only available for EJB 3.0 session beans.

To add a timeout method to an EJB 3.0 session bean in the Modeling Perspective:

1. Open the class diagram for the session bean.
2. Right click on the session bean.
3. Select **New** ▸ **Timeout Method**.
4. Click on the new method to view its properties.
5. Select properties for your new timeout method.

To add a timeout method to an EJB 3.0 session bean using the Code Editor:

1. Open the source code for the session bean.
2. Add the new timeout method directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Session Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Creating a New Session Bean
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding an Interceptor Method to an EJB 3.0 Session Bean
Adding a Post-Construct Method to an EJB 3.0 Session Bean
Adding a Pre-Destroy Method to an EJB 3.0 Session Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Adding an Interceptor Method to an EJB 3.0 Session Bean

This section describes how to add an interceptor method to an EJB 3.0 session bean.

Note: This method is only available for EJB 3.0 session beans.

To add an interceptor method to an EJB 3.0 session bean in the Modeling Perspective:

1. Open the class diagram for the session bean.
2. Right click on the session bean.
3. Select **New** ▸ **Interceptor Method**.
4. Click on the new method to view its properties.
5. Select properties for your new interceptor method.

To add an interceptor method to an EJB 3.0 session bean using the Code Editor:

1. Open the source code for the session bean.
2. Add the new interceptor method directly to the source code.
3. Add annotations.
4. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Session Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Creating a New Session Bean
Adding a New Method to an EJB
Adding a Business Method to an EJB
Adding a Post-Construct Method to an EJB 3.0 Session Bean
Adding a Pre-Destroy Method to an EJB 3.0 Session Bean
Adding a Timeout Method to an EJB 3.0 Session Bean
Creating an EJB 3.0 Interceptor Reference
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Deleting a Method from an EJB

Building a Package of Enterprise Java Beans (EJBs)

This section describes how to build a package of EJBs for later deployment to an application server.

To create a package in the Modeling Perspective:

1. Double click on the process node to open the default class diagram.
2. Choose the **Package** tool from the palette.
3. Place the package in the diagram.
4. Place your EJBs in the package.

Related Concepts

Enterprise Java Bean (EJB) Applications Overview

Related Tasks

Creating an Enterprise Java Bean (EJB) Modeling Project
Creating a Java Class for a Web Service

Creating a Relationship With Primary Key Mapping Between Entity Beans

This section describes how to create a relationship with primary key mapping between entity beans. To use primary key mapping, the source and target beans must have the same primary key field name. The relationship also needs to match the relationship between tables in the underlying database.

To create a relationship with primary key mapping between entity beans in the Modeling Perspective:

1. Open the class diagram for the entity beans.
2. Select the **EJB Relation With PK Mapping** tool from the palette.
3. Select the source entity bean.
4. Select the target entity bean.

To create a relationship with primary key mapping between entity beans in different packages in the Modeling Perspective:

1. Open the class diagram for the source entity bean.
2. Select the **EJB Relation With PK Mapping** tool from the palette.
3. Select the source entity bean.
4. Click on any whitespace in the diagram.
5. Select the target entity bean from the list.

To create a relationship with primary key mapping between entity beans using the Code Editor:

1. Open the source code for the entity beans.
2. Add the new relationship directly to the source code.
3. Save your changes.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)
Creating a Relationship Between Entity Beans
Creating a One-Way Relationship Between Entity Beans
Creating the Primary Key for an Entity Bean
Adding a Primary Key Join Field to an Entity Bean

Creating an EJB 3.0 Application Exception Class

This section describes how to create a new EJB 3.0 application exception class using either the Modeling Perspective or the Code Editor.

To create a new EJB 3.0 application exception class in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Select the **Application Exception** tool.
4. Place the application exception class in the model.

To create a new EJB 3.0 application exception class in the Code Editor:

1. Create a new Java file for your EJB 3.0 application exception class.
2. Code the EJB 3.0 application exception class by hand.
3. Add annotations.
4. Add the new EJB 3.0 application exception class source file to your project.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Creating an EJB 3.0 Mapped Superclass
Creating an EJB 3.0 Embeddable Class

Creating an EJB 3.0 Embeddable Class

This section describes how to create a new EJB 3.0 embeddable class using either the Modeling Perspective or the Code Editor.

To create a new EJB 3.0 embeddable class in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Select the **Embeddable Class** tool.
4. Place the embeddable class in the model.

To create a new EJB 3.0 embeddable class in the Code Editor:

1. Create a new Java file for your EJB 3.0 embeddable class.
2. Code the EJB 3.0 embeddable class by hand.
3. Add annotations.
4. Add the new EJB 3.0 embeddable class source file to your project.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Creating an EJB 3.0 Application Exception Class
Creating an EJB 3.0 Mapped Superclass
Creating an EJB 3.0 Embeddable ID Class Reference

Creating an EJB 3.0 Embeddable ID Class Reference

This section describes how to create an EJB 3.0 embeddable ID class reference.

Note: This feature is only available for EJB 3.0 projects.

To create an EJB 3.0 embeddable ID class reference in the Modeling Perspective:

1. Open the class diagrams for the EJBs.
2. Select the **Embeddable ID Class Reference** tool from the palette.
3. Click on the source EJB.
4. Click on the target embeddable class.

Note: You can create an EJB 3.0 embeddable ID class reference from an EJB in one EJB package to an embedded class in a different package.

To create an EJB 3.0 embeddable ID class reference to a class in a different package:

1. Open the class diagram for the source EJB.
2. Select the **Embeddable ID Class Reference** tool from the palette.
3. Click on the source EJB.
4. Click on any whitespace in the diagram.
5. Select the target embeddable class from the list.

To create an EJB 3.0 embeddable ID class reference using the Code Editor:

1. Open the source code for the EJB.
2. Add the new EJB 3.0 embeddable ID class reference and Java EE 5.0 annotation directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Environment and Resources Overview

Related Tasks

Creating an EJB Reference
Creating an EJB 3.0 Embeddable Class
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating an EJB 3.0 Entity Listener Reference

This section describes how to create an EJB 3.0 entity listener reference.

Note: This feature is only available for EJB 3.0 projects.

To create an EJB 3.0 entity listener reference in the Modeling Perspective:

1. Open the class diagram for the entity bean.
2. Select the **Entity Listener Reference** tool from the palette.
3. Click on the source entity bean.
4. Click on the target entity listener.

Note: You can create an EJB 3.0 interceptor reference from an EJB in one EJB package to an interceptor in a different package.

To create an EJB 3.0 entity listener reference from an EJB in one package to an interceptor in a different package:

1. Open the class diagram for the entity bean.
2. Select the **Entity Listener Reference** tool from the palette.
3. Click on the source entity bean.
4. Click any whitespace in the diagram.
5. Select the listener class in the dialog.

To create an EJB 3.0 entity listener reference using the Code Editor:

1. Open the source code for the entity bean.
2. Add the new EJB 3.0 entity listener reference and Java EE 5.0 annotation directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
Entity Bean Overview

Related Tasks

Creating a Java Class for a Web Service
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating an EJB 3.0 Interceptor Reference

This section describes how to create an EJB 3.0 interceptor reference.

Note: This feature is only available for EJB 3.0 projects.

To create an EJB 3.0 interceptor reference in the Modeling Perspective:

1. Open the class diagrams for the EJBs.
2. Select the **Interceptor Reference** tool from the palette.
3. Click on the source EJB method.
4. Click on the target interceptor.

Note: You can create an EJB 3.0 interceptor reference from an EJB in one EJB package to an interceptor in a different package.

To create an EJB 3.0 interceptor reference using the Code Editor:

1. Open the source code for the EJB.
2. Add the new EJB 3.0 interceptor reference and Java EE 5.0 annotation directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Adding an Interceptor Method to an EJB 3.0 Session Bean
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating an EJB 3.0 Mapped Superclass

This section describes how to create a new EJB 3.0 mapped superclass using either the Modeling Perspective or the Code Editor.

To create a new EJB 3.0 mapped superclass in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Select the **Mapped Superclass** tool.
4. Place the mapped superclass in the model.

To create a new EJB 3.0 mapped superclass in the Code Editor:

1. Create a new Java file for your EJB 3.0 mapped superclass.
2. Code the EJB 3.0 mapped superclass by hand.
3. Add annotations.
4. Add the new EJB 3.0 mapped superclass source file to your project.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java Class for a Web Service
Creating an EJB 3.0 Application Exception Class
Creating an EJB 3.0 Embeddable Class

Creating an Injected EJB Reference

This section describes how to create an injected EJB reference.

Note: This feature is only available for EJB 3.0 projects.

For information on how to create a non-injected EJB reference, refer to the “Creating an EJB Reference” link in the Related Information section at the end of this page.

To create an injected EJB reference in the Modeling Perspective:

1. Open the class diagrams for the EJBs.
2. Select the **Injected EJB Reference** tool from the palette.
3. Click on the source session bean.
4. Click on the target EJB.

Note: You can create an injected EJB reference from an EJB in one EJB package to an EJB in a different package.

To create an injected EJB reference using the Code Editor:

1. Open the source code for the EJBs.
2. Add the new injected reference and Java EE 5.0 annotation directly to the source code.
3. Save your changes.

Note: Adding artifacts in model diagrams generates source code and annotations. When you add artifacts manually, you are responsible for creating both source code and annotations.

Related Concepts

Enterprise Java Beans (EJB) Overview
EJB Environment and Resources Overview

Related Tasks

Creating an EJB Reference
Creating an Environment Entry
Creating an Environment Resource Reference
Viewing the Source Code of an Enterprise Java Bean (EJB)
Modifying an Enterprise Java Bean (EJB)

Creating a New Enterprise Java Bean (EJB)

This section describes how to create a new Enterprise Java Bean (EJB) in either the Modeling Perspective of in the Code Editor.

To create a new EJB in the Modeling Perspective:

1. Open the Modeling Perspective.
2. Bring up the model for your EJB project.
3. Select the appropriate bean tool.
4. Place the EJB in the model.
5. Connect the new EJB to the rest of your project.

To create a new EJB in the Code Editor:

1. Create a new Java file for your EJB.
2. Code the EJB by hand.
3. Add the new bean source file to your project.

Related Concepts

Enterprise Java Beans (EJB) Overview

Related Tasks

Creating an Enterprise Java Bean (EJB) Modeling Project
Creating a New Session Bean
Creating a Container-Managed-Persistence (CMP) Entity Bean
Creating a Container-Managed-Persistence (CMP) Entity Bean
Creating a Message Bean
Removing an Enterprise Java Bean (EJB)

Enabling XDoclet

Many Java EE applications require XDoclet support. This section describes how to enable XDoclet support.

XDoclet 1.2.3 with support for JDK 5.0 ships with JBuilder 2008 and is available in the JBuilder 2008 Eclipse plugins directory.

To enable XDoclet support:

1. Select **Window** ▶ **Preferences** ▶ **XDoclet**.
2. In the**Set XDoclet Runtime Preferences** dialog, check the **Enable XDoclet Builder** box to enable XDoclet support. Specify the home directory in **XDoclet Home** field. Select the appropriate version in the **Version** dropdown menu.
3. Click **Apply** and click **OK**.
4. You may also need to select **Window** ▶ **Preferences** ▶ **XDoclet** ▶ **ejbdoclet/webdoclet** options.
5. In the **ejbdoclet** or **webdoclet** dialogs, check the applicable tasks and servers.
6. Click **Restore Defaults** to restore default settings or**Apply** to apply the designated settings. Click **OK**.

Related Concepts

Java EE Applications Overview
Creating a Java EE Project

Related Tasks

Setting Up a Runtime Server

Web Applications

The Java EE platform provides a simple, unified standard for distributed applications through a component-based application model. Use the following links to learn how to create a Java web application with JBuilder 2008.

In This Section

Creating a Web Application Project
Describes steps to create a web application project in JBuilder 2007

Creating a Web Application Project

A web application includes dynamic web pages containing various types of markup language and generated by web components running in the web tier, and a web browser to render the pages received from the server. Use the following steps to get started creating a web application project in JBuilder 2008.

To create a new project:

1. Select **File** ▶ **File** ▶ **New** ▶ **Project** .
2. Type `Web` in the **Wizards** text entry box (to shift focus to the **Web** folder).
3. Select the **Web** folder and click **+** to view the sub-folders.
4. Choose to create a **Static** or **Dynamic** web project and click **Next**.
5. Type a **Project Name** in the text entry field, allow the default **Target Runtime** and **Configurations** options and click **Next**.
6. Allow the default **Project Facets** and click **Finish** to complete setup.

Tip: To configure detailed web module parameters accept the default **Project Facets** and click **Next**configure the following:

For a **Static** web application set the desired **Context Root** and **Web Content Folder** name then click **Finish**.

For a **Dynamic** web application set the desired **Context Root Content Directory** and **Java Source Directory** then click **Finish**.

Related Concepts

Web Applications Overview
Java EE Applications Overview

Related Tasks

Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server

Related Reference

Eclipse help topic “Server targeting for web applications”
Eclipse help topic “Web Projects'
Eclipse help topic “Creating a static web project”
Eclipse help topic “Dynamic web projects and applications”
Eclipse help topic “Web page design”

Web Services

The JBuilder or JGear web services features allow you to quickly design, deploy, run, and test a web service.

In This Section

Activating the Web Services Designer for Existing Components

The Web Services Designer creates a design surface for visually creating and implementing web services in an existing Java class or WSDL.

Tip: These steps assume correctly configured runtime and server parameters. Links to topics detailing these steps are listed in the Related Procedures section of this topic.

To activate the Web Services Explorer for existing components:

1. Open the desired dynamic web project containing the Java class or WSDL component.
2. If the file is a Java class, right click the component, select **Web Services** and click **Create Web Services from Model** in the drop down menu.

   If the file is a **WSDL**, right click the file in the **Package Explorer**, select **Web Services** and click **Select WSDL on Diagram** in the submenu.
3. To edit the element properties switch to the **Modeling** perspective:

   Select **Window** ▸ **Open Perspective** ▸ **Modeling**.

   Tip: Another way to display the **Properties** editor view for web services elements is to click on the element in the **Web Services Diagram** and select **Window** ▸ **Show View** ▸ **Properties**.
4. The **Properties** view is now open on the workbench.

Related Concepts

Web Services Overview
Runtime Servers

Related Tasks

Setting Up a Runtime Server
Opening the Web Services Designer
Working in the Web Services Designer
Setting Service Properties in the Web Services Designer
Setting WSDL Properties in the Web Services Designer

Configuring Your Workspace

To build a bottom-up web service and run it in JBuilder 2008, you first need to configure the Apache Tomcat server and JRE. The Eclipse Web Tools Project (WTP) uses Apache Axis 1.2 for the web service runtime.

To add JDK 5.0 as the JRE:

1. In Eclipse, open the **Installed JREs** page of the **Preferences** dialog box (**Windows** ▸ **Preferences** ▸ **Java** ▸ **Installed JREs**). You use this page to add Java runtime environments.
2. Click **Add** to display the **Add JRE** dialog box, where you add a JRE.
3. Leave the **JRE** type set to Standard VM.
4. In the **JRE Name** dialog box enter an identifying name for the JRE, such as JDK 5.0.
5. Choose the location of the JRE home folder in the **JRE Home Directory** field. Use the **Browse** button to browse to the location of a JDK 1.4.

   Note: This must be a full JDK, not just the JRE.

6. Enter any default VM arguments in the **Default VM Arguments** field.
7. Select the **Use Default System Libraries** option to use the default libraries.
8. Click **OK** when you are done.
9. Select the new JDK as the default in the **Installed JREs** list.

   This JRE will now be available in **New Server Runtime** dialog box where you configure Tomcat.

To setup Tomcat 5.5 as the server runtime:

1. Download Tomcat 5.5 from `http://tomcat.apache.org/download-55.cgi`.
2. Extract the compressed files to a local folder.
3. In Eclipse, open the **Installed Server Runtimes Environment** page of the **Preferences** dialog box (**Windows** ▸ **Preferences** ▸ **Server** ▸ **Installed Runtimes**). You use this page to configure server runtimes.
4. Click **Add** to display the **New Server Runtime** dialog box, where you add a server.
5. Open the **Apache** node and select **Apache Tomcat 5.5**. Click **Next**.
6. Click the **Browse** button next to the **Tomcat Installation Directory** field to browse to the Tomcat 5.5 local folder.
7. Choose a JDK from the **JRE** drop-down list.

   Note: Choose a 1.4 version of the JDK.

8. Click **Finish** when you are done.

   Apache Tomcat 5.5 is added to the **Installed Server Runtimes** list.

9. Select Tomcat 5.5 as the default and click **OK** to save the server runtime configuration.

   The selected runtime is used when you create new projects.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Creating a Client Project

To test your web service, you can create a web client or a Java utility client.

To create a web client project:

1. Open the **WebContent** node of your project.
2. Right-click the WSDL document that was created when you ran the web service.
3. Select **Web Services** ▸ **Create Client Project**.

   The **Create Client Project** wizard opens.
4. Verify the server. You can click the **Edit** button to change the selected server.
5. In the **Client Project Type** drop-down list, make sure that **Dynamic Web Project** is selected.
6. Change the default name of the client project in the **Client Project** field, if needed.
7. Click **Finish** to create the client project.

A new dynamic web project, that hosts the client project, is created. Generated files are placed in the `/Generated_Source/` folder of the client project. A `JUnit` test file is created. Do not change this test case directly. To update the test case, update the `JUnit` subclass that is written to the client project `/src/` folder. If you must change the test case and want to save your changes, you can set the WSDL **Test Case Overwrite** property to **false**.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Setting WSDL Properties in the Web Services Designer

Creating a Client Web Service from a URL WSDL

With a dynamic web project in place, you can create a client web service from a WSDL at a URL location.

To create a dynamic web client project from a WSDL URL location:

1. Right-click the dynamic web project node and choose **New ▸ Other ▸ Web Services ▸ Web Service Client From URL**.

   The **Add Web Service From URL** dialog box is opened.
2. Verify the server runtime. If is incorrect, click the **Edit** button to select the correct runtime.
3. Choose **Dynamic Web Project** from the **Client Project Type** drop-down list.
4. Enter the name of the client project in the **Client Project** field. The name defaults to `URLClient`.
5. Enter the WSDL location in the **WSDL Location** field. The path must point to a URL location. The filename must end in `.wsdl`.
6. Click **Finish** when you're done.

A new client project is created. Generated files are placed in the project's `Generated_Source` folder. A `JUnit` test file is created. Do not change this test case directly. To update the test case, update the `JUnit` subclass that is written to the client project `src` folder. If you must change the test case, you can set the WSDL **Test Case Overwrite** property to `false`.

Related Concepts

Web Services Overview

Related Tasks

Designing a Top-Down Web Service Using the Apache Axis Runtime

Creating a Dynamic Web Project

A web service is hosted in a dynamic web project.

To create a dynamic web project:

1 Create a new project (**File ▸ New ▸ Project**).

The **New Project** wizard is displayed.

2 Open the **Web** node in the **New Project** wizard, and choose **Dynamic Web Project**. Click **Next**.

The **New Dynamic Web Project** wizard is displayed.

3 Enter the project name in the **Project Name** field.

4 To place the project in the default workspace, select **Use Defaults**. To place the project in a different workspace, turn off **Use Defaults** and click the **Browse** button to browse to the workspace.

5 **Apache Tomcat 5.5**, the default server, is displayed in the **Target Runtime** drop down list. If it is not selected, select it from the list.

Note: Do not select **Add Project to EAR**.

6 Click **Finish** to create the project.

The new project is created in the **Dynamic Web Projects** node of the **Project Explorer.**

Related Concepts

Web Services Overview
Creating a New Web Service

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Creating a Java Class for a Web Service

To create a bottom-up web service, your dynamic web project needs to contain a Java class in the project `/src/` folder.

To create a Java class for a web service:

1. Open the **Project Explorer** (**Window ▸ Show View ▸ Project Explorer**) and open the **Dynamic Web Projects** node.
2. Right-click the project node and choose **New ▸ Other ▸ Class**.

   The **New Java Class** wizard is displayed.
3. Enter the name of the class in the **Name** field. You can leave all other fields at the default settings.

   Note: Eclipse does not recommend that you use the default package. Enter a package name in the **Package** field.

   The new class is opened in the source code editor.
4. Click **Finish** when you are done.

Add methods that can be exported to a web service. Save the class.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime

Creating a New Web Service

A single dynamic web project can contain multiple Java web services.

To add a Java web service to your project:

1. Open a dynamic web project, by selecting the project from the Project Explorer window at the left of the J2EE perspective (**Window** ▶ **Open Perspective** ▶ **Other** ▶ **J2EE**).
2. Open the Web Services Designer
3. Open the Web Services palette.
4. Click the Java web services icon.

   A Java web service representation is displayed on the design surface. Open the **Properties** view (**Window** ▶ **Show View** ▶ **Properties**) to set service properties. If the runtime and server are already configured, the service is immediately runnable.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Setting Service Properties in the Web Services Designer

Creating a New WSDL Web Service in the Web Services Designer

A single dynamic web project can contain multiple WSDL web services.

To add a WSDL web service to your project:

1. Open a dynamic web project.
2. Open the Web Services Designer.
3. Open the Web Services palette.
4. Click the WSDL web services icon.

   A WSDL web service representation is displayed on the design surface. Open the **Properties** view (**Window ▸ Show View ▸ Properties**) to set WSDL properties. If the runtime and server are already configured, the client is immediately runnable.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Setting WSDL Properties in the Web Services Designer

Creating a Web Service from a Java Project with a WSDL

You use the **Convert Into Web Services Client Project** wizard to convert a WSDL in a Java project into a client web service.

Note: The WSDL does not have to be contained in a dynamic web project.

To create a web service from a Java project containing a WSDL:

1. Right-click the `WSDL` file in the Java project and choose **Web Services** ▸ **Convert Into Client Project**.

   The **Convert Into Web Services Client Project** wizard is displayed.
2. Verify the server runtime. If is incorrect, click the **Edit** button to select the correct runtime.
3. In the **Client Project Type** drop-down list, choose the type of client project you want to create, either **Dynamic Web Project** or **Java Utility Project.**
4. Click **Finish** when you're done.

A new client project is created. Generated files are placed in the project's `Generated_Source` folder. A `JUnit` test file is created. Do not change this test case directly. To update the test case, update the `JUnit` subclass that is written to the client project `src` folder. If you must change the test case, you can set the WSDL **Test Case Overwrite** property to `false`.

If the client project is a Java project, generated files are also placed in the `/Generated_Source/` folder and a `JUnit` test file is also created. However, because there is no **WebContent** node in the project, the WSDL file is placed in the root of the`/src/` folder. The `META-INF` folder is also placed in the `/src/` folder.

Related Concepts

Web Services Overview

Related Tasks

Designing a Top-Down Web Service Using the Apache Axis Runtime

Designing a Bottom-Up Web Service Using the Apache Axis Runtime

A bottom-up web service is a web service that is designed from a Java class. This procedure outlines the steps for creating a bottom-up web service using Apache Axis and Tomcat.

- Apache Axis is an open source implementation of Simple Object Access Protocol (SOAP), an XML-based protocol for exchanging information.
- Apache Jakarta Tomcat provides a servlet container for your web service.

To design a web service from a Java class using Axis and Tomcat:

1. Configure your workspace.

   Configuring Your Workspace
2. Create a dynamic web project.

   Creating a Dynamic Web Project
3. Create a Java class for the web service.

   Creating a Java Class for a Web Service
4. Export the class to a web service.

   Exporting a Java Class to a Web Service
5. Set service properties.

   Setting Service Properties in the Web Services Designer
6. Run your web service.

   Running a Web Service
7. Create a client project to test your web service.

   Creating a Client Project
8. Set WSDL properties.

   Setting WSDL Properties in the Web Services Designer
9. Test your web service.

   Testing the Web Service with the Client

Related Concepts

Web Services Overview
Web Services Designer Overview

Related Tasks

Working in the Web Services Designer

Configuring Your Workspace

To build a bottom-up web service and run it in JBuilder 2008, you first need to configure the Apache Tomcat server and JRE. The Eclipse Web Tools Project (WTP) uses Apache Axis 1.2 for the web service runtime.

To add JDK 5.0 as the JRE:

1. In Eclipse, open the **Installed JREs** page of the **Preferences** dialog box (**Windows ▸ Preferences ▸ Java ▸ Installed JREs**). You use this page to add Java runtime environments.
2. Click **Add** to display the **Add JRE** dialog box, where you add a JRE.
3. Leave the **JRE** type set to Standard VM.
4. In the **JRE Name** dialog box enter an identifying name for the JRE, such as JDK 5.0.
5. Choose the location of the JRE home folder in the **JRE Home Directory** field. Use the **Browse** button to browse to the location of a JDK 1.4.

   Note: This must be a full JDK, not just the JRE.

6. Enter any default VM arguments in the **Default VM Arguments** field.
7. Select the **Use Default System Libraries** option to use the default libraries.
8. Click **OK** when you are done.
9. Select the new JDK as the default in the **Installed JREs** list.

   This JRE will now be available in **New Server Runtime** dialog box where you configure Tomcat.

To setup Tomcat 5.5 as the server runtime:

1. Download Tomcat 5.5 from `http://tomcat.apache.org/download-55.cgi`.
2. Extract the compressed files to a local folder.
3. In Eclipse, open the **Installed Server Runtimes Environment** page of the **Preferences** dialog box (**Windows ▸ Preferences ▸ Server ▸ Installed Runtimes**). You use this page to configure server runtimes.
4. Click **Add** to display the **New Server Runtime** dialog box, where you add a server.
5. Open the **Apache** node and select **Apache Tomcat 5.5**. Click **Next**.
6. Click the **Browse** button next to the **Tomcat Installation Directory** field to browse to the Tomcat 5.5 local folder.
7. Choose a JDK from the **JRE** drop-down list.

   Note: Choose a 1.4 version of the JDK.

8. Click **Finish** when you are done.

   Apache Tomcat 5.5 is added to the **Installed Server Runtimes** list.

9. Select Tomcat 5.5 as the default and click **OK** to save the server runtime configuration.

   The selected runtime is used when you create new projects.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Creating a Dynamic Web Project

A web service is hosted in a dynamic web project.

To create a dynamic web project:

1 Create a new project (**File** ▶ **New** ▶ **Project**).

The **New Project** wizard is displayed.

2 Open the **Web** node in the **New Project** wizard, and choose **Dynamic Web Project**. Click **Next**.

The **New Dynamic Web Project** wizard is displayed.

3 Enter the project name in the **Project Name** field.

4 To place the project in the default workspace, select **Use Defaults**. To place the project in a different workspace, turn off **Use Defaults** and click the **Browse** button to browse to the workspace.

5 **Apache Tomcat 5.5**, the default server, is displayed in the **Target Runtime** drop down list. If it is not selected, select it from the list.

Note: Do not select **Add Project to EAR**.

6 Click **Finish** to create the project.

The new project is created in the **Dynamic Web Projects** node of the **Project Explorer.**

Related Concepts

Web Services Overview
Creating a New Web Service

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Creating a Java Class for a Web Service

To create a bottom-up web service, your dynamic web project needs to contain a Java class in the project `/src/` folder.

To create a Java class for a web service:

1. Open the **Project Explorer** (**Window** ▶ **Show View** ▶ **Project Explorer**) and open the **Dynamic Web Projects** node.
2. Right-click the project node and choose **New** ▶ **Other** ▶ **Class**.

   The **New Java Class** wizard is displayed.
3. Enter the name of the class in the **Name** field. You can leave all other fields at the default settings.

   Note: Eclipse does not recommend that you use the default package. Enter a package name in the **Package** field.

   The new class is opened in the source code editor.
4. Click **Finish** when you are done.

Add methods that can be exported to a web service. Save the class.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime

Exporting a Java Class to a Web Service

Exporting a Java class to a web service opens the Web Services Designer and makes the service immediately runnable.

To export a Java class to a web service:

1. Expand the project **src** node so that you can see the class you just created.
2. Right-click the class.
3. Choose **Web Services** ▸ **Create Web Services Model**.

The Web Services Designer opens and creates a service representation. The methods in the class are exposed as a web service. You can set properties to modify the service or WSDL file.

The **Opening Diagram Progress** dialog box is displayed. The Web Services Designer is opened and a service representation is created. The methods in the class are exposed. A WSDL file is created in the **WebContent** node. You can set properties in the Web Services Designer to modify the service.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime

Setting Service Properties in the Web Services Designer

When you create a service in the Web Services Designer, a service representation is created. You can open the **Properties** view to set service properties that control the `Java2WSDL` builder. Default property values are created based on the selected server and toolkit.

The generated WSDL contains both interface and implementation WSDL constructs.

To set server properties:

1. If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.
2. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **Server properties**).

   You can set the following properties:

- **Binding name**: Fully-qualified name of client-side stub class that acts as a proxy for a remote web service.
- **Deploy scope:** Defines how instances of the service are created. **Request** selects one instance per request. **Application** shares one instance among all requests. **Session** selects one instance per authenticated session.
- **Extra classes:** Extra server classes.
- **Location URL:** URL of the service.
- **Port Type name:** Name to assign to the `portType` element in the generated WSDL file.
- **Namespace options:** Namespace options.
- **Service name:** A service interface that defines a `get` method for each port listed in the service element of the WSDL.
- **Service style:** The binding style in the WSDL document. **rpc** assigns Remote Procedure Call as the binding style. This is the default **document** assigns document as the binding style. Document services do not use encoding. **wrapped** assigns wrapped as the binding style. Wrapped services are a specialized form of document services, which unwrap document style data to individual parameters.
- **SOAP action:** Assigns a SOAP action for the operation in the WSDL. **DEFAULT** causes the soap action to be set according to the operation's meta data. **OPERATION** assigns the operation name as the SOAP action for the operation in the WSDL. **NONE** does not assign a SOAP action. This allows the action to be provided in the operation descriptor at runtime.
- **Type mapping version:** The type mapping version. Apache Axis 1.2 uses this setting internally to set up the default type mapping and the SOAP encoding type mappings. **1.1** chooses the default type mapping and no SOAP encoding. **1.2** chooses the default type mapping and SOAP encoding. **1.3** chooses the JAX-RPC 1.1 type mapping and SOAP encoding.
- **Use:** The use of the service and the WSDL document. **literal** specifies that the XML Schema define the representation of the XML for the request. **encoded** specifies that SOAP encoding be specified in the generated WSDL.

Changes are applied to the WSDL file at the next build.

To set web service properties:

1. If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.
2. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **Web service properties**).

   You can set the following properties:

- **Allowed methods:** Methods to expose in the service and the WSDL.
- **Class or interface:** Name of the class to be exported as a web service.
- **Disallowed methods:** Methods to exclude from the service and the WSDL.
- **Display name:** Name of service to be displayed.
- **Enabled:** Checked if this web service is enabled.
- **Exclude package/class from tree:** The classes to exclude from the search tree when exporting data types and methods for the web service.
- **Implementation class :** Name of interface implementation class.
- **Include inherited methods:** Check to include inherited methods.
- **Service port:** Port number of this service.

To set WSDL properties:

1. If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.
2. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **WSDL Properties**).

   You can set the following properties:

- **Implementation namespace:** Source namespace for the implementation WSDL.
- **Implementation WSDL file**: File name of the implementation WSDL.
- **Import schema:** Schema to be imported.
- **Include WSDL file:** WSDL file to be included.
- **Location import URL:** URL of the service.
- **Output:** Name of the input WSDL file. The output WSDL file contains all data from the input WSDL file plus any new constructs.
- **Target Namespace:** Target namepace for the implementation WSDL.

Related Concepts

Web Services Overview
Apache Axis Toolkit

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Working in the Web Services Designer

Related Reference

Java2 WSDL Reference

Running a Web Service

When you open your web service in the Web Services Designer, the service is runnable.

To run a web service:

1. Choose **Run ▸ Run**.

   The **Run** dialog box is displayed.
2. Expand the **Web Service** node in the **Configurations** list. Choose the name of your project.

   On the **Run** page, the web module is selected and the **Launch URI** field is set to the name of the runnable Axis servlet.
3. Click **Run.**

   The **Run On Server** dialog box displayed, where you select a server instance to run the web service on.
4. In the **Select Server Type** list, make sure Tomcat 4.1 Server is selected.
5. Select the **Set Server As Project Default** option so you will not be asked again to select a server for this project.
6. Click **Finish**.

The **Servers** view is opened. The **Console** view is also opened and displays Tomcat startup messages. The **Web browser** opens and shows the available services, including the service exposed in your project. You can select the WSDL link to view the WSDL document generated by Axis.

Note: You use the WSDL document to generate the client project.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Creating a Client Project

To test your web service, you can create a web client or a Java utility client.

To create a web client project:

1. Open the **WebContent** node of your project.
2. Right-click the WSDL document that was created when you ran the web service.
3. Select **Web Services** ▸ **Create Client Project**.

   The **Create Client Project** wizard opens.
4. Verify the server. You can click the **Edit** button to change the selected server.
5. In the **Client Project Type** drop-down list, make sure that **Dynamic Web Project** is selected.
6. Change the default name of the client project in the **Client Project** field, if needed.
7. Click **Finish** to create the client project.

A new dynamic web project, that hosts the client project, is created. Generated files are placed in the `/Generated_Source/` folder of the client project. A `JUnit` test file is created. Do not change this test case directly. To update the test case, update the `JUnit` subclass that is written to the client project `/src/` folder. If you must change the test case and want to save your changes, you can set the WSDL **Test Case Overwrite** property to **false**.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Setting WSDL Properties in the Web Services Designer

Setting WSDL Properties in the Web Services Designer

When you create a service in the Web Services Explorer, a service representation is created. Open the **Properties** view to set properties for the `WSDL2Java` builder. Default property values are created based on the selected server and toolkit.

To set WSDL properties:

1 If the WSDL representation is not displayed in the Web Services Designer, right-click the WSDL you want to create a client project from and choose **Web Services** ▸ **Create Client Project**.

2 Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties**).

You can set the following properties:

- **All:** Set to **true** to generate code for all elements, even un-referenced ones. By default, `WSDL2Java` only generates code for those elements in the WSDL file that are referenced.
- **Debug:** Set to **true** to print debug information (the `WSDL2Java` symbol table).
- **Deploy Scope:** Defines how instances of the service are created. Select **Request** to select one instance per request. Select **Application** to share one instance among all requests. Select **Session** to select one instance per authenticated session.
- **HelperGen:** Set to **true** to generate all type mapping in separate helper classes.
- **No Imports:** Set to **true** to ignore the `import` statements in the WSDL and the schema associated with the WSDL. Uses the immediate WSDL document
- **Output:** The root directory for all generated files.
- **OverwriteTypes:** Set to **true** to overwrite existing bean types of the same name with new Java source.
- **Package For All:** Set to **true** to write all generated files to same package (set with the **Package Name** property).
- **Package Name:** The package name for generated files.
- **Server Side:** Set to **true** to generate the server-side bindings for the web service.
- **Skeleton Deploy:** Set to **true** to generate the optional skeleton class to encapsulate an implementation for the server.
- **Test Case:** Set to **true** to generate a `JUnit` test case the first time you build the project. Any changes you make to the test case will never be overwritten when building, unless you set the **Test Case Overwrite** property.
- **Test Case Overwrite:** Set to **true** to overwrite the existing `JUnit` test case each time you build the project.
- **Timeout:** Timeout in seconds. The default is **0**. Set to **-1** to disable.
- **Typemapping Version:** The type mapping version. Apache Axis 1.2 uses this setting internally to set up the default type mapping and the SOAP encoding type mappings. Choose **1.1** to choose the default type mapping and no SOAP encoding. Choose **1.2** to choose the default type mapping and SOAP encoding. Choose **1.3** to choose the JAX-RPC 1.1 type mapping and SOAP encoding.
- **URL:** The location of the input WSDL file.
- **Verbose:** Set to **true** to display output from builder.
- **Wrapped:** Set to **true** to unwrap data to individual parameters. The WSDL must have **wrapped** specified as the Style property for this option to work.

Changes are applied to the WSDL file at the next build.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime
Working in the Web Services Designer

Related Reference

WSDL2 Java Reference

Testing the Web Service with the Client

When you deploy your web service to the web services server, the Axis Admin console is displayed, where you validate your web service.

To run the client project:

1 Choose Run ▸ Run.

The **Run** dialog box is displayed.

2 Expand the **Web Client** node in the **Configurations** list and choose the client project.

On the **Run** page, the web module is selected and the **Launch URI** field is set to launch the test JSP.

3 Click **Run.**

The **Run On Server** dialog box displayed, where you select the server instance for your client project.

4 In the **Select Server Type** list, make sure Tomcat 4.1 Server is selected.

5 Select the **Set Server As Project Default** option so you will not be asked again to select a server for this project.

6 Click **Finish**.

The **Servers** view is opened. The **Console** view is also opened and displays Tomcat startup messages. The **Web browser** opens and shows the web client test project. Test a method by choosing it from the list on the left and clicking the **Invoke** button.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Designing a Top-Down Web Service Using the Apache Axis Runtime

A top-down web service is a web service that is designed from a WSDL document. This procedure outlines the steps for creating a top-down service using Apache Axis and Tomcat.

- Apache Axis is an open source implementation of Simple Object Access Protocol (SOAP), an XML-based protocol for exchanging information.
- Apache Jakarta Tomcat provides a servlet container for your web service.

To design a web service from a WSDL document using Axis:

1. Configure your workspace.

   Configuring Your Workspace

2. Create a dynamic web project.

   Creating a Dynamic Web Project

3. Create a client web service from a WSDL identified by its URL address.

   Note: You can also create a client web service from a Java project containing a WSDL. (The Java project does not have to be a dynamic web project.)

   Creating a Client Web Service from a URL WSDL

4. Set WSDL properties.

   Setting WSDL Properties in the Web Services Designer

5. Run your web service.

   Running a Web Service

6. Test your web service.

   Testing the Web Service with the Client

Related Concepts

Web Services Overview
Web Services Designer Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Creating a Web Service from a Java Project with a WSDL

Configuring Your Workspace

To build a bottom-up web service and run it in JBuilder 2008, you first need to configure the Apache Tomcat server and JRE. The Eclipse Web Tools Project (WTP) uses Apache Axis 1.2 for the web service runtime.

To add JDK 5.0 as the JRE:

1 In Eclipse, open the **Installed JREs** page of the **Preferences** dialog box (**Windows ▸ Preferences ▸ Java ▸ Installed JREs**). You use this page to add Java runtime environments.

2 Click **Add** to display the **Add JRE** dialog box, where you add a JRE.

3 Leave the **JRE** type set to Standard VM.

4 In the **JRE Name** dialog box enter an identifying name for the JRE, such as JDK 5.0.

5 Choose the location of the JRE home folder in the **JRE Home Directory** field. Use the **Browse** button to browse to the location of a JDK 1.4.

Note: This must be a full JDK, not just the JRE.

6 Enter any default VM arguments in the **Default VM Arguments** field.

7 Select the **Use Default System Libraries** option to use the default libraries.

8 Click **OK** when you are done.

9 Select the new JDK as the default in the **Installed JREs** list.

This JRE will now be available in **New Server Runtime** dialog box where you configure Tomcat.

To setup Tomcat 5.5 as the server runtime:

1 Download Tomcat 5.5 from `http://tomcat.apache.org/download-55.cgi`.

2 Extract the compressed files to a local folder.

3 In Eclipse, open the **Installed Server Runtimes Environment** page of the **Preferences** dialog box (**Windows ▸ Preferences ▸ Server ▸ Installed Runtimes**). You use this page to configure server runtimes.

4 Click **Add** to display the **New Server Runtime** dialog box, where you add a server.

5 Open the **Apache** node and select **Apache Tomcat 5.5**. Click **Next**.

6 Click the **Browse** button next to the **Tomcat Installation Directory** field to browse to the Tomcat 5.5 local folder.

7 Choose a JDK from the **JRE** drop-down list.

Note: Choose a 1.4 version of the JDK.

8 Click **Finish** when you are done.

Apache Tomcat 5.5 is added to the **Installed Server Runtimes** list.

9 Select Tomcat 5.5 as the default and click **OK** to save the server runtime configuration.

The selected runtime is used when you create new projects.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Creating a Dynamic Web Project

A web service is hosted in a dynamic web project.

To create a dynamic web project:

1 Create a new project (**File ▸ New ▸ Project**).

The **New Project** wizard is displayed.

2 Open the **Web** node in the **New Project** wizard, and choose **Dynamic Web Project**. Click **Next**.

The **New Dynamic Web Project** wizard is displayed.

3 Enter the project name in the **Project Name** field.

4 To place the project in the default workspace, select **Use Defaults**. To place the project in a different workspace, turn off **Use Defaults** and click the **Browse** button to browse to the workspace.

5 **Apache Tomcat 5.5**, the default server, is displayed in the **Target Runtime** drop down list. If it is not selected, select it from the list.

Note: Do not select **Add Project to EAR**.

6 Click **Finish** to create the project.

The new project is created in the **Dynamic Web Projects** node of the **Project Explorer.**

Related Concepts

Web Services Overview
Creating a New Web Service

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Creating a Client Web Service from a URL WSDL

With a dynamic web project in place, you can create a client web service from a WSDL at a URL location.

To create a dynamic web client project from a WSDL URL location:

1. Right-click the dynamic web project node and choose **New** ▶ **Other** ▶ **Web Services** ▶ **Web Service Client From URL**.

   The **Add Web Service From URL** dialog box is opened.
2. Verify the server runtime. If is incorrect, click the **Edit** button to select the correct runtime.
3. Choose **Dynamic Web Project** from the **Client Project Type** drop-down list.
4. Enter the name of the client project in the **Client Project** field. The name defaults to `URLClient`.
5. Enter the WSDL location in the **WSDL Location** field. The path must point to a URL location. The filename must end in `.wsdl`.
6. Click **Finish** when you're done.

A new client project is created. Generated files are placed in the project's `Generated_Source` folder. A `JUnit` test file is created. Do not change this test case directly. To update the test case, update the `JUnit` subclass that is written to the client project `src` folder. If you must change the test case, you can set the WSDL **Test Case Overwrite** property to `false`.

Related Concepts

Web Services Overview

Related Tasks

Designing a Top-Down Web Service Using the Apache Axis Runtime

Setting WSDL Properties in the Web Services Designer

When you create a service in the Web Services Explorer, a service representation is created. Open the **Properties** view to set properties for the `WSDL2Java` builder. Default property values are created based on the selected server and toolkit.

To set WSDL properties:

1. If the WSDL representation is not displayed in the Web Services Designer, right-click the WSDL you want to create a client project from and choose **Web Services** ▸ **Create Client Project**.
2. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties**).

   You can set the following properties:

- **All:** Set to **true** to generate code for all elements, even un-referenced ones. By default, `WSDL2Java` only generates code for those elements in the WSDL file that are referenced.
- **Debug:** Set to **true** to print debug information (the `WSDL2Java` symbol table).
- **Deploy Scope:** Defines how instances of the service are created. Select **Request** to select one instance per request. Select **Application** to share one instance among all requests. Select **Session** to select one instance per authenticated session.
- **HelperGen:** Set to **true** to generate all type mapping in separate helper classes.
- **No Imports:** Set to **true** to ignore the `import` statements in the WSDL and the schema associated with the WSDL. Uses the immediate WSDL document
- **Output:** The root directory for all generated files.
- **OverwriteTypes:** Set to **true** to overwrite existing bean types of the same name with new Java source.
- **Package For All:** Set to **true** to write all generated files to same package (set with the **Package Name** property).
- **Package Name:** The package name for generated files.
- **Server Side:** Set to **true** to generate the server-side bindings for the web service.
- **Skeleton Deploy:** Set to **true** to generate the optional skeleton class to encapsulate an implementation for the server.
- **Test Case:** Set to **true** to generate a `JUnit` test case the first time you build the project. Any changes you make to the test case will never be overwritten when building, unless you set the **Test Case Overwrite** property.
- **Test Case Overwrite:** Set to **true** to overwrite the existing `JUnit` test case each time you build the project.
- **Timeout:** Timeout in seconds. The default is **0**. Set to **-1** to disable.
- **Typemapping Version:** The type mapping version. Apache Axis 1.2 uses this setting internally to set up the default type mapping and the SOAP encoding type mappings. Choose **1.1** to choose the default type mapping and no SOAP encoding. Choose **1.2** to choose the default type mapping and SOAP encoding. Choose **1.3** to choose the JAX-RPC 1.1 type mapping and SOAP encoding.
- **URL:** The location of the input WSDL file.
- **Verbose:** Set to **true** to display output from builder.
- **Wrapped:** Set to **true** to unwrap data to individual parameters. The WSDL must have **wrapped** specified as the Style property for this option to work.

Changes are applied to the WSDL file at the next build.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime
Working in the Web Services Designer

Related Reference

WSDL2 Java Reference

Running a Web Service

When you open your web service in the Web Services Designer, the service is runnable.

To run a web service:

1. Choose **Run** ▸ **Run**.

   The **Run** dialog box is displayed.
2. Expand the **Web Service** node in the **Configurations** list. Choose the name of your project.

   On the **Run** page, the web module is selected and the **Launch URI** field is set to the name of the runnable Axis servlet.
3. Click **Run.**

   The **Run On Server** dialog box displayed, where you select a server instance to run the web service on.
4. In the **Select Server Type** list, make sure Tomcat 4.1 Server is selected.
5. Select the **Set Server As Project Default** option so you will not be asked again to select a server for this project.
6. Click **Finish**.

The **Servers** view is opened. The **Console** view is also opened and displays Tomcat startup messages. The **Web browser** opens and shows the available services, including the service exposed in your project. You can select the WSDL link to view the WSDL document generated by Axis.

Note: You use the WSDL document to generate the client project.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Testing the Web Service with the Client

When you deploy your web service to the web services server, the Axis Admin console is displayed, where you validate your web service.

To run the client project:

1. Choose **Run** ▸ **Run**.

   The **Run** dialog box is displayed.
2. Expand the **Web Client** node in the **Configurations** list and choose the client project.

   On the **Run** page, the web module is selected and the **Launch URI** field is set to launch the test JSP.
3. Click **Run.**

   The **Run On Server** dialog box displayed, where you select the server instance for your client project.
4. In the **Select Server Type** list, make sure Tomcat 4.1 Server is selected.
5. Select the **Set Server As Project Default** option so you will not be asked again to select a server for this project.
6. Click **Finish**.

The **Servers** view is opened. The **Console** view is also opened and displays Tomcat startup messages. The **Web browser** opens and shows the web client test project. Test a method by choosing it from the list on the left and clicking the **Invoke** button.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Exporting a Java Class to a Web Service

Exporting a Java class to a web service opens the Web Services Designer and makes the service immediately runnable.

To export a Java class to a web service:

1. Expand the project **src** node so that you can see the class you just created.
2. Right-click the class.
3. Choose **Web Services** ▸ **Create Web Services Model**.

The Web Services Designer opens and creates a service representation. The methods in the class are exposed as a web service. You can set properties to modify the service or WSDL file.

The **Opening Diagram Progress** dialog box is displayed. The Web Services Designer is opened and a service representation is created. The methods in the class are exposed. A WSDL file is created in the **WebContent** node. You can set properties in the Web Services Designer to modify the service.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime

Opening the Web Services Designer

To open the Web Services Designer for a Java web service:

1. Open the dynamic web project containing the Java class you want to export to a web service.
2. Right-click the class and choose **Web Services ▸ Create Web Service Model**.

   The Web Services Designer is opened. Open the **Properties** view (**Window ▸ Show View ▸ Properties**) to set service properties. If the runtime and server are already configured, the service is immediately runnable.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Setting Service Properties in the Web Services Designer

Running a Web Service

When you open your web service in the Web Services Designer, the service is runnable.

To run a web service:

1. Choose Run ▸ Run.

   The **Run** dialog box is displayed.
2. Expand the **Web Service** node in the **Configurations** list. Choose the name of your project.

   On the **Run** page, the web module is selected and the **Launch URI** field is set to the name of the runnable Axis servlet.
3. Click **Run.**

   The **Run On Server** dialog box displayed, where you select a server instance to run the web service on.
4. In the **Select Server Type** list, make sure Tomcat 4.1 Server is selected.
5. Select the **Set Server As Project Default** option so you will not be asked again to select a server for this project.
6. Click **Finish**.

The **Servers** view is opened. The **Console** view is also opened and displays Tomcat startup messages. The **Web browser** opens and shows the available services, including the service exposed in your project. You can select the WSDL link to view the WSDL document generated by Axis.

Note: You use the WSDL document to generate the client project.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Setting Service Properties in the Web Services Designer

When you create a service in the Web Services Designer, a service representation is created. You can open the **Properties** view to set service properties that control the `Java2WSDL` builder. Default property values are created based on the selected server and toolkit.

The generated WSDL contains both interface and implementation WSDL constructs.

To set server properties:

1. If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.
2. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **Server properties**).

   You can set the following properties:

- **Binding name**: Fully-qualified name of client-side stub class that acts as a proxy for a remote web service.
- **Deploy scope:** Defines how instances of the service are created. **Request** selects one instance per request. **Application** shares one instance among all requests. **Session** selects one instance per authenticated session.
- **Extra classes:** Extra server classes.
- **Location URL:** URL of the service.
- **Port Type name:** Name to assign to the `portType` element in the generated WSDL file.
- **Namespace options:** Namespace options.
- **Service name:** A service interface that defines a `get` method for each port listed in the service element of the WSDL.
- **Service style:** The binding style in the WSDL document. **rpc** assigns Remote Procedure Call as the binding style. This is the default **document** assigns document as the binding style. Document services do not use encoding. **wrapped** assigns wrapped as the binding style. Wrapped services are a specialized form of document services, which unwrap document style data to individual parameters.
- **SOAP action:** Assigns a SOAP action for the operation in the WSDL. **DEFAULT** causes the soap action to be set according to the operation's meta data. **OPERATION** assigns the operation name as the SOAP action for the operation in the WSDL. **NONE** does not assign a SOAP action. This allows the action to be provided in the operation descriptor at runtime.
- **Type mapping version:** The type mapping version. Apache Axis 1.2 uses this setting internally to set up the default type mapping and the SOAP encoding type mappings. **1.1** chooses the default type mapping and no SOAP encoding. **1.2** chooses the default type mapping and SOAP encoding. **1.3** chooses the JAX-RPC 1.1 type mapping and SOAP encoding.
- **Use:** The use of the service and the WSDL document. **literal** specifies that the XML Schema define the representation of the XML for the request. **encoded** specifies that SOAP encoding be specified in the generated WSDL.

Changes are applied to the WSDL file at the next build.

To set web service properties:

1. If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.
2. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **Web service properties**).

   You can set the following properties:

- **Allowed methods:** Methods to expose in the service and the WSDL.
- **Class or interface:** Name of the class to be exported as a web service.
- **Disallowed methods:** Methods to exclude from the service and the WSDL.
- **Display name:** Name of service to be displayed.
- **Enabled:** Checked if this web service is enabled.
- **Exclude package/class from tree:** The classes to exclude from the search tree when exporting data types and methods for the web service.
- **Implementation class :** Name of interface implementation class.
- **Include inherited methods:** Check to include inherited methods.
- **Service port:** Port number of this service.

To set WSDL properties:

1. If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.
2. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **WSDL Properties**).

   You can set the following properties:

- **Implementation namespace:** Source namespace for the implementation WSDL.
- **Implementation WSDL file**: File name of the implementation WSDL.
- **Import schema:** Schema to be imported.
- **Include WSDL file:** WSDL file to be included.
- **Location import URL:** URL of the service.
- **Output:** Name of the input WSDL file. The output WSDL file contains all data from the input WSDL file plus any new constructs.
- **Target Namespace:** Target namepace for the implementation WSDL.

Related Concepts

Web Services Overview
Apache Axis Toolkit

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Working in the Web Services Designer

Related Reference

Java2 WSDL Reference

Setting WSDL Properties in the Web Services Designer

When you create a service in the Web Services Explorer, a service representation is created. Open the **Properties** view to set properties for the `WSDL2Java` builder. Default property values are created based on the selected server and toolkit.

To set WSDL properties:

1 If the WSDL representation is not displayed in the Web Services Designer, right-click the WSDL you want to create a client project from and choose **Web Services** ▸ **Create Client Project**.

2 Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties**).

You can set the following properties:

- **All:** Set to **true** to generate code for all elements, even un-referenced ones. By default, `WSDL2Java` only generates code for those elements in the WSDL file that are referenced.
- **Debug:** Set to **true** to print debug information (the `WSDL2Java` symbol table).
- **Deploy Scope:** Defines how instances of the service are created. Select **Request** to select one instance per request. Select **Application** to share one instance among all requests. Select **Session** to select one instance per authenticated session.
- **HelperGen:** Set to **true** to generate all type mapping in separate helper classes.
- **No Imports:** Set to **true** to ignore the `import` statements in the WSDL and the schema associated with the WSDL. Uses the immediate WSDL document
- **Output:** The root directory for all generated files.
- **OverwriteTypes:** Set to **true** to overwrite existing bean types of the same name with new Java source.
- **Package For All:** Set to **true** to write all generated files to same package (set with the **Package Name** property).
- **Package Name:** The package name for generated files.
- **Server Side:** Set to **true** to generate the server-side bindings for the web service.
- **Skeleton Deploy:** Set to **true** to generate the optional skeleton class to encapsulate an implementation for the server.
- **Test Case:** Set to **true** to generate a `JUnit` test case the first time you build the project. Any changes you make to the test case will never be overwritten when building, unless you set the **Test Case Overwrite** property.
- **Test Case Overwrite:** Set to **true** to overwrite the existing `JUnit` test case each time you build the project.
- **Timeout:** Timeout in seconds. The default is **0**. Set to **-1** to disable.
- **Typemapping Version:** The type mapping version. Apache Axis 1.2 uses this setting internally to set up the default type mapping and the SOAP encoding type mappings. Choose **1.1** to choose the default type mapping and no SOAP encoding. Choose **1.2** to choose the default type mapping and SOAP encoding. Choose **1.3** to choose the JAX-RPC 1.1 type mapping and SOAP encoding.
- **URL:** The location of the input WSDL file.
- **Verbose:** Set to **true** to display output from builder.
- **Wrapped:** Set to **true** to unwrap data to individual parameters. The WSDL must have **wrapped** specified as the Style property for this option to work.

Changes are applied to the WSDL file at the next build.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime
Working in the Web Services Designer

Related Reference

WSDL2 Java Reference

Testing the Web Service with the Client

When you deploy your web service to the web services server, the Axis Admin console is displayed, where you validate your web service.

To run the client project:

1. Choose **Run** ▸ **Run**.

   The **Run** dialog box is displayed.
2. Expand the **Web Client** node in the **Configurations** list and choose the client project.

   On the **Run** page, the web module is selected and the **Launch URI** field is set to launch the test JSP.
3. Click **Run.**

   The **Run On Server** dialog box displayed, where you select the server instance for your client project.
4. In the **Select Server Type** list, make sure Tomcat 4.1 Server is selected.
5. Select the **Set Server As Project Default** option so you will not be asked again to select a server for this project.
6. Click **Finish**.

The **Servers** view is opened. The **Console** view is also opened and displays Tomcat startup messages. The **Web browser** opens and shows the web client test project. Test a method by choosing it from the list on the left and clicking the **Invoke** button.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Working in the Web Services Designer

To work in the Web Services Designer:

1. Open the Web Services Designer.

   Opening the Web Services Designer
2. Activate the Web Services Designer for existing components.

   Activating the Web Services Designer for Existing Components
3. Create a new web service.

   Creating a New Web Service
4. Create a new web service.

   Creating a New WSDL Web Service in the Web Services Designer
5. Set service properties.

   Setting Service Properties in the Web Services Designer
6. Set WSDL options.

   Setting WSDL Properties in the Web Services Designer

Related Concepts

Web Services Overview
Web Services Designer Overview

Opening the Web Services Designer

To open the Web Services Designer for a Java web service:

1 Open the dynamic web project containing the Java class you want to export to a web service.

2 Right-click the class and choose **Web Services** ▸ **Create Web Service Model**.

The Web Services Designer is opened. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties**) to set service properties. If the runtime and server are already configured, the service is immediately runnable.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Setting Service Properties in the Web Services Designer

Activating the Web Services Designer for Existing Components

The Web Services Designer creates a design surface for visually creating and implementing web services in an existing Java class or WSDL.

Tip: These steps assume correctly configured runtime and server parameters. Links to topics detailing these steps are listed in the Related Procedures section of this topic.

To activate the Web Services Explorer for existing components:

1. Open the desired dynamic web project containing the Java class or WSDL component.
2. If the file is a Java class, right click the component, select **Web Services** and click **Create Web Services from Model** in the drop down menu.

   If the file is a **WSDL**, right click the file in the **Package Explorer**, select **Web Services** and click **Select WSDL on Diagram** in the submenu.
3. To edit the element properties switch to the **Modeling** perspective:

   Select **Window** ▶ **Open Perspective** ▶ **Modeling**.

   Tip: Another way to display the **Properties** editor view for web services elements is to click on the element in the **Web Services Diagram** and select **Window** ▶ **Show View** ▶ **Properties**.
4. The **Properties** view is now open on the workbench.

Related Concepts

Web Services Overview
Runtime Servers

Related Tasks

Setting Up a Runtime Server
Opening the Web Services Designer
Working in the Web Services Designer
Setting Service Properties in the Web Services Designer
Setting WSDL Properties in the Web Services Designer

Creating a New Web Service

A single dynamic web project can contain multiple Java web services.

To add a Java web service to your project:

1. Open a dynamic web project, by selecting the project from the Project Explorer window at the left of the J2EE perspective (**Window** ▸ **Open Perspective** ▸ **Other** ▸ **J2EE**).
2. Open the Web Services Designer
3. Open the Web Services palette.
4. Click the Java web services icon.

   A Java web service representation is displayed on the design surface. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties**) to set service properties. If the runtime and server are already configured, the service is immediately runnable.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Setting Service Properties in the Web Services Designer

Creating a New WSDL Web Service in the Web Services Designer

A single dynamic web project can contain multiple WSDL web services.

To add a WSDL web service to your project:

1. Open a dynamic web project.
2. Open the Web Services Designer.
3. Open the Web Services palette.
4. Click the WSDL web services icon.

   A WSDL web service representation is displayed on the design surface. Open the **Properties** view (**Window ▸ Show View ▸ Properties**) to set WSDL properties. If the runtime and server are already configured, the client is immediately runnable.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Setting WSDL Properties in the Web Services Designer

Setting Service Properties in the Web Services Designer

When you create a service in the Web Services Designer, a service representation is created. You can open the **Properties** view to set service properties that control the `Java2WSDL` builder. Default property values are created based on the selected server and toolkit.

The generated WSDL contains both interface and implementation WSDL constructs.

To set server properties:

1 If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.

2 Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **Server properties**).

You can set the following properties:

- **Binding name**: Fully-qualified name of client-side stub class that acts as a proxy for a remote web service.
- **Deploy scope:** Defines how instances of the service are created. **Request** selects one instance per request. **Application** shares one instance among all requests. **Session** selects one instance per authenticated session.
- **Extra classes:** Extra server classes.
- **Location URL:** URL of the service.
- **Port Type name:** Name to assign to the `portType` element in the generated WSDL file.
- **Namespace options:** Namespace options.
- **Service name:** A service interface that defines a `get` method for each port listed in the service element of the WSDL.
- **Service style:** The binding style in the WSDL document. **rpc** assigns Remote Procedure Call as the binding style. This is the default **document** assigns document as the binding style. Document services do not use encoding. **wrapped** assigns wrapped as the binding style. Wrapped services are a specialized form of document services, which unwrap document style data to individual parameters.
- **SOAP action:** Assigns a SOAP action for the operation in the WSDL. **DEFAULT** causes the soap action to be set according to the operation's meta data. **OPERATION** assigns the operation name as the SOAP action for the operation in the WSDL. **NONE** does not assign a SOAP action. This allows the action to be provided in the operation descriptor at runtime.
- **Type mapping version:** The type mapping version. Apache Axis 1.2 uses this setting internally to set up the default type mapping and the SOAP encoding type mappings. **1.1** chooses the default type mapping and no SOAP encoding. **1.2** chooses the default type mapping and SOAP encoding. **1.3** chooses the JAX-RPC 1.1 type mapping and SOAP encoding.
- **Use:** The use of the service and the WSDL document. **literal** specifies that the XML Schema define the representation of the XML for the request. **encoded** specifies that SOAP encoding be specified in the generated WSDL.

Changes are applied to the WSDL file at the next build.

To set web service properties:

1 If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.

2 Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **Web service properties**).

You can set the following properties:

- **Allowed methods:** Methods to expose in the service and the WSDL.
- **Class or interface:** Name of the class to be exported as a web service.
- **Disallowed methods:** Methods to exclude from the service and the WSDL.
- **Display name:** Name of service to be displayed.
- **Enabled:** Checked if this web service is enabled.
- **Exclude package/class from tree:** The classes to exclude from the search tree when exporting data types and methods for the web service.
- **Implementation class :** Name of interface implementation class.
- **Include inherited methods:** Check to include inherited methods.
- **Service port:** Port number of this service.

To set WSDL properties:

1. If the class representation is not displayed in the Web Services Designer, right-click the Java class you want to export to a web service and choose **Web Services** ▸ **Create Web Services Model**.
2. Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties** ▸ **WSDL Properties**).

   You can set the following properties:

- **Implementation namespace:** Source namespace for the implementation WSDL.
- **Implementation WSDL file**: File name of the implementation WSDL.
- **Import schema:** Schema to be imported.
- **Include WSDL file:** WSDL file to be included.
- **Location import URL:** URL of the service.
- **Output:** Name of the input WSDL file. The output WSDL file contains all data from the input WSDL file plus any new constructs.
- **Target Namespace:** Target namepace for the implementation WSDL.

Related Concepts

Web Services Overview
Apache Axis Toolkit

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Working in the Web Services Designer

Related Reference

Java2 WSDL Reference

Setting WSDL Properties in the Web Services Designer

When you create a service in the Web Services Explorer, a service representation is created. Open the **Properties** view to set properties for the `WSDL2Java` builder. Default property values are created based on the selected server and toolkit.

To set WSDL properties:

1 If the WSDL representation is not displayed in the Web Services Designer, right-click the WSDL you want to create a client project from and choose **Web Services** ▸ **Create Client Project**.

2 Open the **Properties** view (**Window** ▸ **Show View** ▸ **Properties**).

You can set the following properties:

- **All:** Set to **true** to generate code for all elements, even un-referenced ones. By default, `WSDL2Java` only generates code for those elements in the WSDL file that are referenced.
- **Debug:** Set to **true** to print debug information (the `WSDL2Java` symbol table).
- **Deploy Scope:** Defines how instances of the service are created. Select **Request** to select one instance per request. Select **Application** to share one instance among all requests. Select **Session** to select one instance per authenticated session.
- **HelperGen:** Set to **true** to generate all type mapping in separate helper classes.
- **No Imports:** Set to **true** to ignore the `import` statements in the WSDL and the schema associated with the WSDL. Uses the immediate WSDL document
- **Output:** The root directory for all generated files.
- **OverwriteTypes:** Set to **true** to overwrite existing bean types of the same name with new Java source.
- **Package For All:** Set to **true** to write all generated files to same package (set with the **Package Name** property).
- **Package Name:** The package name for generated files.
- **Server Side:** Set to **true** to generate the server-side bindings for the web service.
- **Skeleton Deploy:** Set to **true** to generate the optional skeleton class to encapsulate an implementation for the server.
- **Test Case:** Set to **true** to generate a `JUnit` test case the first time you build the project. Any changes you make to the test case will never be overwritten when building, unless you set the **Test Case Overwrite** property.
- **Test Case Overwrite:** Set to **true** to overwrite the existing `JUnit` test case each time you build the project.
- **Timeout:** Timeout in seconds. The default is **0**. Set to **-1** to disable.
- **Typemapping Version:** The type mapping version. Apache Axis 1.2 uses this setting internally to set up the default type mapping and the SOAP encoding type mappings. Choose **1.1** to choose the default type mapping and no SOAP encoding. Choose **1.2** to choose the default type mapping and SOAP encoding. Choose **1.3** to choose the JAX-RPC 1.1 type mapping and SOAP encoding.
- **URL:** The location of the input WSDL file.
- **Verbose:** Set to **true** to display output from builder.
- **Wrapped:** Set to **true** to unwrap data to individual parameters. The WSDL must have **wrapped** specified as the Style property for this option to work.

Changes are applied to the WSDL file at the next build.

Related Concepts

Web Services Overview

Related Tasks

Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime
Working in the Web Services Designer

Related Reference

WSDL2 Java Reference

Working with Runtime Servers

A server runtime environment is used to test, debug and run a project. It provides the environment, libraries and infrastructure that a “server” needs. A server is an instance of the server runtime used to host web applications and other server-side components. Your JBuilder or JGears product comes bundled with several runtimes and supports various others. Refer to the Runtime Servers concept topic for more details.

In This Section

Publishing a Java EE Application to a Server Runtime

This section describes how to publish a Java application to a runtime server. The server runtime has to be configured in the **Servers** view with a Java EE project from the current workspace added to the runtime server for deployment. The publish action redeploys the selected projects for an application server.

To publish an application to a server runtime:

1. Select **Servers** **Windows** ▸ **Show View** ▸ **Other** ▸ **Server** ▸ **Servers** to open the **Servers** view window.
2. Installed server runtimes appear in the **Servers** view window. Right-click on the runtime server name and click **Add and Remove Projects** to deploy or undeploy any Java EE projects in your workspace.
3. Right-click on the runtime server name and click **Publish**, or click the **Publish to the server** icon at the top of the **Servers** view window. The Publish action redeploys available projects for the selected server.

Related Concepts

Java EE Applications Overview
Runtime Servers

Related Tasks

Creating a Java EE Project
Setting Up a Runtime Server
Running an Application on a Runtime Server
Developing Enterprise Java Bean (EJB) Applications

Related Reference

Eclipse help topic “Web application overview”
Eclipse help topic “ Server targeting for Web applications”
Eclipse help topic “Running a Java program”
Eclipse help topic “Debugging a servlet on a server”

Running an Application on a Runtime Server

JBuilder 2008 supports various Java EE runtime servers. This topic describes how to run an Java application on a runtime server using the JBoss application server technology.

To run an application on the JBoss server:

1. Open the project in JBuilder 2008.
2. In the **Navigation** view select the project folder.
3. Right click the highlighted project to reveal the drop down menu options and select **Run as**.
4. Click **Run on Server** and click **JBoss** in the **Select the server type** list and click **Next**.
5. Confirm the default configuration settings in the **New JBoss Server** window and click **Next**.
6. In the **Add and Remove Projects** window, confirm the project is listed in the **Configured projects** section and click **Finish**.

Related Concepts

Runtime Servers
Java EE Applications Overview

Related Tasks

Creating a Java EE Project
Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Developing Enterprise Java Bean (EJB) Applications

Related Reference

Eclipse help topic "Web application overview"
Eclipse help topic " Server targeting for Web applications"
Eclipse help topic "Running a Java program"
Eclipse help topic "Debugging a servlet on a server"

Setting Up a Runtime Server

Java EE 5.0 applications work with a runtime application server. There are several types of Java EE applications, including Enterprise Java Bean (EJB) 3.0 applications, web applications, and web services. This section describes how to set up a runtime server for any of these application types.JBuilder 2008 supports various Java EE runtime servers.

You create a server using the runtime environment best suited to the project by defining a pointer from the workbench to an existing installation of an application server. Use the following steps to set up a runtime server in JBuilder 2008.

Note: This topic describes setting up a runtime server. The tasks describe creating a runtime server using the **JBoss** application server technology. Most other runtime servers can be set up in a similar fashion.

To set up a runtime server:

1. From the **Workbench** select **Window** ▸ **Preferences** .
2. Select **Servers** ▸ **Installed Runtimes**.

   Tip: A best practice when using the runtime servers is to choose one of the versions that has (CodeGear or Borland) after it. These versions have been extended to support specific features.

   Note: The remaining steps describe creating an application server using the JBoss runtime environment.

3. Click **Add**, choose the appropriate JBoss (Borland) runtime environment, and click **Next**.
4. Accept the default **JRE** and click **Browse** to choose the **Application Server Directory**.
5. Select the root directory of JBuilder 2008 and choose the **JBuilder 2007** folder.
6. Select the **thirdparty** folder, select the folder representing the desired **JBoss** runtime environment, and click **Finish**.

   The desired **JBoss** runtime now appears in the **Installed Server Runtime Environments** list.

To associate the desired runtime with the project folder:

1. Right click on the project folder in the **Navigation View**.
2. Select **Properties**.
3. Select **Targeted Runtimes**.
4. Activate the checkbox next to the desired server, click **Apply** and **OK**.

   The desired runtime is now associated with the project.

Related Concepts

Runtime Servers
Java EE Applications Overview
Web Applications Overview
Web Services Overview
Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java EE Project
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server

Related Reference

Borland Application Server Documentation
Eclipse help topic “Server targeting for Web applications”
Eclipse help topic “Web application overview”

Setting Up and Using a Borland Application Server

Java EE 5.0 applications work with a runtime application server. This section describes how to set up a Borland Application Server (BAS) runtime

You create a server using the runtime environment best suited to the project by defining a pointer from the workbench to an existing installation of an application server. Use the following steps to set up a BAS runtime server in JBuilder 2008.

To set up a BAS runtime server in JBuilder 2007:

1 From the **Workbench** select **Window** ▶ **Preferences**.

2 Select **Servers** ▶ **Installed Runtimes**.

3 Click **Add**, choose the appropriate BAS runtime environment (with OpenJMS or Tibco). Click **Next**.

4 Accept the default **JRE** and click **Browse** to choose the root of the BAS directory.

5 Select the root directory of JBuilder 2008 and choose the **JBuilder 2007** folder.

6 Click **Finish**.

The desired **BAS** runtime now appears in the **Installed Server Runtime Environments** list.

7 To configure the server for deployment select**Window** ▶ **Show View** ▶ **Other** ▶ **Server** ▶ **Servers**

Note: The default server setup for deployment is the **j2eeSample** configuration with the partition, **WelcomePartition**, which is the default managed partition in the sample configuration. You can only start managed partitions in JBuilder 2008; therefore, you must only setup managed partitions for startup and deployment from within the IDE.

To debug with the BAS Runtime in JBuilder 2007:

1 To prepare to debug a partition in JBuilder 2008, you must complete the following steps to configure a partition for remote debugging. Start the Borland Management Console.

2 Start the BAS server.

3 Locate the partition you want to debug under the Management Hub.

4 Right-click on the partition name and choose **Properties**. Select the **Partition Process Settings** tab.

5 Check the **Enable JPDA Remote Debugging** option. Set the transport address field to the desired port number. Uncheck the **Suspend Partition Until Debugger Attaches** option. Click **OK.**

6 Shut down the BAS server from the console. Launch JBuilder 2008. Configure the server runtime for deployment as described in the previous task.

7 Start the server in the **Servers** view. With the server selected, click **Run** ▶ **Debug**. In the **Debug** window, click on **Remote Java Application** in the left-side list. The icon meanings appear on the right-side. Click on the **New** icon at the top of the left-side list. The right-side pane now has a dialog to attach a Java virtual machine that accepts debug connections. Name the configuration in the **Name** field. Set the host name in the **Host** field to `localhost`. Set the port number to the partition's remote debug port number.

8 Click on **Debug** to start the debug session.

9 After the debugger launches successfully and stops at the breakpoint, add the project to the **Default Source Lookup** for the debugger if you encounter **Source not found** errors in the Debug perspective.

To create and run an EJB client:

1 Create a new Java class with a main method. In the main method, modify the Java Naming and Directory Interface (JNDI) code to lookup the EJB. Lookup codes does not need any server-specific properties for BAS.

2 Select **Run ▸ Run**.

3 In the **Run** window, click on **Java Application** in the left-side list. The icon meanings appear on the right-side. Click on the **New** icon at the top of the left-side list. The right-side pane now has a dialog to specify a new configuration. Name the configuration in the **Name** field. Set the main class to the EJB client class in the **Main class** field.

4 Click the**Arguments** tab. Set the **VM arguments** field to:

```
-Dvbroker.agent.port=port_no- Djava.endorsed.dirs=/BorlandAppServer/lib/endorsed
```

where `port_no` is the osagent port for the application server.

5 Start the BAS server and deploy the EJB application.

6 Go to **Window ▸ Preferences**. Type `User Libraries` in the **type filter text** area. Click on **User Libraries**.

7 Click on **New** in the **User Libraries** pane on the right-side of the screen. Enter `EJB Stubs` in the **User library name** field. Click **OK**.

8 Select the new library from the list. Click on **Add JARs**. Add the deployed EJB JAR from:

```
/AppServer/var/domains/configurations/configuration_name/mos/partition_name
```

where `configuration_name` and `partition_name` have been replaced with your server configuration data. Close the **Preferences** dialog window.

9 In the **Navigator** view, right-click on the project and select **Properties**. Select **Java Build Path** and **Libraries.**. Click **Add** and add the `EJB Stubs` library to the project. Click on **Add** again and add the Client Library for BAS 6.7 to the project. Run the client configuration.

To stop the management agent after stopping the BAS server:

1 When the server is stopped in the IDE, only the configuration and partition are stopped which improves wait times during restarts. The management agent cannot be stopped from within the IDE.

2 To stop the management agent, launch the BAS console from **/BorlandAppServer/bin**.

3 Expand the Management Hubs node, right-click on the hub and select **Shutdown**.

Related Concepts

Runtime Servers
Java EE Applications Overview
Web Applications Overview
Web Services Overview
Enterprise Java Beans (EJB) Overview

Related Tasks

Creating a Java EE Project
Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server

Related Reference

Borland Application Server Documentation

Creating a Java EE Project

The Java EE perspective includes the following workbench views:

- Java Servlet and JavaServer Pages (JSP)
- Application clients and applets components that run on the client
- Technology web components that run on the server
- Enterprise JavaBeans (EJB)

To create a Java EE project

1. From the main window, click **File New Project**.
2. Click the **+** next to the **J2EE** folder to reveal all options.
3. Depending on the project requirements, choose from one of the following project types:
   - **Application Client Project**
   - **Connector Project**
   - **Enterprise Application Project**
   - **Utility Project**
4. Click **Next**.
5. Type a project name in the **Project Name** text field.
6. Select the desired configuration parameters for:
   - **Project Contents**
   - **Target Runtime**
   - **Configurations**
   - **EAR Membership**
7. Click **Next**.
8. Configure the desired **Project Facet** parameters and click **Next**.
9. Configure the desired **Source Folder** or accept the default value and click **Finish**.

Related Concepts

Java EE Applications Overview
Creating a Java EE Project
Web Services Overview
Enterprise Java Bean (EJB) Applications Overview
Modeling Applications Overview
Runtime Servers

Related Tasks

Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server

Related Reference

Eclipse help topic (J2EE) “Reference”
Eclipse help topic “J2EE Applications”
Eclipse help topic “Working with projects”
Eclipse help topic “Project Explorer view in the J2EE perspective”
Eclipse help topic “J2EE architecture”
Eclipse help topic “J2EE perspective”

Developing Java EE Applications

Use the following steps to develop a new Java EE project with JBuilder 2008.

To create a new project:

1. Set the workbench perspective to Java:

   Window ▸ Perspective ▸ Java

2. Select **File ▸ New ▸ Project**.
3. Select the **J2EE Node** and click **Next** .
4. Set the **Project Name**, **Target Runtime**, and **Configurations** preferences and click **Finish**.

Related Concepts

Java EE Applications Overview
Runtime Servers

Related Tasks

Creating a Java EE Project
Setting Up a Runtime Server
Publishing a Java EE Application to a Server Runtime
Running an Application on a Runtime Server
Developing Enterprise Java Bean (EJB) Applications

Related Reference

"Eclipse Help Topic "Changing the Java compiler version for a J2EE project
"Eclipse Help Topic "J2EE Applications"

Importing a Java EE Project

To import a Java EE project into the IDE:

1. Open JBuilder 2008.
2. Select **File** ▸ **New** ▸ **Project** to invoke the **New Project Wizard**.
3. Browse to the **EJB** folder and select **EJB Project**.
4. Configure the following project preferences:

- Project Name
- Target Runtime
- Configurations
- Project Contents

5. For **Project Contents**, deactivate the **Use Default** checkbox and browse to the desired directory to select the Java EE project to be imported.

   Click **Next** to invoke the **Project Facets** dialog.
6. Activate the checkbox next to **Java Version 5.0** and click **Finish**.

   The Java EE project is now open in the **Navigation View**.

Related Concepts

Java EE Applications Overview

Related Tasks

Developing Java EE Applications
Setting Up a Runtime Server
Running an Application on a Runtime Server

JPA Applications

Java Persistence API (JPA) was included in your JBuilder or JGear product to simplify the development of Java EE and Java SE applications using data persistence.

In This Section

Creating a Dynamic Web Java Persistence API (JPA) Modeling Project
Descripes the steps to create a Dynamic Web JPA modeling project.

Creating a Java Persistence API (JPA) Modeling Project
Describes the steps to create a JPA modeling project.

Creating a Dynamic Web Java Persistence API (JPA) Modeling Project

To create a Dynamic Web JPA modeling project:

1 Select **File** ▸ **New** ▸ **Project** to invoke the **New Project** wizard.

2 In the **Select a Wizard** window, navigate to the **JPA** folder, select **Dynamic Web JPA Modeling Project** and click **Next**.

Tip: Type `JPA` in the wizard text box to quickly navigate to the **JPA** folder.

3 Name the project and select the **Hibernate** or **Toplink** **Persistence Manager**.

Activate the **Add library to the class path** checkbox, accept the default settings for the remaining parameters, and click **Next**.

4 Configure the following **Persistence Unit** settings:

- Persistence Unit Name
- Transaction Type
- Database Type
- Database Connection
- Schema

Tip: If an active connection is not already configured, click the **Add Connection** link to complete the task.

Click **Finish**.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Runtime Servers

Related Tasks

Setting Up a Runtime Server

Related Reference

Hibernate Documentation

Creating a Java Persistence API (JPA) Modeling Project

To create a Java Persistence API (JPA) modeling project:

1. Select **File** ▸ **New** ▸ **Project** to invoke the **New Project** wizard.
2. In the **Select a Wizard** window navigate to the **JPA** folder, select **JPA Modeling Project** and click **Next**.

   Tip: Type `JPA` in the wizard text box to quickly navigate to the **JPA** folder.

3. Name the project and select the **Hibernate** or **Toplink** **Persistence Manager**.

   Activate the **Add library to the class path** checkbox, accept the default settings for the remaining parameters, and click **Next**.
4. Configure the following **Persistence Unit** settings:

- Persistence Unit Name
- Transaction Type
- Database Type
- Database Connection
- Schema

   Tip: If an active connection is not already configured, click the **Add Connection** link to complete the task.

Click **Finish**.

Related Concepts

Modeling Applications Overview
Java EE Applications Overview
Runtime Servers

Related Tasks

Setting Up a Runtime Server

Related Reference

New JPA Modeling Project: Persistence unit settings page
New JPA Modeling Project: Java Settings
Hibernate Documentation
TopLink Resources

Setting Up Database Connections

This section provides links to information about creating database applications with JBuilder 2008.

In This Section

Connecting to an InterBase Database
Describes how to create an InterBase connection from your CodeGear product.

Connecting to Blackfish SQL for Java
Describes how to create a connection to Blackfish SQL for Java from your CodeGear product.

Connecting to an InterBase Database

This topic describes how to create an InterBase connection.

To connect to an InterBase database using the DTP tools in JBuilder:

1. Open the **Data Source Explorer** view by selecting **Window** ▸ **Show View** ▸ **Other** and filter on `Data Source Explorer` in the TYPE FILTER TEXT
2. Right-click on the Databases node in the **Data Source Explorer**. Select **New**.
3. Click on **Generic JDBC Connection**.
4. Enter the database name.
5. Click **Next**. Click the **(...)** button next to the driver drop-down list.
6. Expand the InterBase node and click on **2007**.
7. Click on **Add** to add a JDBC driver template.
8. Expand the version node and click on the **JDBC Driver** definition. Accept the default name for the driver and click **OK**.
9. The default JDBC jar location points to the InterClient JDBC driver delivered as a plugin with JBuilder. Do not make any changes to the JDBC driver location.
10. Set JDBC connection properties to the values specified in the table below. Change properties as per the database to which you are connecting.
    - **Connection URL**: jdbc:interbase://localhost/C:\Borland\InterBase?\examples\database\employee.gdb
    - **User Name**: SYSDBA
    - **Password**: masterkey
11. Click **OK** to close the driver configuration dialog.
12. Click **OK** to close the driver definition dialog.
13. Click on the **Test Connection** button in the **JDBC Connection Profile** dialog to ensure that the connection is configured correctly.
14. Click on **Save Password** to store the password.
15. Click on **Finish** to close the connection dialog.
16. Right-click on the connection under the Databases node in the **Data Source Explorer** view and select **Connect** to connect to the database.
17. Expand the database connection to explore tables and columns. Use the right-click context menu options to view data, modify schema and content.

For detailed information about how to use InterBase, please see the InterBase documentation included with this product.

Related Reference

InterBase Documentation

Connecting to Blackfish SQL for Java

This topic describes how to create a connection to Blackfish SQL for Java. Blackfish SQL for Java remote connections are supported in your CodeGear product.

To connect to a BlackFish SQL database using the DTP tools in JBuilder:

1. Open the **Data Source Explorer** view by selecting **Window** ▶ **Show View** ▶ **Other** and filter on `Data Source Explorer` in the TYPE FILTER TEXT area.
2. Right-click on the Databases node in the **Data Source Explorer**. Select **New**.
3. Click on **Generic JDBC Connection**.
4. Enter the database name.
5. Click **Next**. Click the **(...)** button next to the driver drop-down list.
6. Select the **Generic JDBC Driver** node from the list.
7. Click on **Add** to add a JDBC driver template.
8. Expand the version node and click on the **JDBC Driver** definition, at the lowest level. Accept the default name for the driver and click **OK**.
9. Click the **Add Jar/Zip** button to add the Blackfish SQL JDBC driver to the list. The default location for the driver is JDataStore/lib/jdsserver.jar. .
10. Set JDBC connection properties to the values specified in the table below. Change properties as per the database to which you are connecting.
    - **Connection URL**:jdbc:borland:dsremote://localhost/c:\JDataStore7?\samples\JDataStore?\datastores \employee.jds
    - **User Name**: SYSDBA
    - **Password**: masterkey
11. Click **OK** to close the driver configuration dialog.
12. Click **OK** to close the driver definition dialog.
13. Click on the **Test Connection** button in the **JDBC Connection Profile** dialog to ensure that the connection is configured correctly.
14. Click on **Save Password** to store the password.
15. Click on **Finish** to close the connection dialog.
16. Right-click on the connection under the Databases node in the **Data Source Explorer** view and select **Connect** to connect to the database.
17. Expand the database connection to explore tables and columns. Use the right-click context menu options to view data, modify schema and content.

For detailed information about how to use Blackfish SQL for Java, please see the Blackfish SQL (or JDataStore) documentation included with this product.

Related Reference

Blackfish SQL Documentation

Using Application Factory

This section contains links to the Application Factory procedural topics.

In This Section

Creating Data-Aware Web Applications with Application Factory

This section describes using Application Factory to create data-aware web applications built on such technologies as JSF, Spring MVC, and Struts 2.

In This Section

Adding a User/Login Module

The user/login module for all data-aware application modules is installed by default. The application module configuration wizard provides the option of not including the module when importing the data-aware application into the workspace. This topic describes how to add a user/login module to your application with Application Factory, if you did not select the option of including the module when importing the data-aware application into the workspace.

To add user/login module to a data-aware web application:

1. Switch to the **Application Factory Modeling** perspective.
2. Right-click on the web application project in the workspace and select the menu option **Data-Aware Application Tools ▸ Add User and Login Module**.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Changing Company Name

This topic describes how to change the company name in a data-aware web application project.

To change the company name in a data-aware web application:

1 Open the **Application Factory Modeling** perspective by either of the following methods:

- **Window** ▸ **Open Perspective** ▸ **Other** ▸ **Application Factory Modeling**
- Use the toolbar icon**Open Perspective**. Select **Other** ▸ **Application Factory Modeling**

Note: If you have previously had the **Application Factory Modeling** perspective open, you can switch to it by clicking the **Application FactoryModeling perspective** icon on the toolbar.

2 Click on the **Scripts - Application Factory** view.

3 Double-click on the script name **Change Company Name.js** to launch the script.

4 Select the web application project from the dropdown menu.

5 Enter the changed company name in the **text** field. Click **OK** .

6 Click the **Show Changes** button to display the changes that are to be made by the script. This option brings up the changes listed in the **Script Learn/Resolve/Commit** view in the IDE. It provides the opportunity to examine each change and accept or discard changes at the file level. The **Script Learn/Resolve/Commit** view also provides user interface to resolve missing resources that are required by the script, as well as change the location in which code snippets are inserted into existing files.

7 Click on the **Save** button in the toolbar to commit all changes. This action writes out all changes to files.

8 Click the **Commit Now** button to commit all changes to files without browsing through and resolving changes.

9 Check the option **Do not show this dialog again** to always commit without browsing through changes made by a script run. Script commit behavior can be set in Application Factorypreferences under **Window** ▸ **Preferences,** ▸ **Application Factory** .

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Changing the CSS Theme

This topic describes how to change the Cascading Style Sheet (CSS) theme for a data-aware web application project.

To change the CSS theme:

1 Open the **Application Factory Modeling** perspective by either of the following methods:

- **Window** ▸ **Open Perspective** ▸ **Other** ▸ **Application Factory Modeling**
- Use the toolbar icon**Open Perspective**. Select **Other** ▸ **Application Factory Modeling**

Note: If you have previously had the **Application FactoryModeling** perspective open, you can switch to it by clicking the **Application Factory Modeling perspective** icon on the toolbar.

2 Click on the **Scripts - Application Factory** view.

3 Double-click on the script name **Choose CSS Theme.js** to launch the script. The script can also be launched by right-clicking on the web application project in the **Application Factory Modeling** perspective and selecting **Data-Aware Application Tools** ▸ **Choose CSS Theme**.

4 Select the web application project from the dropdown menu.

5 Select from a pre-defined list of CSS themes from the dropdown menu. The available themes are:

- Simplicity
- Andreas 01
- Puzzle with Style

Selecting a style displays a preview image of the application in the dialog. Click **OK** to apply the selected theme.

6 Click the **Show Changes** button to display the changes that will be made by the script. This option brings up the changes listed in the **Script Learn/Resolve/Commit** view in the IDE. It provides the opportunity to examine each change and accept or discard changes at the file level. The **Script Learn/Resolve/Commit** view also provides user interface to resolve missing resources that are required by the script, as well as change the location in which code snippets are inserted into existing files.

7 Click on the **Save** button in the toolbar to commit all changes. This action writes out all changes to files.

8 Click the **Commit Now** button to commit all changes to files without browsing through and resolving changes.

9 Check the option **Do not show this dialog again** to always commit without browsing through changes made by a script run. Script commit behavior can be set in Application Factorypreferences under **Window** ▸ **Preferences** ▸ **Application Factory** .

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Creating a JSF Data-Aware Web Application

This topic describes how to create a JSF data-aware web application module through Application Factory.

To create a JSF data-aware application module:

1. Open or switch to the **Application Factory Repository Exploring** perspective (the default perspective) by one of the following paths:
   - Window ▸ Open Perspective ▸ Other ▸ Application Factory Repository Exploring
   - Click on the **Open Perspective** icon and select Other Application Factory Repository Exploring
   - If this perspective has been previously opened, you can switch back to it by clicking the **Application Factory Repository Exploring perspective** icon in the toolbar.

2. The **Application Factory Explorer** view is opened by default in this perspective. Click on the **Application Factory Explorer** view to make it the active window.
3. Double-click the template module **JSF Data-Aware Application** in the right-hand pane of the **Application Factory Explorer**. This opens this template application in the **Application Preview** pane.
4. Click **Create Application** button to import the JSF data-aware template application into the workspace
5. The**New JSF Data-Aware Web Application** wizard is launched. This is a 4–page or 5–page page wizard. On the **Web Application Settings** page:
   - Enter a project name in the **Project name** field.
   - Specify the directory for the project contents in the **Project contents** area. If the**Use default** box is checked, the default directory name is entered automatically in the **Directory** field. If unchecked, this field becomes active and you can specify or browse to your desired project content directory location.
   - Select the **Target runtime** from the server runtime dropdown menu. You can add a target runtime, by clicking the **New** button, selecting a runtime to add and completing the runtime information.
   - Select the **Default server** from the default server dropdown menu. You can add a default server, by clicking the **New** button, selecting a runtime to add and completing the server information.
   - Under the **Existing sources** field, check one of the options: **Create new project in workspace**, **Create new project from existing JPA project's source.**, or **Create project from database schema**. The latter option results in a fifth page being added to the wizard, which defines the table entities to use from the database. Table data can be imported after the project is created by right-clicking the project and selecting **Import Entities from the Database**.
   - Select the **Disable validators** option if you want to disable code validators for your application. If not disabled, the workbench validates your files automatically after any build or you can validate manually. Validators can also be reset after project creation by selecting the project as the active project in the workbench, and choosing Window ▸ Preference ▸ Validation and indicating the validators you want to enable or disable.
   - Select the **Switch off autobuild option for the workspace** option if you want to disable autobuilds for your workspace. This can also be switched on and off after the JSF application creation by checking or unchecking the Project ▸ Build Automatically option for the active project in the workspace.

   .
6. Click **Next** to configure persistence frameworks and database settings for the application. This opens page 2 (**Persistence Frameworks and Database Settings**) of the **New JSF Data-Aware Web Application** wizard. On the **Persistence Frameworks and Database Settings** page, you can define the following options:
   - Specify the persistence framework in the **Application Frameworks** area via the dropdown menu. You can specify JPA or Hibernate as your data persistence framework.

- Specify the database connection settings in the **Database Settings** area. Any active database connection appear in the drowdown menu of the **Connection** field . You can add a database connection by clicking **Add connection**. This walks you through a wizard to add a new database connection.
- Specify the database schema settings in the **Database Settings** area. Any active database schema appear in the drowdown menu of the **Schema** field .
- Specify the dialect of the interaction with underlying database in the **Dialect** field via the dropdown menu. A database dialect controls the behavior of the database objects and deals with DDL statements (metadata) executed against the database

Note: Refer to the Eclipse DTP (Eclipse Data Tools Project) documentation for information on configuring a new database connection. This documentation can be found in the Eclipse Help in JBuilder by following the IDE path **Help ▸ Help Contents ▸ Data Tools Platform <document name>**. The database schema required by the User/Login module is created and populated with default data in the selected database/schema.

7 Select **Next** to proceed to the **AppFuse Settings** page (page 3) of the **New JSF Data-Aware Web Applications** wizard. This page sets the AppFuse and Maven specific settings for the project. All the data-aware application modules are based on AppFuse (which is an open-source project based on popular Java and web application frameworks). Refer to the AppFuse Home Page for more details on AppFuse. On the **AppFuse Settings** dialog page, you can define the following options:

- Define the Maven **Artifact Id/Project Name**, **Group Id/Package**, and **Version** fields in the **Maven Settings** area. These fields are initially populated with default values. All data-aware web application projects are Maven projects that can be compiled and deployed using either WTP or Maven (from the command line).
- Check the option **Include AppFuse Framework sources** in the **AppFuse Settings** area to include AppFuse sources in the web application. Turning on this option extracts all base AppFuse classes for the persistence, business and front-end layers into the web application project. This option is turned off by default.
- Check the option **Use generic Manager classes** in the **AppFuse Settings** area to use AppFuse DAO and service classes during CRUD code generation. This option is turned on by default. AppFuse provides generic DAO and business classes to perform basic CRUD options from any database table. Unchecking this option generates custom DAO and manager classes during CRUD generation for an entity.
- Complete the fields in the **Application Mail Settings** to set up mail preferences.

8 Click **Next** to proceed to the **Modules Settings** page (page 4) of the **New JSF Data-Aware Web Applications** wizard. This page selects additional modules to be included in the web application. Available modules are:

- **User and Login Module**: This module includes user management with security implemented using ACEGI. The module also includes a login page. The default user and password for the user/login module is admin/ admin. This module is installed by default.
- **JasperReports Module**: This module includes generation of JasperReports in standard formats (HTML, CSV, Excel, Word) based on the user module. This module is not installed by default.
- **SearchModule**: This module includes the option to add search capabilities to the web application based on either Apache Lucene or Compass. This module is not installed by default. The default implementation provided for the Search module when enabled is Apache Lucene.

9 Click **Next** to proceed to the **Generate Entities from Tables** page (page 5) of the **New JSF Data-Aware Web Applications** wizard. This page is only available if you checked the field **Create project from database schema** on the **Web Application Settings** page (page 1) of the wizard. This page allows you to select tables and related entity names from an existing database to include in your data-aware web application project.

10 Click **Finish** from the last page of the wizard (page 4 or page 5) to complete the configuration wizard for the JSF data-aware web application module. This creates a web application project with the selected options and the Application Factory project containing tags, application diagram and code-generation scripts in the workspace.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
Article: Developing Web Applications with JavaServer Faces
JavaServer Faces Technology-Documentation
Hibernate Documentation
Adopting a Java Persistence Framework: Which, When, and What?
Eclipse Data Tools Platform Project Home Page
AppFuse Home Page
Apache Maven Project Page
ACEGI Security System for Spring Home Page
JasperReportsHome Page
Apache Lucene Overview and Documentation Page
Lucene Compass Home Page

Creating a Spring MVC Data-Aware Web Application

This topic describes how to create a Spring MVC data-aware web application module through Application Factory.

To create a Spring MVC data-aware application module:

1 Open or switch to the **Application Factory Repository Exploring** perspective (the default perspective) by one of the following paths:

- **Window ▸ Open Perspective ▸ Other ▸ Application Factory Repository Exploring**
- Click on the **Open Perspective** icon and select **Other Application Factory Repository Exploring**
- If this perspective has been previously opened, you can switch back to it by clicking the **Application Factory Repository Exploring perspective** icon in the toolbar.

2 The **Application Factory Explorer** view is opened by default in this perspective. Click on the **Application Factory Explorer** view to make it the active window.

3 Double-click the template module **Spring MVC Data-Aware Application** in the right-hand pane of the **Application Factory Explorer**. This opens this template application in the **Application Preview** pane.

4 Click **Create Application** button to import the Spring MVC data-aware template application into the workspace

5 The**New Spring MVC Data-Aware Web Application** wizard is launched. This is a 4-page or 5–page wizard. On the **Web Application Settings** page:

- Enter a project name in the **Project name** field.
- Specify the directory for the project contents in the **Project contents** area. If the**Use default** box is checked, the default directory name is entered automatically in the **Directory** field. If unchecked, this field becomes active and you can specify or browse to your desired project content directory location.
- Select the **Target runtime** from the server runtime dropdown menu. You can add a target runtime, by clicking the **New** button, selecting a runtime to add and completing the runtime information.
- Select the **Default server** from the default server dropdown menu. You can add a default server, by clicking the **New** button, selecting a runtime to add and completing the server information.
- Under the **Existing sources** field, check one of the options: **Create new project in workspace**, **Create new project from existing JPA project's source.**, or **Create project from database schema**. The latter option results in a fifth page being added to the wizard, which defines the table entities to use from the database. Table data can be imported after the project is created by right-clicking the project and selecting **Import Entities from the Database**.
- Select the **Disable validators** option if you want to disable code validators for your application. If not disabled, the workbench validates your files automatically after any build or you can validate manually. Validators can also be reset after project creation by selecting the project as the active project in the workbench, and choosing **Window ▸ Preference ▸ Validation** and indicating the validators you want to enable or disable.
- Select the **Switch off autobuild option for the workspace** option if you want to disable autobuilds for your workspace. This can also be switched on and off after the Spring MVC application creation by checking or unchecking the **Project ▸ Build Automatically** option for the active project in the workspace.

.

6 Click **Next** to configure persistence frameworks and database settings for the application. This opens page 2 (**Persistence Frameworks and Database Settings**) of the **New Spring MVC Data-Aware Web Application** wizard. On the **Persistence Frameworks and Database Settings** page, you can define the following options:

- Specify the persistence framework in the **Application Frameworks** area via the dropdown menu. You can specify JPA or Hibernate as your data persistence framework.

- Specify the database connection settings in the **Database Settings** area. Any active database connection appear in the drowdown menu of the **Connection** field . You can add a database connection by clicking **Add connection**. This walks you through a wizard to add a new database connection.
- Specify the database schema settings in the **Database Settings** area. Any active database schema appear in the drowdown menu of the **Schema** field .
- Specify the dialect of the interaction with underlying database in the **Dialect** field via the dropdown menu. A database dialect controls the behavior of the database objects and deals with DDL statements (metadata) executed against the database

Note: Refer to the Eclipse DTP (Eclipse Data Tools Project) documentation for information on configuring a new database connection. This documentation can be found in the Eclipse Help in JBuilder by following the IDE path **Help ▸ Help Contents ▸ Data Tools Platform <document name>**. The database schema required by the User/Login module is created and populated with default data in the selected database/schema.

7 Select **Next** to proceed to the **AppFuse Settings** page (page 3) of the **New Spring MVC Data-Aware Web Applications** wizard. This page sets the AppFuse and Maven specific settings for the project. All the data-aware application modules are based on AppFuse (which is an open-source project based on popular Java and web application frameworks). Refer to the AppFuse Home Page for more details on AppFuse. On the **AppFuse Settings** dialog page, you can define the following options:

- Define the Maven **Artifact Id/Project Name**, **Group Id/Package**, and **Version** fields in the **Maven Settings** area. These fields are initially populated with default values. All data-aware web application projects are Maven projects that can be compiled and deployed using either WTP or Maven (from the command line).
- Check the option **Include AppFuse Framework sources** in the **AppFuse Settings** area to include AppFuse sources in the web application. Turning on this option extracts all base AppFuse classes for the persistence, business and front-end layers into the web application project. This option is turned off by default.
- Check the option **Use generic Manager classes** in the **AppFuse Settings** area to use AppFuse DAO and service classes during CRUD code generation. This option is turned on by default. AppFuse provides generic DAO and business classes to perform basic CRUD options from any database table. Unchecking this option generates custom DAO and manager classes during CRUD generation for an entity.
- Complete the fields in the **Application Mail Settings** to set up mail preferences.

8 Click **Next** to proceed to the **Modules Settings** page (page 4) of the **New Spring MVC Data-Aware Web Applications** wizard. This page selects additional modules to be included in the web application. Available modules are:

- **User and Login Module**: This module includes user management with security implemented using ACEGI. The module also includes a login page. The default user and password for the user/login module is admin/ admin. This module is installed by default.
- **JasperReports Module**: This module includes generation of JasperReports in standard formats (HTML, CSV, Excel, Word) based on the user module. This module is not installed by default.
- **SearchModule**: This module includes the option to add search capabilities to the web application based on either Apache Lucene or Compass. This module is not installed by default. The default implementation provided for the Search module when enabled is Apache Lucene.

9 Click **Next** to proceed to the **Generate Entities from Tables** page (page 5) of the **New Spring MVC Data-Aware Web Applications** wizard. This page is only available if you checked the field **Create project from database schema** on the **Web Application Settings** page (page 1) of the wizard. This page allows you to select tables and related entity names from an existing database to include in your data-aware web application project.

10ClickFinish from the last page of the wizard (page 4 or page 5) to complete the configuration wizard for the Spring MVC data-aware web application module. This creates a web application project with the selected options and the Application Factory project containing tags, application diagram and code-generation scripts in the workspace.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
Spring Framework Home Page
Hibernate Documentation
Adopting a Java Persistence Framework: Which, When, and What?
Eclipse Data Tools Platform Project Home Page
AppFuse Home Page
Apache Maven Project Page
ACEGI Security System for Spring Home Page
JasperReportsHome Page
Apache Lucene Overview and Documentation Page
Lucene Compass Home Page

Creating a Struts 2 Data-Aware Web Application

This topic describes how to create a Struts 2 data-aware web application module through Application Factory.

To create a Struts 2 data-aware application module:

1. Open or switch to the **Application Factory Repository Exploring** perspective (the default perspective) by one of the following paths:
   - Window ▸ Open Perspective ▸ Other ▸ Application Factory Repository Exploring
   - Click on the **Open Perspective** icon and select Other Application Factory Repository Exploring
   - If this perspective has been previously opened, you can switch back to it by clicking the **Application Factory Repository Exploring perspective** icon in the toolbar.

2. The **Application Factory Explorer** view is opened by default in this perspective. Click on the **Application Factory Explorer** view to make it the active window.
3. Double-click the template module **Struts 2 Data-Aware Application** in the right-hand pane of the **Application Factory Explorer**. This opens this template application in the **Application Preview** pane.
4. Click **Create Application** button to import the Struts 2 data-aware template application into the workspace
5. The**New Struts 2 Data-Aware Web Application** wizard is launched. This is a 4-page or 5–page wizard. On the **Web Application Settings** page:
   - Enter a project name in the **Project name** field.
   - Specify the directory for the project contents in the **Project contents** area. If the**Use default** box is checked, the default directory name is entered automatically in the **Directory** field. If unchecked, this field becomes active and you can specify or browse to your desired project content directory location.
   - Select the **Target runtime** from the server runtime dropdown menu. You can add a target runtime, by clicking the **New** button, selecting a runtime to add and completing the runtime information.
   - Select the **Default server** from the default server dropdown menu. You can add a default server, by clicking the **New** button, selecting a runtime to add and completing the server information.
   - Under the **Existing sources** field, check one of the options: **Create new project in workspace**, **Create new project from existing JPA project's source.**, or **Create project from database schema**. The latter option results in a fifth page being added to the wizard, which defines the table entities to use from the database. Table data can be imported after the project is created by right-clicking the project and selecting **Import Entities from the Database**.
   - Select the **Disable validators** option if you want to disable code validators for your application. If not disabled, the workbench validates your files automatically after any build or you can validate manually. Validators can also be reset after project creation by selecting the project as the active project in the workbench, and choosing Window ▸ Preference ▸ Validation and indicating the validators you want to enable or disable.
   - Select the **Switch off autobuild option for the workspace** option if you want to disable autobuilds for your workspace. This can also be switched on and off after the Struts 2 application creation by checking or unchecking the Project ▸ Build Automatically option for the active project in the workspace.

   .
6. Click **Next** to configure persistence frameworks and database settings for the application. This opens page 2 (**Persistence Frameworks and Database Settings**) of the **New Struts 2 Data-Aware Web Application** wizard. On the **Persistence Frameworks and Database Settings** page, you can define the following options:
   - Specify the persistence framework in the **Application Frameworks** area via the dropdown menu. You can specify JPA or Hibernate as your data persistence framework.

- Specify the database connection settings in the **Database Settings** area. Any active database connection appear in the drowdown menu of the **Connection** field . You can add a database connection by clicking **Add connection**. This walks you through a wizard to add a new database connection.
- Specify the database schema settings in the **Database Settings** area. Any active database schema appear in the drowdown menu of the **Schema** field .
- Specify the dialect of the interaction with underlying database in the **Dialect** field via the dropdown menu. A database dialect controls the behavior of the database objects and deals with DDL statements (metadata) executed against the database

Note: Refer to the Eclipse DTP (Eclipse Data Tools Project) documentation for information on configuring a new database connection. This documentation can be found in the Eclipse Help in JBuilder by following the IDE path **Help ▸ Help Contents ▸ Data Tools Platform <document name>**. The database schema required by the User/Login module is created and populated with default data in the selected database/schema.

7 Select **Next** to proceed to the **AppFuse Settings** page (page 3) of the **New Struts 2 Data-Aware Web Applications** wizard. This page sets the AppFuse and Maven specific settings for the project. All the data-aware application modules are based on AppFuse (which is an open-source project based on popular Java and web application frameworks). Refer to the AppFuse Home Page for more details on AppFuse. On the **AppFuse Settings** dialog page, you can define the following options:

- Define the Maven **Artifact Id/Project Name**, **Group Id/Package**, and **Version** fields in the **Maven Settings** area. These fields are initially populated with default values. All data-aware web application projects are Maven projects that can be compiled and deployed using either WTP or Maven (from the command line).
- Check the option **Include AppFuse Framework sources** in the **AppFuse Settings** area to include AppFuse sources in the web application. Turning on this option extracts all base AppFuse classes for the persistence, business and front-end layers into the web application project. This option is turned off by default.
- Check the option **Use generic Manager classes** in the **AppFuse Settings** area to use AppFuse DAO and service classes during CRUD code generation. This option is turned on by default. AppFuse provides generic DAO and business classes to perform basic CRUD options from any database table. Unchecking this option generates custom DAO and manager classes during CRUD generation for an entity.
- Complete the fields in the **Application Mail Settings** to set up mail preferences.

8 Click **Next** to proceed to the **Modules Settings** page (page 4) of the **New Struts 2 Data-Aware Web Applications** wizard. This page selects additional modules to be included in the web application. Available modules are:

- **User and Login Module**: This module includes user management with security implemented using ACEGI. The module also includes a login page. The default user and password for the user/login module is admin/ admin. This module is installed by default.
- **JasperReports Module**: This module includes generation of JasperReports in standard formats (HTML, CSV, Excel, Word) based on the user module. This module is not installed by default.
- **SearchModule**: This module includes the option to add search capabilities to the web application based on either Apache Lucene or Compass. This module is not installed by default. The default implementation provided for the Search module when enabled is Apache Lucene.

9 Click **Next** to proceed to the **Generate Entities from Tables** page (page 5) of the **New Struts 2 Data-Aware Web Applications** wizard. This page is only available if you checked the field **Create project from database schema** on the **Web Application Settings** page (page 1) of the wizard. This page allows you to select tables and related entity names from an existing database to include in your data-aware web application project.

10 Click **Finish** from the last page of the wizard (page 4 or page 5) to complete the configuration wizard for the Struts 2 data-aware web application module. This creates a web application project with the selected options and the Application Factory project containing tags, application diagram and code-generation scripts in the workspace.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
Struts 2 Home Page
Hibernate Documentation
Adopting a Java Persistence Framework: Which, When, and What?
Eclipse Data Tools Platform Project Home Page
AppFuse Home Page
Apache Maven Project Page
ACEGI Security System for Spring Home Page
JasperReportsHome Page
Apache Lucene Overview and Documentation Page
Lucene Compass Home Page

Creating and Setting Up a Pet Store Module with the Pet Store Template

This topic describes how to create a Pet Store Java EE application module through the Pet Store template of Application Factory.

To create a PetStore application module from template:

1 Open or switch to the **Application Factory Repository Exploring** perspective (the default perspective) by one of the following paths:

- **Window ▸ Open Perspective ▸ Other ▸ Application Factory Repository Exploring**
- Click on the **Open Perspective** icon and select **Other Application Factory Repository Exploring**
- If this perspective has been previously opened, you can switch back to it by clicking the **Application Factory Repository Exploring perspective** icon in the toolbar.

2 The **Application Factory Explorer** view is opened by default in this perspective. Click on the **Application Factory Explorer** view to make it the active window.

3 Double-click the template module **Pet Store** in the right-hand pane of the **Application Factory Explorer**. This opens this template application in the **Application Preview** tab of the **Application Module Editor**. There are also tabs for Diagram, Tags, and License.

4 Click **Create Application** button to import the Pet Store template application into the workspace . A progress bar appears during import.

5 An**Import Application Module** dialog wizard appears asking if you want to run the application creation Setup.js script now or later. Click**Now** to immediately launch the Pet Store Setup.js script (see following procedure). If you select **Later**, the script can always be launched by:

- double-clicking on Setup.js in the **Script—Application Factory** view.
- right-clicking on Setup.js in the **Script—Application Factory** view and selecting **Execute Script**.

To setup the Pet Store application:

1 You can setup your new Pet Store application as part of the project creation (see above), or later from the **Setup.js** script that appears in the **Script—Application Factory** view.

2 The script can be launched by:

- double-clicking on **Setup.js** in the **Script—Application Factory** view.
- right-clicking on **Setup.js** in the **Script—Application Factory** view and selecting **Execute Script**.

3 The **Setup.js: Select Glassfish Server Runtime** dialog opens Select a server from the dropdown menu or click **New** to add a new runtime server. This opens a series of dialogs to add a runtime server.

When the correct runtime server is installed, click **OK** on the **Select Glassfish Server Runtime** page.

4 The **Setup.js: Select Domain** dialog opens. Select a server from the dropdown menu or click **New** to add a new runtime server. This opens a series of dialogs to add a runtime server.

When the correct runtime server is installed, click **OK** on the **Select Glassfish Server Runtime** page.

5 The **Setup.js: Select Database Connection** dialog opens Select a server from the dropdown menu or click **New** to add a new runtime server. This opens a series of dialogs to add a runtime server.

When the correct runtime server is installed, click **OK** on the **Select Glassfish Server Runtime** page.

6 Click**OK** to setup your Pet Store project. A progress bar appears and the new Pet Store module switches to the **Application Factory Modeling perspective** and opens the **Application Diagram** for the project.

7 You can now proceed with executing the other available Pet Store JavaScript functions. These scripts need to be executed in the following order:

1 Adding Tag Support

2 Adding Search Mechanism

3 Adding Maps

4 Adding Seller

5 Adding CAPTCHA

6 Adding RSS support bar

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Creating Tables

This topic describes how to create database tables and foreign-key relationships based on entities in the Data-Aware web application.

To create database tables based on entities:

1. Switch to the **Application Factory Modeling perspective**.
2. Click on the **Scripts - Application Factory** view
3. Double-click on the script **Create Tables from Entities.js** to launch the script. The script can also be launched by right-clicking on the project and selecting **Data-Aware Application Tools ▸ Create Tables**.
4. Select the web application project from the dropdownlist.
5. The create tables dialog displays a list of entities in the project and matching database tables in the currently active database schema for the project. Select tables that you want to create (or re-create) in the database schema and click **OK**.

   Note: Selecting entities that are associated with existing tables drops and re-creates the existing tables in the database.

6. Click the **Show Changes** button to display the changes that are to be made by the script. This option brings up the changes listed in the **Script Learn/Resolve/Commit** view. This provides the opportunity to examine each change and accept or discard changes at the file level. The **Script Learn/Resolve/Commit** view also provides UI to resolve missing resources that are required by the script, as well to change the location in which code snippets are inserted into existing files. Click on the **Save** button in the toolbar to commit all changes. This action writes out all changes to the files.
7. Click the **Commit Now** button to commit all changes to files without browsing through and resolving changes.
8. Check the option **Do not show this dialog again** to always commit without browsing through changes made by a script run. Script commit behavior can be set in Application Factory preferences under **Window ▸ Preferences ▸ Application Factory** .

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Exporting the Server Maven Configuration

All data-aware web application projects are Maven projects, which can be compiled and deployed using either WTP or Maven (from the command line). The data-aware application tooling currently supports exporting the WTP server runtime configuration to the Maven configuration file (pom.xml) for selected WTP servers (such as JBoss, Glassfish).

To export the WTP server runtime configuration to Maven:

1. Switch to the **Application Factory Modeling perspective**.
2. Right-click on the project and select **Data-Aware Application Tools** ▸ **Export server configuration to Maven**. This sets up a deployment profile, with all required dependencies, for the WTP runtime in the project's pom.xml.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Generating CRUD for an Entity

This topic describes how to generate CRUD for an entity or related entities for a data-aware web application project.

To create an entity:

1 Open the **Application Factory Modeling** perspective by either of the following methods:

- Window ▸ Open Perspective ▸ Other ▸ Application Factory Modeling perspective
- Use the toolbar icon **Open Perspective**. Select Other ▸ Application Factory Modeling

Note: If you have previously had the Application Factory **Modeling** Perspective open, you can switch to it by clicking the **Application Factory Modeling perspective** icon on the toolbar.

2 Expand the web application project (click on the + sign).

3 Double-click on the package containing the entities. If entities exist, proceed to the following subtask to create CRUD for the entities. If no entities exist, create entities by either of the following methods:

- Right-click on the web application project and select **Import entities from database** to create entities from database tables.
- Click on the **Entity** icon in the palette and drag/drop onto the Application Diagram to create a new entity. Use the context menu options for the entity to add fields/methods and so forth. Use the property editor for the bean/ fields/methods to modify any entity properties.

4 Go on to the following subtopic to generate CRUD for entities.

To generate CRUD for an entity or related entities:

1 If you have existing entities, you should follow Steps 1 through 2 in the preceding task to locate the entities.

If you do not have entities defined, complete the task above to create entities. When entities have been created, continue with the following steps to generate CRUD (Create, Read, Update, Delete) for the entities.

2 Double-click on the package containing the entities, or double-click on the project to open the default modeling diagram for the project and create a new package.

3 Click on each non-primary key field in the entity. Use the Data-Aware Application **Properties** tab to set properties for the view layer for the field. Select from the **Editor Type** dropdown to choose the UI widget that the database field is represented by (for example. text, password, radio button, and so forth). Check the option **Hide** in entity editor if you do not want to display the database field in an edit form. Check the option **Hide** in the entity list if you do not want to display the database field in the entity list.

4 Click on the **Scripts - Application Factory** view.

5 Double-click on the script **Generate CRUD from Entity.js** to launch the script. The script can also be launched by right-clicking on the entity in the diagram and selecting **Generate CRUD from Entity**.

6 This step only applies if the script was launched from the **Scripts - Application Factory** view.

- **Project** field: select the web application project name in the dropdown selection area.
- **Entity** field: select the entity in the dropdown selection area.

Click **OK**.

7 Click the **Show Changes** button to display the changes that are to be made by the script. This option brings up the changes listed in the **Script Learn/Resolve/Commit** view in the IDE. It provides the opportunity to examine each change and accept or discard changes at the file level. The **Script Learn/Resolve/Commit** view also

provides user interface to resolve missing resources that are required by the script, as well as to change the location in which code snippets are inserted into existing files.

8 Click on the **Save** button in the toolbar to commit all changes. This action writes out all changes to files.

9 Click the **Commit Now** button to commit all changes to files without browsing through and resolving changes.

10 Check the option **Do not show this dialog again** to always commit without browsing through changes made by a script run. Script commit behavior can be set in Application Factorypreferences under **Window ▸ Preferences ▸ Application Factory** .

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Generating CRUD with Master Detail

This topic escribes how to generate CRUD with master detail for an entity through Application Factory.

To create an entity:

1 Open the **Application Factory Modeling** perspective by either of the following methods:

- **Window** ▸ **Open Perspective** ▸ **Other** ▸ **Application Factory Modeling** perspective
- Use the toolbar icon **Open Perspective**. Select **Other** ▸ **Application Factory Modeling**

Note: If you have previously had the **Application Factory Modeling** perspective open, you can switch to it by clicking the **Application FactoryModeling perspective** icon on the toolbar

2 Expand the web application project (click on the + sign).

3 Double-click on the package containing the entities. If entities exist, proceed to the following subtask to create CRUD for the entities. If no entities exist, create entities by either of the following methods:

- Right-click on the web application project and select **Import entities from database** to create entities from database tables.
- Click on the **Entity** icon in the palette and drag/drop onto the Application Diagram to create a new entity. Use the context menu options for the entity to add fields/methods and so forth. Use the property editor for the bean/fields/methods to modify any entity properties.

4 Go on to the following subtopic to generate CRUD for entities with master detail.

To generate CRUD for an entity or related entities with master detail:

1 If you have existing entities, you should follow Steps 1 through 2 in the preceding task to locate the entities.

If you do not have entities defined, complete the task above to create entities. When entities have been created, continue with the following steps to generate CRUD (Create, Read, Update, Delete) for the entities with master detail.

2 Double-click on the package containing the entities.

3 To create relationships between entities, select the bi-directional or uni-directional relationship from the palette, click on the source bean and drag and release onto the target bean.

4 Click on the **CMR Application Factory** field and the relationship link to edit relationship properties.

5 Click on the **Scripts - Application Factory** view.

Click **OK**.

6 Double-click on the script **Generate CRUD from Entity.js** to launch the script. The script can also be launched by right-clicking on the entity in the diagram and selecting **Generate CRUD from Entity**.

7 This step only applies if the script was launched from the **Scripts - Application Factory** view.

- **Project** field: select the web application project name in the dropdown selection area.
- **Entity** field: select the entity in the dropdown selection area.

Click **OK**.

8 Click the **Show Changes** button to display the changes that are to be made by the script. This option brings up the changes listed in the **Script Learn/Resolve/Commit** view in the IDE. It provides the opportunity to examine each change and accept or discard changes at the file level. The **Script Learn/Resolve/Commit** view also

provides user interface to resolve missing resources that are required by the script, as well as to change the location in which code snippets are inserted into existing files.

9 Click on the **Save** button in the toolbar to commit all changes. This action writes out all changes to files.

10 Click the **Commit Now** button to commit all changes to files without browsing through and resolving changes.

11 Check the option **Do not show this dialog again** to always commit without browsing through changes made by a script run. Script commit behavior can be set in Application Factorypreferences under **Window** ▸ **Preferences,** ▸ **Application Factory** .

12 Run the **Generate CRUD from Entity.js** script for each entity in the relationship.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Running Maven Goals

All Data-Aware web application projects are Maven projects that can be compiled and deployed using either WTP or Maven (from the command line). This topic describes how to invoke commonly used Maven goals from the IDE.

To invoke commonly used Maven goals from the IDE:

1 Switch to the **Application Factory Modeling** perspective.

2 Right-click on the project and select **Data-Aware Application Tools** ▸ **Run Maven Goals**. Select from the following Maven goals:

- **Clean**—invokes the clean goal for the project.
- **Generate Tables**—invokes the hbm2ddl goal for the project, which creates (or re-creates) database schema based on project entities.
- **Run Jetty**—starts Jetty and deploys the project.
- **Package**—invokes the package goal for the project, which produces a deployable web archive (war).
- **Run Integration Tests**—runs the integration-test goal for the project.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Creating E-commerce Applications With Application Factory

This page links to the various procedural topics for E-commerce applications with Application Factory.

In This Section

Adding and Editing E-Commerce Products
Describes how to add new products and edit existing products to your E-commerce applicaton.

Creating an E-commerce Application from the Template Application
Describes how to create an E-commerce application module using the OfBiz E-commerce template.

Debugging an E-Commerce Application
Describes how to create and designate an add-on application module.

Opening an E-Commerce Application
Describes how to open an E-commerce application.

Running an E-Commerce Application
Describes how to run an E-commerce application

Setting up a Catalog
Describes how to set up a catalog for your E-commerce application.

Setting up an E-Commerce Store
Describes how to use the E-commerce template scripts to set up a store.

Specifying E-Commerce Store Parameters through JavaScripts
Describes how to use the E-commerce template scripts to define store parameters

Stopping an E-Commerce Application
Describes how to stop an E-commerce application.

Adding and Editing E-Commerce Products

This topic describes how add new products and edit existing products to your E-commerce applicaton.

To add products to your E-commerce store:

1 If not open, open the **Script —Application Factory** view by selecting the menu pathWindow ▸ Show View ▸ Others ▸ Application Factory ▸ Script —Application Factory .

2 Locate E-commerce application store and expand the tree structure to view all available task-related JavaScripts. Right-click on the add product script name, select Execute Script, and the following dialogs open to allow you to further define your product.

3 If not open, open the **Script —Application Factory** view by selecting the menu pathWindow ▸ Show View ▸ Others ▸ Application Factory ▸ Script —Application Factory .

4 **Add New Product.js**—executing this script opens a multi-page wizard. for adding a new product.

- **Setup Product Wizard: Edit Product (page 1)**—to define the product properties. There are basic and advanced modes that determine what properties can be set.
- **Setup Product Wizard: Category Members (page 2)**—to edit category members. Click Add to open Add Category Member dialog.
- **Setup Product Wizard: Override default content (page 3)**—to specify content properties
- **Setup Product Wizard: Product Prices (page 4)**—specifies product prices.
- **Setup Product Wizard: Product Keywords (page 5)**—specifies product keywords.
- **Setup Product Wizard: Edit product features (page)**—allows you to edit product features. Click **Add** to open **Add new feature** dialog. Click on **. . .** after the **Product Feature** field in this dialog to open the **Lookup dialog** page that allows you to input parameters, find results and choose the needed values.

5 Click **Finish** to implement the product changes/additions.

To edit product:

1 If not open, open the **Script —Application Factory** view by selecting the menu path Window ▸ Show View ▸ Others ▸ Application Factory ▸ Script —Application Factory .

2 Locate E-commerce application store setup JavaScript (**Edit Product.js**), The script can be launched by:

- double-clicking on **Edit Product.js** in the **Script—Application Factory** view.
- right-clicking on **Edit Product.js** in the **Script—Application Factory** view and selecting **Edit Product.js**.

3 This script launches a multipage wizard for editing products.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Creating E-commerce Applications With Application Factory
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Creating an E-commerce Application from the Template Application

This topic describes how to create an E-commerce application module using the OfBiz E-commerce template.

To create a new E-commerce Module Application from the Application Factory template application:

1 Open or switch to the **Application Factory Repository Exploring** perspective (the default perspective) by one of the following paths:

- Window ▸ Open Perspective ▸ Other ▸ Application Factory Repository Exploring
- Click on the **Open Perspective** icon and select Other Application Factory Repository Exploring
- If this perspective has been previously opened, you can switch back to it by clicking the **Application Factory Repository Exploring perspective** icon in the toolbar.

2 In the default **Application Factory Explorer** view, all available framework files appear in the right-side pane. Select the link to **Ecommerce Application** or further filter the right-side pane to specifically locate that application.

> **Note:** To filter the view in the **Application Factory Explorer** view for only the E-commerce applications, deselect the **Framework** item on the left-side of the pane. Reselect only the **Framework** sub-item **Ecommerce.** Selecting only the **Ecommerce** shows only the available E-commerce template applications in the right-side pane

3 When the **ECommerce Application** link is selected in the right-side pane of the**Application Factory Explorer** view, the application opens in the **Application Module Editor**. It initially opens in the **Application Preview** tab. There are also tab views for Diagram, Tags , and License information.

4 Click **Create Application** to create a new application.

> **Note:** An Application Factory project must exist in the workspace to work with application modules; however, template applications create the Application Factory project along with the template module.

5 **Create New ECommerce Application Project** wizard opens. This is the **ECommerce Application Project Settings** page. Use the default values or enter alternate data in the following fields;

- **Project name**: specifies the Ecommerce project name. There is no default value so you must enter a name.
- **Project contents**: specifies the default directory for the project content. You can specify your own directory name if you uncheck the **Use default** box and specify or browse to the desired directory.
- **Data loaded**: specifies that the demo products are loaded. This is the default value. Uncheck the **Load Demo products** option if you do not want to load the demo products.
- **Use embedded derby database connection** :specfies that the embedded Apache Derby database is used so no database setup is required. If this option is unchecked, you are required to setup another database.

6 Click **OK** to create the DOM plugin project. The project appears in the **Package Explorer** view list and the **Ecommerce Application Configuration** dialog opens. Specify the look and feel you want for you Ecommerce application.

7 Specify the look and feel you want for your Ecommerce application in the **Ecommerce Application Configuration** dialog. When selected, each option is presented in a graphic format below the dropdown menu.

8 Click **Next** . The Ecommerce application loads and the **Set Properties** dialog opens.

9 In the **Set Properties** dialog, you are setting the properties used on your E-commerce web page in the header and footer areas. Specify the desired data in the following fields:

- Store Name
- Company Name
- Title
- Subtitle
- Logo image
- Footer text
- Footer website

Click **OK**.

10 Your Ecommerce project is loaded and opens to the application diagram. The application diagram and the **Scripts — Application Factory** view show the various project components and provided JavaScript for altering these components. The **Package Explorer** view shows the new Ecommerce project in a tree-structure.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Creating E-commerce Applications With Application Factory
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Debugging an E-Commerce Application

When working with an Application Module, the workspace must contain an existing Application Factory project.

To debug an E-commerce application:

1. If not open, open the **Package Explorer** view by selecting the menu path**Window ▶ Show View ▶ Package Explorer**.
2. Locate your E-commerce project. Right-click on the project name and select **ECommerce Application ▶ Debug ECommerce Application**.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Opening an E-Commerce Application

To open an E-commerce application:

1. If not open, open the **Package Explorer** view by selecting the menu path**Window ▸ Show View ▸ Package Explorer**.
2. Locate your E-commerce project. Right-click on the project name and select **ECommerce Application ▸ Open ECommerce Application Site**.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Running an E-Commerce Application

This topic describes how to run an E-commerce application

To run an E-commerce application:

1 If not open, open the **Package Explorer** view by selecting the menu path**Window ▸ Show View ▸ Package Explorer**.

2 Locate your E-commerce project. in the **Package Explorer** view.

3 Right-click on the project name and select **ECommerce Application ▸ Run ECommerce Application**.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Creating E-commerce Applications With Application Factory
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Setting up a Catalog

This topic describes how to set up a catalog for your E-commerce application.

To setup catalog:

1 If not open, open the **Script —Application Factory** view by selecting the menu path **Window** ▸ **Show View** ▸ **Others** ▸ **Application Factory** ▸ **Script —Application Factory** .

2 Locate E-commerce application store setup JavaScript (**Setup Catalog.js**), The script can be launched by:

- double-clicking on **Setup Catalog.js** in the **Script—Application Factory** view.
- right-clicking on **Setup Catalog.js** in the **Script—Application Factory** view and selecting **Execute Script**.

3 This script launches a multipage wizard.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Creating E-commerce Applications With Application Factory
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Setting up an E-Commerce Store

This topic describes how to use the E-commerce template scripts to set up a store. When you create the E-commerce application using the E-commerce template module, many task-related JavaScripts are created and appear in the **Script —Application Factory** view. Executing these scripts opens dialogs that allow you to customize your store.

To set up an E-commerce store:

1 If not open, open the **Script —Application Factory** view by selecting the menu path **Window** ▸ **Show View** ▸ **Others** ▸ **Application Factory** ▸ **Script —Application Factory** .

2 Locate E-commerce application store setup JavaScript (**Setup Stores.js**), The script can be launched by:

- double-clicking on **Setup Stores.js** in the **Script—Application Factory** view.
- right-clicking on **Setup Stores.js** in the **Script—Application Factory** view and selecting **Execute Script**.

3 Executing the **Setup Stores.js** script launches a multi-page wizard.

4 Click **Finish** view to complete the setup of your E-commerce store.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Specifying E-Commerce Store Parameters through JavaScripts

This topic describes how to use the E-commerce an module to created an E-commerce application . When you create the E-commerce application using the E-commerce template module, many task-related JavaScripts are created and appear in the **Script —Application Factory** view. Executing these scripts opens dialogs that allow you to customize your store. You can execute these scripts at any time:

These scripts fall into several categories according to type of function:

- Creating ECommerce Application (see task topic Setting up an E-Commerce Store)
- Setting up Store and Store Details (see task topic Setting up an E-Commerce Store)
- Adding/Editing Product (see task topic Adding and Editing an E-Commerce Product)
- Setting Product Details
- Editing Products Features
- Setting Product Details
- Setting up Catalog and Catalog Details (see task topic Setting up a Catalog
- Edit Payment Properties
- Edit Shipment Properties

To add a new product:

1. If not open, open the **Script —Application Factory** view by selecting the menu pathWindow ▶ Show View ▶ Others ▶ Application Factory ▶ Script —Application Factory .
2. Locate E-commerce application store and expand the tree structure to view all available task-related JavaScripts. Right-click on the add product script name, select Execute Script, and the following dialogs open to allow you to further define your product.
3. If not open, open the **Script —Application Factory** view by selecting the menu pathWindow ▶ Show View ▶ Others ▶ Application Factory ▶ Script —Application Factory .
4. **Add New Product.js**—executing this script opens a multi-page wizard. for adding a new product.
   - **Setup Product Wizard: Edit Product (page 1)**—to define the product properties. There are basic and advanced modes that determine what properties can be set.
   - **Setup Product Wizard: Category Members (page 2)**—to edit category members. Click Add to open Add Category Member dialog.
   - **Setup Product Wizard: Override default content (page 3)**—to specify content properties
   - **Setup Product Wizard: Product Prices (page 4)**—specifies product prices.
   - **Setup Product Wizard: Product Keywords (page 5)**—specifies product keywords.
   - **Setup Product Wizard: Edit product features (page)**—allows you to edit product features. Click **Add** to open **Add new feature** dialog. Click on **. . .** after the **Product Feature** field in this dialog to open the **Lookup dialog** page that allows you to input parameters, find results and choose the needed values.
5. Click **Finish** to implement the product changes/additions.

Change WebSite for ECommerce.js:

1. **Select project** (page 1) specified project name.
2. **Set Properties** (page 2) sets the WebSite ID.

3 **File Changes Commit Now**

Create ECommerce Application.js:

1 **Create New ECommerce Application Project: ECommerce Application Project Settings** (page 1) specifies project name, location and necessary data to be loaded by default.

2 **Ecommerce Application Configuration** (page 2) specifies application look and feel.

3 Set Properties (page 3) sets properties for E-commerce application.

4 THIS ISN'T DONE — wasn't sure what it was doing so I stopped.

CyberSource Payment Setup.js Dialogs:

1 **Select project** (page 1) specifies project name.

2 **Set Properties** (page 2) specifies CyberSource payment properties.

3 **File Changes Commit Now**

PayPal Payment Setup.js:

1 Select Project (page 1) selects the project where you want to edit PayPal Payment Properties.

2 Set Properties (page 2) sets PayPal payment properties.

3 File Changes (page 3) checks to see the file changes before they are committed.

4 Select **Show Changes** or **Commit Now**.

PayPal Processor Details.js:

1 Select Project (page 1) sets PayPal payment properties.

2 Set Payment Properties (page 2) sets payment processor details.

3 File Changes (page 3) checks to see the file changes before they are committed.

4 Select **Show Changes** or **Commit Now**.

For a complete list of wizards, please see the UI. They are used in a similar fashion as described above.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Stopping an E-Commerce Application

To stop an E-commerce application:

1 If not open, open the **Package Explorer** view by selecting the menu path**Window ▸ Show View ▸ Package Explorer**.

2 Locate your E-commerce project. Right-click on the project name and select **ECommerce Application ▸ Stop ECommerce Application**.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Template Applications

This topic describes the template applications that come packaged with your JBuilder or JGear product.

In This Section

Book Store Template Application
Describes the pre-existing book store template application that comes with your JBuilder or JGear product.

Pet Store Template Application
Describes the pre-packaged Pet Store template application that comes with your JBuilder or JGear product.

Book Store Template Application

This section describes task topics for using the pre-existing book store template application that comes with your JBuilder or JGear product.

In This Section

Creating a BookStore Module Application

This topic describes how to create a application module using the BookStore Template available with Application Factory.

To create a BookStore application module:

1 Open or switch to the **Application Factory Repository Exploring** perspective (the default perspective) by one of the following paths:

- **Window ▸ Open Perspective ▸ Other ▸ Application Factory Repository Exploring**
- Click on the **Open Perspective** icon and select **Other Application Factory Repository Exploring**
- If this perspective has been previously opened, you can switch back to it by clicking the **Application Factory Repository Exploring perspective** icon in the toolbar.

2 The **Application Factory Explorer** view is opened by default in this perspective. Click on the **Application Factory Explorer** view to make it the active window.

3 Double-click the template module **BookStore** in the right-hand pane of the **Application Factory Explorer**. This opens this template application in the **Application Preview** pane.

4 Click **Create Application** button to import the BookStore template application into the workspace

5 An**Import Application Module** dialog wizard appears asking if you want to run the application creation **Create Bookstore Project.js** application creation script now or later. Click**Now** to immediately launch the **Create Bookstore Project.js** script. If you select **Later**, the script can always be launched by:

- double-clicking on **Create Bookstore Project.js** in the **Script—Application Factory** view.
- right-clicking on **Create Bookstore Project.js** in the **Script—Application Factory** view and selecting **Execute Script**.

6 The**New BookStore Application** wizard is launched. This is a 4–page or 5–page page wizard. On the **Web Application Settings** page:

- Enter a project name in the **Project name** field.
- Specify the directory for the project contents in the **Project contents** area. If the**Use default** box is checked, the default directory name is entered automatically in the **Directory** field. If unchecked, this field becomes active and you can specify or browse to your desired project content directory location.
- Select the **Target runtime** from the server runtime dropdown menu. You can add a target runtime, by clicking the **New** button, selecting a runtime to add and completing the runtime information.
- Select the **Default server** from the default server dropdown menu. You can add a default server, by clicking the **New** button, selecting a runtime to add and completing the server information.
- Under the **Existing sources** field, check one of the options: **Create new project in workspace**, **Create new project from existing JPA project's source.**, or **Create project from database schema**. The latter option results in a fifth page being added to the wizard, which defines the table entities to use from the database. Table data can be imported after the project is created by right-clicking the project and selecting **Import Entities from the Database**.
- Select the **Disable validators** option if you want to disable code validators for your application. If not disabled, the workbench validates your files automatically after any build or you can validate manually. Validators can also be reset after project creation by selecting the project as the active project in the workbench, and choosing **Window ▸ Preference ▸ Validation** and indicating the validators you want to enable or disable.

.

7 Click **Next** to configure persistence frameworks and database settings for the application. This opens page 2 (**Persistence Frameworks and Database Settings**) of the **New BookStore Application** wizard. On the **Persistence Frameworks and Database Settings** page, you can define the following options:

- Specify the persistence framework in the **Application Frameworks** area via the dropdown menu. You can specify JPA or Hibernate as your data persistence framework.
- Specify the database connection settings in the **Database Settings** area. Any active database connection appear in the drowdown menu of the **Connection** field . You can add a database connection by clicking **Add connection**. This walks you through a wizard to add a new database connection.
- Specify the database schema settings in the **Database Settings** area. Any active database schema appear in the dropdown menu of the **Schema** field .
- Specify the dialect of the interaction with underlying database in the **Dialect** field via the dropdown menu. A database dialect controls the behavior of the database objects and deals with DDL statements (metadata) executed against the database

Note: Refer to the Eclipse DTP (Eclipse Data Tools Project) documentation for information on configuring a new database connection. This documentation can be found in the Eclipse Help in JBuilder by following the IDE path **Help ▸ Help Contents ▸ Data Tools Platform <document name>**. The database schema required by the User/Login module is created and populated with default data in the selected database/schema.

8 Select **Next** to proceed to the **Customize BookStore** page (page 3) of the **New BookStore Application** wizard. This page customizes settings for your Book Store.

- **Root package name** —specifies the root package name of your Book Store application module.
- **Store name** —specifies the name of the store.
- **Store tagline**—specifies any tagline phrase for the store.
- **Company name**—specifies the company name
- **Company URL**—specifies the company URL
- **Sample Data (CSV)**—specifies the path to a CSV (comma-separated value) file of sample data.

9 Click**Finish** to complete the configuration wizard for the BookStore application module. This creates a web application project with the selected options and the Application Factory project containing tags, application diagram and code-generation scripts in the workspace.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
Article: Developing Web Applications with JavaServer Faces
JavaServer Faces Technology-Documentation
Hibernate Documentation
Adopting a Java Persistence Framework: Which, When, and What?
Eclipse Data Tools Platform Project Home Page

Pet Store Template Application

The Pet Store Template Application is not implemented for BETA testing.

In This Section

Adding a Search Function to the Pet Store Application

This topic describes how to add search function to the Pet Store application module.

To add a search function to the Pet Store:

1 You can add search functionality to your new Pet Store application from the **02_Add_Search.js** script that appears in the **Script—Application Factory** view after project creation.

2 The script can be launched by:

- double-clicking on **02_Add_Search.js** in the **Script—Application Factory** view.
- right-clicking on **02_Add_Search.js** in the **Script—Application Factory** view and selecting **Execute Script**.

3 A **File Changes** dialog opens. This dialogs asks whether you want to see the file changes before the commit. You can click **Show Changes** to see the changes proposed before you commit them. This opens the **Script Learn/Resolve/ Commit** view where you can resolve problems and later commit by using the **Commit Changes** icon.

Click **Commit Now** to commit the changes immediately. You can also check **Always show file changes before commit instead of this dialog** if you do not want to see this dialog again.

4 When the commit progress bar completes, search support is now enabled in your Pet Store project. A Search box appears in the **Application Diagram**.

5 You can now proceed with executing the other available Pet Store JavaScript functions. These scripts need to be executed in the following order:

1 Adding Tag Support

2 Adding Search Mechanism

3 Adding Maps

4 Adding Seller

5 Adding CAPTCHA

6 Adding RSS support bar

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Adding an RSS Feed Bar to the Pet Store Application

This topic describes how to add an RSS feed bar to the Pet Store application module.

To add an RSS feed bar in the Pet Store application:

1. You can add RSS feed bar functionality to your new Pet Store application from the **Add_RSS_Bar.js** script that appears in the **Script—Application Factory** view after project creation.
2. The script can be launched by:
   - double-clicking on **Add_RSS_Bar.js** in the **Script—Application Factory** view.
   - right-clicking on **Add_RSS_Bar.js** in the **Script—Application Factory** view and selecting **Execute Script**.
3. The **Add_RSS_Bar.js: Enter RSS feed XML URL** dialog appears. Enter any valid RSS feed value. This field defaults to a CodeGear blog feed. Click **OK** .
4. A **File Changes** dialog opens. This dialogs asks whether you want to see the file changes before the commit. You can click **Show Changes** to see the changes proposed before you commit them. This opens the **Script Learn/Resolve/ Commit** view where you can resolve problems and later commit by using the **Commit Changes** icon.

   Click **Commit Now** to commit the changes immediately. You can also check **Always show file changes before commit instead of this dialog** if you do not want to see this dialog again.
5. When the commit progress bar completes, RSS feed bar support is now enabled in your Pet Store project. An RSS entity appears in the **Application Diagram**.
6. You can now proceed with executing the other available Pet Store JavaScript functions. These scripts need to be executed in the following order:

1. Adding Tag Support
2. Adding Search Mechanism
3. Adding Maps
4. Adding Seller
5. Adding CAPTCHA
6. Adding RSS support bar

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Adding CAPTCHA to the Pet Store Application

This topic describes how to add CAPTCHA (Completely Automated Turing Test To Tell Computers and Humans Apart) functionailty to the Pet Store application module. CAPTCHA is a program that can generate and grade tests that humans can pass but current computer programs cannot. For example, humans can read distorted text that is shown on a web store site, but computer programs cannot read such distored text. This gives an extra measure of security to your Pet Store transactions.

To add captcha information to the Pet Store:

1 You can add CAPTCHA functionality to your new Pet Store application from the **05_Add_Captcha.js** script that appears in the **Script—Application Factory** view after project creation.

2 The script can be launched by:

- double-clicking on **05_Add_Captcha.js** in the **Script—Application Factory** view.
- right-clicking on **05_Add_Captcha.js** in the **Script—Application Factory** view and selecting **Execute Script**.

3 A **File Changes** dialog opens. This dialogs asks whether you want to see the file changes before the commit. You can click **Show Changes** to see the changes proposed before you commit them. This opens the **Script Learn/Resolve/ Commit** view where you can resolve problems and later commit by using the **Commit Changes** icon.

Click **Commit Now** to commit the changes immediately. You can also check **Always show file changes before commit instead of this dialog** if you do not want to see this dialog again.

4 When the commit progress bar completes, CAPTCHA support is now enabled in your Pet Store project. A Captcha entity appears in the **Application Diagram**.

5 You can now proceed with executing the other available Pet Store JavaScript functions. These scripts need to be executed in the following order:

1 Adding Tag Support

2 Adding Search Mechanism

3 Adding Maps

4 Adding Seller

5 Adding CAPTCHA

6 Adding RSS support bar

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Adding Map Functionality to the Pet Store Application

This topic describes how to add map functionality to the Pet Store application module.

To add map functionality to the Pet Store:

1 You can add map functionality to your new Pet Store application from the **03_Add_Maps.js** script that appears in the **Script—Application Factory** view after project creation.

2 The script can be launched by:

- double-clicking on **03_Add_Maps.js** in the **Script—Application Factory** view.
- right-clicking on **03_Add_Maps.js** in the **Script—Application Factory** view and selecting **Execute Script**.

3 The **03_Add_Maps.js: Enter Google Maps API Key** dialog appears. It defaults to the Google Maps API Key. Click **OK** to use the default value.

4 A **File Changes** dialog opens. This dialogs asks whether you want to see the file changes before the commit. You can click **Show Changes** to see the changes proposed before you commit them. This opens the **Script Learn/Resolve/ Commit** view where you can resolve problems and later commit by using the **Commit Changes** icon.

Click **Commit Now** to commit the changes immediately. You can also check **Always show file changes before commit instead of this dialog** if you do not want to see this dialog again.

5 When the commit progress bar completes, map support is now enabled in your Pet Store project. A Map entity appears in the **Application Diagram**.

6 You can now proceed with executing the other available Pet Store JavaScript functions. These scripts need to be executed in the following order:

1 Adding Tag Support

2 Adding Search Mechanism

3 Adding Maps

4 Adding Seller

5 Adding CAPTCHA

6 Adding RSS support bar

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Adding Sellers to the Pet Store Application

This topic describes how to add seller information to the Pet Store application module.

To add sellers to the Pet Store:

1. You can add map functionality to your new Pet Store application from the **04_Add_Seller.js** script that appears in the **Script—Application Factory** view after project creation.
2. The script can be launched by:
   - double-clicking on **04_Add_Seller.js** in the **Script—Application Factory** view.
   - right-clicking on **04_Add_Seller.js** in the **Script—Application Factory** view and selecting **Execute Script**.
3. A **File Changes** dialog opens. This dialogs asks whether you want to see the file changes before the commit. You can click **Show Changes** to see the changes proposed before you commit them. This opens the **Script Learn/Resolve/ Commit** view where you can resolve problems and later commit by using the **Commit Changes** icon.

   Click **Commit Now** to commit the changes immediately. You can also check **Always show file changes before commit instead of this dialog** if you do not want to see this dialog again.
4. When the commit progress bar completes, seller support is now enabled in your Pet Store project. A Seller entity appears in the **Application Diagram**.
5. You can now proceed with executing the other available Pet Store JavaScript functions. These scripts need to be executed in the following order:

1. Adding Tag Support
2. Adding Search Mechanism
3. Adding Maps
4. Adding Seller
5. Adding CAPTCHA
6. Adding RSS support bar

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Adding Tag Support to the Pet Store Application

To add tags to the Pet Store application:

1 You can add tags to your new Pet Store application from the **01_Add_Tag_Support.js** script that appears in the **Script—Application Factory** view after project creation.

2 The script can be launched by:

- double-clicking on **01_Add_Tag_Support.js** in the **Script—Application Factory** view.
- right-clicking on **01_Add_Tag_Support.js** in the **Script—Application Factory** view and selecting **Execute Script**.

3 A **File Changes** dialog opens. This dialogs asks whether you want to see the file changes before the commit. You can click **Show Changes** to see the changes proposed before you commit them. This opens the **Script Learn/Resolve/ Commit** view where you can resolve problems and later commit by using the **Commit Changes** icon.

Click **Commit Now** to commit the changes immediately. You can also check **Always show file changes before commit instead of this dialog** if you do not want to see this dialog again.

4 When the commit progress bar completes, tag support is now enabled in your Pet Store project. A Tag entity appears in the **Application Diagram**.

5 You can now proceed with executing the other available Pet Store JavaScript functions. These scripts need to be executed in the following order:

1 Adding Tag Support

2 Adding Search Mechanism

3 Adding Maps

4 Adding Seller

5 Adding CAPTCHA

6 Adding RSS support bar

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Using Archeology Views

Java EE components are assembled into an application and are deployed to production, to be run and managed by the Java EE server. Use the following links to discover detailed information about creating Java EE applications using JBuilder 2008.

In This Section

Displaying File Archeology
Describes the file archeology functions used to display information from the scripts runs that have changed a file.

Displaying Project Archeology
Describes procedures to display information, from the project level, from the scripts runs that have changed a file within that project.

Displaying Script Archeology
Describes the script archeology functions used to display information from the scripts runs that have changed a file.

Filtering with Archeology View
Describes how to filter using the **Archeology** view.

Focusing on a Script Run through the Archeology View
Describes how to focus on a script run using the **Archeology** view.

Displaying File Archeology

The **Archeology** view displays script run history for a project, file, or script. This topic describes the procedure to display information, at a file level, for the scripts runs that have changed a file.

To display script run change information at the file level:

1. Right-click on any file in the **Package Explorer**, **Navigator**, or **Scripts—Application Factory** views.
2. Select the path option **Application Factory** ▸ **Open Script Run Archeology**.
3. All script runs that have changed the file are displayed. If a script run involved in multiple scripts, these are listed as children of the script run.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Displaying Project Archeology

The **Archeology** view displays script run history for a project, file, or script. This topic describes the procedure to display information, from the project level, for the scripts runs, that have changed files within that project.

To display script run change information at the project level:

1. Right-click on Application Factory project in the **Package Explorer**, **Navigator**, or **Model Navigator** views.
2. Select the path option **Application Factory** ▸ **Open All Script Run Archeology**. This displays all script runs executed for the Application Factory project. If a script run involved in multiple scripts, these are listed as children of the script run, so the files modified by each script can be viewed.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Displaying Script Archeology

The **Archeology** view displays script run history for a project, file, or script. This topic describes the procedure to display information, from the script level, for the scripts runs that have changed a script.

To display script run change information at the script level:

1. Right-click on any script in the **Scripts – Application Factory** view.
2. Select the option **Open Script Run Archeology**.
3. All script runs for the selected script are displayed in the **Archeology** view. If a script run involved multiple scripts, these are listed as children of the script run, so the files changed by each script can be viewed.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Filtering with Archeology View

The **Archeology** view displays script run history for a project, file, or script. This topic describes how to filter using the **Archeology** view.

To filter using the Archeology view:

1. Open the **Archeology** view for the desired project, script, or file by right-clicking on the item in the in the **Package Explorer**, **Navigator**, or **Scripts-Application Factory** views.
2. Select the path option **Application Factory** ▸ **Open Script Run Archeology**.
3. In the **type filter text** area, type in either a script name or author name. Only scripts matching the entered criteria are displayed in the script run list. Delete the filter text to turn off the filter.
4. To filter by date, check the **Date** box and enter a date range in the **Start** and **End** areas to filter the script runs by date range. Deselect the **Date**box to remove the filter from the view.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Focusing on a Script Run through the Archeology View

The **Archeology** view displays script run history for a project, file, or script. This topic describes how to focus on a script run using the **Archeology** view.

To focus on a script run:

1. Open the **Archeology** view for the desired project, script or file by right-clicking on the item in the in the **Package Explorer**, **Navigator**, or **Scripts-Application Factory** views.
2. Select the path option **Application Factory** ▸ **Open Script Run Archeology**.
3. Select a script run from the list. If a script run involved in multiple scripts, these are listed as children of the script run, so the files modified by each script can be viewed. The focus can only be selected from the parent script run.
4. Click the **Apply selected Script Run focus to main views** icon in the toolbar. This focuses the **Package Explorer**, **Navigator** and **Scripts - Application Factory** views on the resources that were affected by the selected script run.

Note: The same action can be performed by right-clicking on the script run in the list and selecting the context menu option **Apply selected Script Run focus to main views**.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Using Scripts

This section details how to use Application Factory script functionality.

In This Section

Accessing Javadoc for the DOM API

Javadoc is a tool for generating API documentation in HTML format from comments in the source code. Javadoc for the DOM API is provided with JBuilder. This Javadoc can be accessed from within a script while the user is in the process of editing a script

To access Javadoc for the DOM API within JBuilder:

1. Open the **Scripts - Application Factory** view.
2. Right-click on any script and select **Edit**.
3. Right-click on the DOM API declarations in the script and select **Application Factory** ▶ **Navigate to DOM JavaDoc**.
4. The Javadoc for the DOM API is opened in the internal web browser.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts

Related Reference

Application Factory Dialogs and Preferences Reference

Creating a Connecting Script

The **Scripts—Application Factory** view allows you to select multiple scripts and create a connecting script to chain script runs. The resultant script invokes each script, collects output from the script, and passes it as an argument to the next script run.

To create a connecting script:

1 Open the **Scripts—Application Factory** view by selecting either of the following paths:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Scripts—Application Factory**
- **Window** ▸ **Show View** ▸ **Other** and type `Scripts-Application Factory` in the **type filter text** field.

.

2 Select the scripts whose output you want to chain, right click-and select **Create Connecting Script.**

3 A dialog with the list of selected scripts to run with this script is displayed. Use the **Up** or **Down** buttons to reorder the scripts. Click on **Add** to designate additional scripts to connect to this script.

4 Click on **Save**, select a target directory and provide a file name for the generated script.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Creating a Template

Scripting functionality in Application Factory uses FreeMarker templates (
FreeMarker Template Engine Overview
) for codegeneration, in combination with JavaScript. A FreeMarker template requires a context with values for the FreeMarker variables it contains and the FreeMarker engine to generate a file. At least one Application Factory script must exist to perform file generation from a template.

You can create template/script pairs using the **Package Explorer** context action Application Factory | Create Script from File, and also from the Recipe Editor. A template can be created from an existing file by replacing strings with template variables of the format ${} where the variable is within the brackets. Application Factory can generate some of these automatically for certain file types. Additionally you can use programming techniques within a template to define objects and invoke methods on those objects in order to create content plus define directives for conditional execution and looping.

To create a template/script pairing from the Package Explorer:

1. From the **Package Explorer**, right-click on the name of a file you want to use as a template.
2. Select **Application Factory** ▶ **Create Script from File**
3. From the **Package Explorer**, right-click on the name of a project you want to use as a template.
4. Select **Application Factory** ▶ **Create Script Recipe from Project**.

Note: You can also create a template/script pairing directly from a **Script Recipe for Application Factory**.

To create a template/script pairing using a Script Recipe:

1. Create a script recipe by selecting **File** ▶ **New** ▶ **Other** ▶ **Application Factory** ▶ **Script Recipe for Application Factory**. **Do not choose a project unless you want to create a script that creates entire projects.**
2. The **Recipe Editor** opens. If no project was selected in the wizard, it displays the **Add Task** dialog. If a project was selected in the wizard, it displays a dialog to selected files.
3. Populate the tasks by dragging and dropping required files for automatic code generation. The **Recipe Editor** makes copies of these files for template resources.
4. Right-click on the task and select **Create Script for Task**. This creates a FreeMarker template for each resource file and displays a dialog to help you customize the task script to generate required file/project.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Creating Scripts

There are multiple ways to create an Application Factory script. Any script creation method requires that there be an Application Factory project in your workspace first (see appropriate subtask below). You can create a script by:

- Using the **Script for Application Factory** wizard—creates a script with the proper metadata and optionally helps you generate the code for an user interface.
- Using a Script Recipe for Application Factory and the **Recipe Editor** —this editor helps you generate and customize both FreeMarker templates and Application Factory scripts.
- Right-clicking on a file name and selecting the Application Factory ▸ Create Script from File option—this action automatically creates both a FreeMarker template for the selected file and a script (which shows an user interface). That input is used to generate a new file from the template. The two files generated by the action are stored in the Application Factory project under the Scripts ▸ Global folder.
- Right-clicking on a project and selecting Application Factory ▸ Create Script Recipe from Project option.

To create an Application Factory project in your workspace:

1 Select the menu path File ▸ New ▸ Project or Other ▸ Application Factory ▸ Application Factory Project. Click **Next**.

2 Specify a name for the Application Factory project or accept the default name.

3 If you want to open the skeleton readme or cheat sheet file, ensure that the appropriate checkbox is marked.

4 Click **Finish**.

To create a script using the Script for Application Factory script wizard:

1 With an Application Factory project open and active in your workspace and perspective set to Application Factory Producer, select the menu path File ▸ New ▸ Script for Application Factory . Click **Next**.

Note: You can also open the **Script for Application Factory** wizard by selecting the menu path File ▸ New ▸ Other ▸ Application Factory ▸ Script for Application Factory

2 The **Script for Application Factory: Create Application Factory Script File** wizard. Choose or create a parent folder under the Scripts folder within the Application Factory project on the first wizard page.

Note: The **Script Preview** pane appears in all pages of the **Script for Application Factory** wizard. It previews the current state of theApplication Factory script you are creating. Select or deselect the **Generate code to report input values** , as desired. Click **Test Script** button at any time to test your script in its current state.

3 Enter a name for your script file in the **File name** area and click **Next**.

4 The **Script for Application Factory: Define your APIs and Other Metadata** page appears with the author name and description completed. All currently installed DOMs are listed in the **DOMs to load** area. Check or uncheck the DOMs to add to your script and click **Next**. You can also select or deselect all items in the list or click **Install New DOM** to add a DOM to this list. Changes you make on subsequent wizard pages automatically select DOMs as needed if they are not yet selected.

5 The **Script for Application Factory: Add a User Interface to your Script** page appears with a default value in the **Name** , **Title**, and **Description** fields for your script. These are only used if you add UI elements to the script on this page.

Note: The **Script Preview** pane appears in all pages of the **Script for Application Factory** wizard. It previews the current state of theApplication Factory script you are creating. Select or deselect the **Generate code to report input values** , as desired. Click **Test Script** button at any time to test your script in its current state.

6 The **UI elements** field lists all elements defined. To add variable UI elements, click **New**. This opens the **New UI Element** dialog, containing:

- Workspace resources, including project, package, class, project file and EJB entity elements.
- File system resources, including file and directory elements
- Data entry and selection, including text, check box, combination box, list box, and check tree elements
- Template resources (template variables extracted from a FreeMarker template that you select)

Click **OK** when the UI elements are selected. The code for the UI elements selected shows in your **Script Preview** window.

When selecting each variable in the UI elements list, different properties with default values for that UI element type appear below. Each field contains a default value. The default values can be used or changed, as desired.

Note: The Scope property appears on those UI elements that need a project reference in order to work (for example: the Package element). By default UI elements use the project reference last selected using any Project element. This works if you have a single Project element, however you can override this behavior by selecting a particular Project element when there are more than one.

Click **Next** when done.

7 The **Script for Application Factory: Add Code to Change Workspace Files** dialog appears. This dialog page allows you to generate code that can create and delete files, or change text in existing files.

Note: The **Script Preview** pane appears in all pages of the **Script for Application Factory** wizard. It previews the current state of theApplication Factory script you are creating. Select or deselect the **Generate code to report input values** , as desired. Click **Test Script** button at any time to test your script in its current state.

8 The following fields can be specified on this dialog page:

- Click on **New** to add a new code snippet.
- Choose the type of operation (create, delete, or modify) from the dropdown menu in the **Operation type** field. .
- Enter a description in the **File change description** field.
- If you want to apply tags, click on **. . .** in the **Tags to apply** field. This allows you to select or deselect tags that you want to apply to this file when the change made by this snippet is committed.
- In the **Select project** area, specify or browse to the **Project workspace name** or select the **Project reference variable** to use this value instead as the project identifier.
- In the **Select project file** area, specify or browse to the **Project-relative path** or select the **Project file reference variable** to use this value instead as the project file identifier.
- The **Configure insert/replace indicator** specifies an expression to locate the insertion point and defines operation relative to it (insert before, insert after, insert on next line, and replace the matched text).
- The **Select source** area specifies whether the source of the input text is:
- a string (that you insert in the text area). If it is a string, you can check **Treat string as template**.
- a template file (for which you can specify a name or browse to for selection).

9 Click **Finish** to complete the script wizard and create your script file. The generated script does not include the code to “report input values.”

To create a script using the Recipe Editor:

1 Create a **Script Recipe for Application Factory** by opening File ▶ New ▶ Other ▶ Application Factory ▶ Script Recipe for Application Factory. Click **Next**..

2 The **Script Recipe for Application Factory** dialog opens.

3 Enter a name for the script in the **Name** field. Click **Finish**.

4 The **Script Recipe** editor open and shows the **Add Task** dialog. Select the task type.

5 You can add additional tasks using the **Add Task** icon on the toolbar or create a project task by dragging and dropping a project from **Package Explorer** to the **Recipe Editor**.

6 See the following topics for further details on using the **Recipe Editor** for script creation.

- Creating Scripts from Files Using a Script Recipe
- Creating Scripts from Projects Using a Script Recipe
- Creating Scripts from VCS Mining Using a Script Recipe

To create a script from a file:

1 Either at the file name, or from within a file, right-click and select the Application Factory ▶ Create Script from File option.

2 A FreeMarker template for the selected file and a script (which shows an user interface) are generated. . The two files generated by the action are stored in the Application Factory project under the Scripts folder.

See the topic Creating Scripts from Files Using a Script Recipe for further details.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts

Related Reference

Application Factory Dialogs and Preferences Reference
FreeMarker Template Engine Overview
Eclipse Monkey Help Front Page

Creating Scripts from Files Using a Script Recipe

You first must have an open Application Factory project in your workspace to create a script recipe and open the **Recipe Editor**. The **Recipe Editor** organizes itself by tasks. Templates and scripts created with the **Recipe Editor** are stored in the **Scripts** folder of the Application Factory project.

To create a script recipe and open the Recipe Editor:

1. You must have an open Application Factory project in your workspace.
2. Select **File** ▸ **New** ▸ **Other** ▸ **Application Factory** ▸ **Script Recipe for Application Factory**.
3. Complete the **Script Recipe for Application Factory** wizard. A script recipe is created and the **Recipe Editor** opens.

To add a task to the script recipe:

1. You may add a task when the script recipe is created by checking the **Add a project task when open the new recipe** option in the **Script Recipe for Application Factory** wizard. After the script recipe has been created, you can create tasks in the **Recipe Editor** for code generation using the **Add Task** toolbar icon.
2. You can populate the tasks by dragging and dropping required files for automatic code generation using the following methods:

- Right-click on any file in the workspace and select the context menu option **Application Factory** ▸ **Add File to Task**.
- Drag any file listed from the **Package Explorer** and drop it onto a task.

To create a script from an existing file:

1. Right-click on a task and choose **Create Script for Task**. This action creates templates for each resource under the task and then opens a dialog box where you can generate UI and snippets that are added to the generated scripts. Any FreeMarker variables in the templates contribute to this UI. You can customize how this is presented by introducing your own variables and regenerating the script. Setting changes made in the dialog box are saved and you do not lose them by regenerating the script.
2. The template and script appear beneath the task or file. The FreeMarker template may contain generated template variables. The generated script also contains UI to prompt for values needed to populate the template.
3. Selecting a file opens it for viewing or editing in the lower pane of the **Recipe Editor** script executes the script.
4. The lower pane has a **Test** icon in the toolbar. Clicking the **Test** button for a template verifies that it can be parsed successfully. Clicking the **Test** button for a script executes the script.

Note: A copy of each non-binary file added to the **Script Recipe** editor is used as a FreeMarker template for you to customize. If that template has strings in it which the FreeMarker engine thinks are wrong, an exception is thrown either when the template is parsed (look in the Error Log view) or at runtime (dialog displayed) depending on the problem type. You can modify the resource file to remove the conflict and regenerate the template as a fix. Another solution is to right-click on the resource file (not the template) and use the **Change File Type** context menu to flag it as a binary file. Binary files are not made into templates. They are copied directly to the target directory by the generated script.

To reopen an existing recipe:

1 Click on the **Package Explorer** view to activate it.

2 Double-click on an existing recipe in the **Package Explorer** view. It is under the Scripts Recipes folder of the Application Factory project.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Creating Scripts from Projects Using a Script Recipe

You first must have an openApplication Factory project in your workspace to create a script recipe and open the **Recipe Editor**. The **Recipe Editor** organizes itself by tasks. Templates and scripts created with the **Recipe Editor** are stored in theScripts ▸ Recipes folder of the workspace Application Factory project.

To create a script recipe and open the Recipe Editor:

1 You must have an open Application Factory project in your workspace.

2 Select File ▸ New ▸ Other ▸ Application Factory ▸ Script Recipe for Application Factory.

3 Complete the **Script Recipe for Application Factory** wizard. A script recipe is created and the **Recipe Editor** opens.

To add a task to the script recipe:

1 You may add a task when the script recipe is created by checking the **Add a project task when open the new recipe** option in the **Script Recipe for Application Factory** wizard. After the script recipe has been created, you can create tasks in the**Recipe Editor** for code generation using the **Add Task** toolbar icon.

2 You can also populate the tasks by dragging and dropping required files for automatic code generation using the following methods. All files are added by making a copy of the file in order to customize it as part of the template creation process without affecting the original file:

- Right-click on any file in the workspace and select the context menu option Application Factory ▸ Add File to Task.
- Drag any file listed from the **Package Explorer** and drop it onto a task.

To create a script from a project:

1 Drag and drop any existing projects from the workspace to the **Recipe Editor**.

2 A dialog appears that prompts the user to select project files that should be included in task creation. Select required resources and click **OK**. The selected files are copied into the **Recipe Editor** and listed under the project task.

3 Right-click on a task and choose **Create Script for Task**.

4 The templates and script appear beneath the task or file. A FreeMarker template may contain generated template variables or those you add yourself. The generated script may also contains UI to prompt for values needed to populate the template

5 Selecting a task resource file opens it for viewing or editing in the lower pane of the **Recipe Editor**.

6 The lower pane has a **Test** icon in the toolbar. Clicking the **Test** button for a template verifies that it can be parsed successfully. Clicking the **Test** button for a script executes the script.

To reopen an existing recipe:

1 Click on the **Package Explorer** view and highlight the existing recipe.

2 Double-click on an existing recipe in the **Package Explorer** view under the Script ▸ Recipes folder of the Application Factory project. The recipe opens in the **Recipe Editor** view.

Note: A copy of each non-binary file added to the **Script Recipe** editor is used as a FreeMarker template for you to customize. If that template has strings in it which the FreeMarker engine thinks are wrong, an exception is thrown either when the template is parsed (look in the Error Log view) or at runtime (dialog displayed) depending on the problem type. You can modify the resource file to remove the conflict and regenerate the template as a fix. Another solution is to right-click on the resource file (not the template) and use the **Change File Type** context menu to flag it as a binary file. Binary files are not made into templates. They are copied directly to the target directory by the generated script.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Creating Scripts from VCS Mining Using a Script Recipe

Data mining of information in your version control system can be done using the **Commit History** view in Application Factory. The **Commit History** view pulls in the VCS information from projects in the workspace that are under source control (JBuilder 2008 only has support for Subversion). It then aggregates them by date. The resulting data can be filtered and searched by date, author, and commit comment text.

You first must have an open Application Factory project in your workspace to create a script recipe and open the **Recipe Editor**. The **Recipe Editor** organizes itself by tasks. Templates and scripts created with the **Recipe Editor** are stored in theScripts ▸ Recipe folder of the Application Factory project.

To create a script recipe and open the Recipe Editor:

1 You must have an open Application Factory project in your workspace.

2 Select File ▸ New ▸ Other ▸ Application Factory ▸ Script Recipe for Application Factory.

3 Complete the **Script Recipe for Application Factory** wizard. A script recipe is created and the **Recipe Editor** opens.

To add a task to the script recipe:

1 You may add a task when the script recipe is created by checking the **Add a project task when open the new recipe** option in the **Script Recipe for Application Factory** wizard. After the script recipe has been created, you can create tasks in the**Recipe Editor** for code generation using the **Add Task** toolbar icon.

2 You can also populate the tasks by dragging and dropping required files for automatic code generation using the following methods:

- Right-click on any file in the workspace and select the context menu option Application Factory ▸ Add File to Task.
- Drag any file listed from the **Package Explorer** and drop it onto a task.

To create a script through VCS data mining:

1 The script recipe should have been created and the **Recipe Editor** opened.

2 Select the **Commit History** view by either of the following methods:

- WIndow ▸ Show View ▸ Other ▸ Application Factory ▸ Commit History
- WIndow ▸ Show View ▸ Other and type `Commit History` in the **type filter text** field.
- Use **Explore Workspace Respository** icon in the upper right pane toolbar of **Script Recipe Editor**.

3 The **Commit History** view opens. This view pulls in the VCS information from projects in the workspace that are under source control. It then aggregates them by date. The resulting data can be filtered and searched by date, author, and commit comment text.

Tip: The upper pane of the **Commit History** view shows of the all the commits. When a commit is selected, the lower-left pane shows all the files that were changed in that commit. When a file is selected, the lower-right pane shows the entire commit history of that file with the revision for the selected commit highlighted.

4 Drag a file revision (or use the context menu) and drop it on a selected task in the **Recipe Editor** to create code snippets file. The snippets contains the changes that were made to produce the selected revision from the previous version. The base revision used to determine the snippets depends on what revision was dropped on

the task. You need to use the **Create Script for Task** dialog and navigate to its second page to complete configuration by identifying which files(s) are to be modified by inserting the snippets.

Note: If a revision was selected from the lower-left pane, the code snippets file compares that version of the file with the version prior to that revision. You can also multiple-select any two revisions in the lower-right pane and compare them.

5 Select **Create Script for Task** to generate a script with UI that allows you to select a file and insert one or more snippets into the selected file. You will need to either customize the script to provide the proper insertion point, or use the **Script Learn/Resolve/Commit** view to adjust the insertion point prior to committing the changes through the **Commit Changes** toolbar icon in this view.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Debugging a Script

When a script fails, a dialog showing an error message and line number is shown. The dialog optionally allows for the file to be opened for edit. The file is opened for edit at the line causing the error.

To debug a script:

1 In the **Scripts – Application Factory** view, execute a script by:

- Double-click on a script name.
- Right-click on a script name and select the **Execute Script** option.

2 When a script fails, click on the **Open Script File** button in the error dialog to open the script at the line that caused the failure.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts

Related Reference

Application Factory Dialogs and Preferences Reference

Editing a Script

Application Factory scripts can be opened for editing from the **Scripts – Application Factory** view in the IDE.

To edit a script:

1. Open the **Scripts—Application Factory** view by selecting either of the following paths:
   - **Window ▶ Show View ▶ Other ▶ Application Factory ▶ Scripts—Application Factory**
   - **Window ▶ Show View ▶ Other** and type `Scripts-Application Factory` in the **type filter text** field.

   .
2. Right-click on a script name and select **Edit** to open the script in the JavaScript editor.

You can also open the script you want to edit from the **Package Explorer** from the Scripts folder within the Application Factory project.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Filtering in Scripts—Application Factory View

This topic describes how to filter the scripts that appear in the **Scripts—Application Factory** view. By default, the **Scripts — Application Factory** view in the IDE displays only runnable scripts in the Application Factory project.

To display all scripts in the Scripts—Application Factory view for the Application Factory project:

1 Open the **Scripts—Application Factory** view by selecting either of the following paths:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Scripts—Application Factory**
- **Window** ▸ **Show View** ▸ **Other** and enter `Scripts — Application Factory` in the **type filter text** field.

2 Click on the dropdown menu (down arrow) in the toolbar and uncheck the option **Show Only Runnable Scripts**

The Scripts view provides a text field at the top of the pane which initially says “type filter text.” You can enter a string in this field that causes it to only show folders/files that contain that string.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts

Related Reference

Application Factory Dialogs and Preferences Reference

Focusing on a Script Run

The **Scripts—Application Factory** view has a toolbar button to focus IDE views (the**Package Explorer**, **Navigator** and **Scripts—Application Factory** views) on the currently select script run.

To focus on a script run from the Scripts —Application Factory view:

1. Open the **Archeology** view for the desired script by right-clicking on the script in the **Scripts - Application Factory** view.
2. Select **Open Script Run Archeology** option.
3. Select a script run from the list.
4. Click the **Focus on Script Run selected in Archeology View** icon in the **Scripts—Application Factory** view toolbar . This action focuses the **Package Explorer, Navigator** and **Scripts—Application Factory** views on the resources that were affected by the selected script run.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Resolving Code Snippets from a Script Run

When a script is run that uses CodeGear Application Factory DOM to create/delete/modify/ tag files, change snippets are persisted under the Application Factory project. If there are any problems with the script run, the **Script Learn/ Resolve/Commit** view is displayed at the end of the script run. If there are no problems with the script run, a dialog appears asking if the user wants to see a detailed list of changes. If the user chooses this option, the **Script Learn/ Resolve/Commit** view is displayed. This view allows users to see all the changes from a script run, fix any problems, and either commit or abandon changes, either entirely or on an individual file basis.

The **Script Learn/Resolve/Commit** view contains a list of files that were added/modified/deleted in a script run. Double-clicking or using the right-click context menu allows opening a view that compares the original file and what the file would look like if changes to it are committed.

When a file is selected, the bottom pane shows all the scripts that changed the file (with highlighting to show lines that have the changes). The pane on the right shows both the descriptive text and the change snippets.

To resolve missing resources:

1. Right-click on a file that was not found.
2. Select **Resolve Conflict** and then select the correct location of the file.

To change the location of a code snippet:

1. Right-click on any file that is to be modified in the change list using insert-type snippets and select the **Change Insert Location**.
2. The **Adjust Insert Location** dialog is displayed with a green arrow pointing to the location where the snippet will be inserted. A number indicating the number of snippets to be inserted at the same location in the file (if greater than 1) appears to the right of the arrow. Click on the green bar to the right to navigate between code snippet insertion points. Drag and move the green arrow to change the location of the insertion point for a code snippet.

To change a script:

1. Click on the **Open File in Separate Editor** toolbar icon in the **Script Learn/Resolve/Commit** view. Modify the script as required in the **Script Editor**.

To accept all snippet changes:

1. Click on the **Commit Changes** toolbar icon.

To discard all snippet changes:

1. Double-click on a script name in the **Scripts – Application Factory** view to execute a script.
2. Click on the **Show Changes** button to bring up the **Script Learn/Resolve/Commit** view.
3. Click on the **Clear All Unresolved Changes** toolbar icon.

To discard snippet changes for a file:

1. Right-click on any file that is to be modified in the list and select **Clear Changes for File**.

To open the Compare Editor for a file:

1 Right-click on any file that is to be modified in the list and select **Open Compare Editor**.

2 Double-click the file.

To filter the file change list:

1 Click on the dropdown in the toolbar and select appropriate filtering action.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Running a Script

Application Factoryscripts can be run from the **Script – Application Factory** view in the IDE.

To run a script:

1. Open the **Scripts—Application Factory** view by selecting either of the following paths:
   - **Window** ▶ **Show View** ▶ **Other** ▶ **Application Factory** ▶ **Scripts — Application Factory**
   - **Window** ▶ **Show View** ▶ **Other** and enter `Scripts — Application Factory` in the **type filter text** field.

2. Double-click on a script name to execute that script or right-click on a script name and select the **Execute Script** option.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Using Tags

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes how to use tags in Application Factory development.

In This Section

Adding a Parent-Child Relationship for Tags
Describes how to add a parent-child relationship between tags in Application Factory.

Creating a Tag
Describes how to create a new tag in Application Factory.

Creating a Tag from the Application Diagram
Describes how to create a tag from the **Application Diagram** of Application Factory.

Exposing a Resource in the Application Diagram
Describes how to expose a link between resources in the **Application Diagram**.

Exposing a Tag in the Application Diagram
Describes how to expose a tag in the **Application Diagram**.

Focusing on a Tag
Describes how to focus on a tag.

Focusing on Untagged Resources
Describes how to focus on untagged resources.

Opening the Application Diagram
Describes how to open the application diagram.

Tagging a Resource
Describes how to tag a resource in Application Factory.

Viewing of Tags
Describes how to view tags in Application Factory.

Adding a Parent-Child Relationship for Tags

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes how to create parent-child relationships between tags in Application Factory.

To add a child relationship to a tag:

1 Open the **Tags** view by either of the following methods:

- **Window ▸ Show View ▸ Other ▸ Application Factory ▸ Tags**
- **Window ▸ Show View ▸ Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 To add a child tag to a selected tag, right-click on a tag and select **Tag Children** option.

3 Click on the tag that you want to make a child of the selected tag.

4 The tag blinks to indicate that the resource has been added as a child to the parent tag.

To add a parent relationship to a tag:

1 Open the **Tags** view by either of the following methods:

- **Window ▸ Show View ▸ Other ▸ Application Factory ▸ Tags**
- **Window ▸ Show View ▸ Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 To add a parent tag to a selected tag, right-click on a tag and select **Tag Parent** option.

3 Click on the tag that you want to make a parent of the selected tag.

4 The tag blinks to indicate that the resource has been added as a parent to the child tag.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Tags
Adding Notes to Tags
Adding a Parent-Child Relationship for Tags
Exposing a Resource in the Application Diagram
Exposing a Tag in the Application Diagram
Filtering on Tag Notes in the Application Diagram
Focusing on a Tag
Focusing on Untagged Resources
Opening the Application Diagram
Tagging a Resource
Viewing of Tags

Creating a Tag

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes how to create a new tag in Application Factory.

To create a new tag in Application Factory:

1 If not open in your workbench, open the **Tags** view by either of the following methods:

- **Window** ▶ **Show View** ▶ **Other** ▶ **Application Factory** ▶ **Tags**
- **Window** ▶ **Show View** ▶ **Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 Click on the **Create Tag** toolbar icon on the right side of the **Tags** view.

3 The **Create Tag** dialog appears.

4 Enter a tag name in the **Tag name** field.

5 Enter a tag description in the **Description** field.

6 If you want to mark this a personal tag in the application module (a tag that is not shared with the team), check the **Personal tag, not shared with the team** box.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Tags
Adding Notes to Tags
Adding a Parent-Child Relationship for Tags
Exposing a Resource in the Application Diagram
Exposing a Tag in the Application Diagram
Filtering on Tag Notes in the Application Diagram
Focusing on a Tag
Focusing on Untagged Resources
Opening the Application Diagram
Tagging a Resource
Viewing of Tags

Creating a Tag from the Application Diagram

This section describes how to create a tag from the **Application Diagram** of Application Factory.

To create a tag from the Application Diagram:

1 Open the application diagram:

- Switch to the **Application Factory Modeling** perspective. If the application diagram is not already open, expand the Application Factory project and double-click on the application diagram.
- Right-click on the Application Factory project in the **Package Explorer** or **Navigator** views and select **Open Application Diagram**.

2 Click on the tag element in the palette and drop it on the diagram to create a new public tag. Click on the tag to rename the tag. The tag automatically appears in the **Tags** view.

3 Click on the Personal tag element in the palette and drop it on the diagram to create a new personal tag. Click on the tag to rename the tag. The tag automatically appears in the **Tags** view. A personal tag is not exported when publishing (exporting) an application module.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Tags
Adding Notes to Tags
Adding a Parent-Child Relationship for Tags
Exposing a Resource in the Application Diagram
Exposing a Tag in the Application Diagram
Filtering on Tag Notes in the Application Diagram
Focusing on a Tag
Focusing on Untagged Resources
Opening the Application Diagram
Tagging a Resource
Viewing of Tags

Exposing a Resource in the Application Diagram

The **Application Diagram** describes application architecture and functionality. Tags in the **Application Diagram** can be related to other tags. This topic describes how to expose links between resources in the **Application Diagram** .

To expose a resource link in the Application Diagram:

1 If not open in your workbench, open the **Tags** view by either of the following methods:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Tags**
- **Window** ▸ **Show View** ▸ **Other**. Type `Tags` in the TYPE FILTER TEXT area.

2 Select the resource(s) you want to expose in the application diagram in either the **Package Explorer** or **Navigator** views.

3 Enter **Link Mode** using the toolbar dropdown menu in the **Tags** view.

4 Right-click on the tag associated with the resource and select **Show Link in Application Diagram**. This displays the selected resource(s) associated with the tag, along with the tag in the application diagram.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Exposing a Tag in the Application Diagram

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes the procedure for exposing a tag in the **Application Diagram**.

To expose a tag in the Application Diagram:

1 Open the **Tags** view by either of the following methods:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Tags**
- **Window** ▸ **Show View** ▸ **Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 Enter **Browse Mode** using the toolbar dropdown menu in the **Tags** view.

3 Right-click on the tag you want to add to the diagram and select **Show Tag as Application Diagram Package**.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Focusing on a Tag

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes how to focus on a tag in Application Factory.

To focus on a tag:

1 Open the **Tags** view by either of the following paths:

- **Window ▸ Show View ▸ Other ▸ Application Factory ▸ Tags**
- **Window ▸ Show View ▸ Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 The default mode for the **Tags** view is the browse mode. You can also enter the browse mode by clicking on the **Browse all tags by their weight and show associated resources** toolbar icon.

3 Click on the tag on which to focus.

4 Click on the **Apply selected Tag focus in main views** toolbar button.

The IDE views (Package Explorer, Navigator, Scripts) now focus on resources associated with the selected tag

5 Click on another tag to switch focus.

6 Click on toolbar button again (now labeled **Remove selected Tag focus from main views**) to remove focus.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Tags
Adding Notes to Tags
Adding a Parent-Child Relationship for Tags
Exposing a Resource in the Application Diagram
Exposing a Tag in the Application Diagram
Filtering on Tag Notes in the Application Diagram
Focusing on a Tag
Focusing on Untagged Resources
Opening the Application Diagram
Tagging a Resource
Viewing of Tags

Focusing on Untagged Resources

The topic describes how to focus on untagged resources.

To focus on untagged resources:

1 Open the **Tags** view by either of the following paths:

- **Window ▸ Show View ▸ Other ▸ Application Factory ▸ Tags**
- **Window ▸ Show View ▸ Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 The default mode for the **Tags** view is the browse mode. You can also enter the browse mode by clicking on the **Browse all tags by their weight and show associated resources** toolbar icon.

3 Click the **Focus IDE on untagged resources** toolbar button to focus on resources without tags.

4 The IDE views (Package Explorer, Navigator, Scripts) now focus on all untagged resources

5 Click on another tag to switch focus.

6 Click on toolbar button again (now labeled **Remove IDE focus from untagged resources**) to remove focus.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Tags
Adding Notes to Tags
Adding a Parent-Child Relationship for Tags
Exposing a Resource in the Application Diagram
Exposing a Tag in the Application Diagram
Filtering on Tag Notes in the Application Diagram
Focusing on a Tag
Focusing on Untagged Resources
Opening the Application Diagram
Tagging a Resource
Viewing of Tags

Opening the Application Diagram

The **Application Diagram** shows a representational model of your application. The **Application Diagram** is created from tags that are marked as **Application Diagram** candidates.

To open the Application Diagram from Package Explorer or Navigator Views:

1. Right-click on the Application Factory project in the **Package Explorer** or **Navigator** views.
2. Select the Open Application Diagram option.
3. The Application Factory modeling diagram opens in the workbench.

To open the Application Diagram through the Application Factory Modeling Perspective:

1. Switch to the **Application Factory Modeling** perspective by either of the following methods:
   - **Window** ▶ **Open Perspective** ▶ **Application Factory Modeling**
   - Click on the **Open Perspective** icon on the top-right of the workbench and select **Application FactoryModeling**.
2. Expand the Application Factory project tree in the **Model Package Explorer**.
3. Double-click on the Application Diagram in the tree.

To open the Application Diagram for a selected tag:

1. If not open in your workspace, open the **Tags** view by either of the following methods:
   - **Window** ▶ **Show View** ▶ **Other** ▶ **Application Factory** ▶ **Tags**
   - **Window** ▶ **Show View** ▶ **Other**. Type `Tags` in the **type filter text** area.
2. Right-click on a tag in the **Tags** view.
3. Select the **Select in Application Diagram** option.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Working with Application Diagrams
Adding Notes to Tags
Exposing a Resource in the Application Diagram
Exposing a Resource in the Application Diagram
Opening the Tags View in the Application Diagram
Opening the Application Diagram
Focusing on a Tag
Filtering on Tag Notes in the Application Diagram

Tagging a Resource

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes how to tag a resource in Application Factory.

To link a resource to a tag:

1 Open the **Tags** view by either of the following methods:

- **Window** ▶ **Show View** ▶ **Other** ▶ **Application Factory** ▶ **Tags**
- **Window** ▶ **Show View** ▶ **Other**. Type `Tags` in the TYPE FILTER TEXT area.

2 Select a file in the **Package Explorer** or **Navigator** views.

3 Switch to **Link Mode** using the toolbar dropdown menu in the **Tags** view.

4 Click on a tag in the **Tags** view. The tag blinks to indicate that the resource has been linked.

To remove a resource link from a tag:

1 Open the **Tags** view by either of the following methods:

- **Window** ▶ **Show View** ▶ **Other** ▶ **Application Factory** ▶ **Tags**
- **Window** ▶ **Show View** ▶ **Other**. Type `Tags` in the TYPE FILTER TEXT area.

2 Select a linked resource in the **Package Explorer** or **Navigator** views.

3 Switch to **Link Mode** using the toolbar dropdown menu in the **Tags** view.

4 Click on the linked resource in the **Package Explorer** or **Navigator** views.

5 Click on the associated tag in the **Tags** view to remove the link.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Adding Notes to Tags
Adding a Parent-Child Relationship for Tags
Exposing a Resource in the Application Diagram
Exposing a Tag in the Application Diagram
Filtering on Tag Notes in the Application Diagram
Focusing on a Tag
Focusing on Untagged Resources
Opening the Application Diagram
Tagging a Resource
Viewing of Tags

Viewing of Tags

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes how to view tags in Application Factory.

To use the Browse Mode for viewing tags:

1 If not open in the workbench, open the **Tags** view by either of the following methods:

- **Window ▸ Show View ▸ Other ▸ Application Factory ▸ Tags**
- **Window ▸ Show View ▸ Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 Enter **Browse Mode** using the toolbar dropdown menu in the **Tags** view . This is the default mode when in the **Tags** view.

To use the Parent or Child Mode to view tags:

1 If not open in the workbench, open the **Tags** view by either of the following methods:

- **Window ▸ Show View ▸ Other ▸ Application Factory ▸ Tags**
- **Window ▸ Show View ▸ Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 Enter **Parent Mode** or **Child Mode** using the toolbar dropdown menu in the **Tags** view .

To use the Link Mode to view tags:

1 If not open in the workbench, open the **Tags** view by either of the following methods:

- **Window ▸ Show View ▸ Other ▸ Application Factory ▸ Tags**
- **Window ▸ Show View ▸ Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 Enter **Link Mode** using the toolbar dropdown menu in the **Tags** view .

To use the Cloud mode for tags:

1 If not open in the workbench, open the **Tags** view by either of the following methods:

- **Window ▸ Show View ▸ Other ▸ Application Factory ▸ Tags**
- **Window ▸ Show View ▸ Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 Enter **Cloud Mode** using the toolbar dropdown menu in the **Tags** view .

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Tags
Adding Notes to Tags
Adding a Parent-Child Relationship for Tags
Exposing a Resource in the Application Diagram
Exposing a Tag in the Application Diagram
Filtering on Tag Notes in the Application Diagram
Focusing on a Tag
Focusing on Untagged Resources
Opening the Application Diagram
Tagging a Resource
Viewing of Tags

Working with Application Diagrams

This section contains links to procedural topics for application diagrams.

The **Application Diagram** describes application architecture and functionality. The diagram provides a high level summary of the application. It can include application architecture, employed technologies, third-party dependencies, and so forth. The diagram is useful as a tool to describe how the internals of the application work to a new user.

The diagram surfaces information from the tags and resources marked as diagram candidates. The diagram displays description and notes for tags and represents parent-child relationships and related tags. The **Application Diagram** is stored in the Application Factory project.

In This Section

Adding Notes to Tags

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes how to create a note for a tag and associate it to a tag in an Application Factory application diagram.

To add a note to a Tag:

1 If not open in your workbench, open the **Tags** view by either of the following methods:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Tags**
- **Window** ▸ **Show View** ▸ **Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 Right-click on a tag in the **Tags** view and select the **Select on Application Diagram**.

3 Click on the **Note** icon in the diagram palette and drop onto the diagram.

4 Click on the note in the diagram to edit the content.

5 Click on the **Link Note** icon in the diagram palette, click on the note and drag and release onto the tag that you wish to associate the note with.

6 Click on the note in the diagram.

7 Drag and release the note onto the tag with which you want the note to be associated.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Tags
Adding Notes to Tags
Adding a Parent-Child Relationship for Tags
Exposing a Resource in the Application Diagram
Exposing a Tag in the Application Diagram
Filtering on Tag Notes in the Application Diagram
Focusing on a Tag
Focusing on Untagged Resources
Opening the Application Diagram
Tagging a Resource
Viewing of Tags

Exposing a Resource in the Application Diagram

The **Application Diagram** describes application architecture and functionality. Tags in the **Application Diagram** can be related to other tags. This topic describes how to expose links between resources in the **Application Diagram** .

To expose a resource link in the Application Diagram:

1 If not open in your workbench, open the **Tags** view by either of the following methods:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Tags**
- **Window** ▸ **Show View** ▸ **Other**. Type `Tags` in the TYPE FILTER TEXT area.

2 Select the resource(s) you want to expose in the application diagram in either the **Package Explorer** or **Navigator** views.

3 Enter **Link Mode** using the toolbar dropdown menu in the **Tags** view.

4 Right-click on the tag associated with the resource and select **Show Link in Application Diagram**. This displays the selected resource(s) associated with the tag, along with the tag in the application diagram.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Exposing a Tag in the Application Diagram

A tag is a keyword associated with a piece of information. Tags are used to group related resources. This section describes the procedure for exposing a tag in the **Application Diagram**.

To expose a tag in the Application Diagram:

1 Open the **Tags** view by either of the following methods:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Tags**
- **Window** ▸ **Show View** ▸ **Other**. Type `Tag` in the TYPE FILTER TEXT area.

2 Enter **Browse Mode** using the toolbar dropdown menu in the **Tags** view.

3 Right-click on the tag you want to add to the diagram and select **Show Tag as Application Diagram Package**.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Filtering on Tag Notes in the Application Diagram

This topic describes how to filter using tag notes in the **Application Diagram** of Application Factory.

To filter by tag notes in the application diagram:

1. Click **Hide/Show Elements on Diagram** toolbar button.
2. Select the note you wish to hide.
3. Click **OK**.

To show tag notes, repeat the steps above, but select the note you wish to show.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Opening the Application Diagram

The **Application Diagram** shows a representational model of your application. The **Application Diagram** is created from tags that are marked as **Application Diagram** candidates.

To open the Application Diagram from Package Explorer or Navigator Views:

1 Right-click on the Application Factory project in the **Package Explorer** or **Navigator** views.
2 Select the **Open Application Diagram** option.
3 The Application Factory modeling diagram opens in the workbench.

To open the Application Diagram through the Application Factory Modeling Perspective:

1 Switch to the **Application Factory Modeling** perspective by either of the following methods:

- **Window** ▸ **Open Perspective** ▸ **Application Factory Modeling**
- Click on the **Open Perspective** icon on the top-right of the workbench and select **Application FactoryModeling**.

2 Expand the Application Factory project tree in the **Model Package Explorer**.
3 Double-click on the Application Diagram in the tree.

To open the Application Diagram for a selected tag:

1 If not open in your workspace, open the **Tags** view by either of the following methods:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Tags**
- **Window** ▸ **Show View** ▸ **Other**. Type `Tags` in the **type filter text** area.

2 Right-click on a tag in the **Tags** view.
3 Select the **Select in Application Diagram** option.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Working with Application Diagrams
Adding Notes to Tags
Exposing a Resource in the Application Diagram
Exposing a Resource in the Application Diagram
Opening the Tags View in the Application Diagram
Opening the Application Diagram
Focusing on a Tag
Filtering on Tag Notes in the Application Diagram

Opening the Tags View in the Application Diagram

This topic describes how to open the **Tags** view in the **Application Diagram** of Application Factory.

To open the tags view in the application diagram:

1. Open the **Application Factory Modeling** perspective.
2. Expand the Application Factory project in the **Model Navigator** and double-click on the Application Diagram to open the diagram.
3. Right-click on any tag in the diagram and select the option **Select in Tags view**.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Tags
Working with Application Diagrams

Using Drag and Drop Functionality in the Application Diagram

This topic describes how to use drag and drop functionality in the **Application Diagram** of Application Factory.

To drag and drop tags onto the Application Diagram:

1 Open the **Application Diagram**.

2 Open the **Tags** view.

3 Select a tag in the **Tags** view and drag and drop onto the application diagram to add the tag to the diagram.

To drag and drop resources onto tags in the Application Diagram:

1 Open the **Application Diagram**.

2 Drag and drop a resource from the **Package Explorer** or **Navigator** views onto a tag in the **Application Diagram** to link the resource to the tag.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Concepts

Related Tasks

Using Application Factory
Using Tags
Working with Application Diagrams

Related Reference

Application Factory Dialogs and Preferences Reference

Working with Application Modules

An Application Factory Module is a set of application projects, including the Application Factory project. The Application Factory project contains information (metadata) about the application. This metadata enables an application-driven development model through the Application Factory functionality. An Application Factory Module is stored as a zip file (.mar). The zip file contains the Application Factory project and all application projects.

JBuilder ships with pre-packaged application modules. Users can browse through and install these modules using the **Application Factory Explorer** view in JBuilder. Users can consume and publish application modules using the **Application Factory Explorer** view and the **Application Module Editor**.

In This Section

Browsing Modules in Application Factory Explorer View
Describes how to browse Application Factory modules in the **Application Explorer**.

Consuming Application Modules
Describes how to consume (import) an Application Module.

Creating a Application from the Eclipse Monkey DOM Template Application Module
Describes how to create an application from the Eclipse Monkey DOM template.

Creating an Application Factory Project
Describes how to create an Application Factory project.

Creating and Using Add-on Modules
Describes how to publish (export) and consume (import) an Application Module as an add-on module.

Creating and Using RSS/Atom Feeds
Describes how to create and use RSS/Atom feeds in publishing and consuming an Application Factory module.

Editing an Application Module Cheat Sheet
Describes how to edit an application module cheat sheet.

Editing an Application Module Readme
Describes how to edit a template readme file.

Editing Application Modules
Describes how to edit an Application Module in Application Factory.

Publishing an Application Module
Describes how to publish an Application Module.

Setting an Application Module Search or Export Directory
Describes how to set an Application Modules directory.

Browsing Modules in Application Factory Explorer View

The **Application Factory Explorer** view allows users to easily browse all available application modules. **Application Factory Explorer** view is part of the **Application Factory Repository Exploring** perspective.

Using the **Application Factory Explorer** view, users can filter by application type, frameworks or license used in the application modules. Clicking on an application module link opens the **Application Module Editor** for the selected module. The **Application Module Editor** displays read-only information about the module (screenshots, license, application diagram and tag snapshots).

To open the Application Factory Explorer view:

1 Switch to the **Application Factory Repository Exploring** perspective (the default perspective). The **Application Factory Explorer** view is opened by default in this perspective.

2 Or, specifically open the **Application Factory Explorer** view in your current perspective by selecting one of the following paths:

- **Window** ▶ **Show View** ▶ **Other** ▶ **Application Factory** ▶ **Application Factory Explorer**
- **Window** ▶ **Show View** ▶ **Other** and enter `Application Factory Explorer` in the TYPE FILTER TEXT field.

To filter in the Application Factory Explorer view:

1 You can filter the application listed in the **Application Factory Explorer** view by Add-on Module, Application Kind, Framework, Import Location and License.

2 Select the appropriate filters in the left-side pane to filter the list in the right-side pane of the **Application Factory Explorer**.

To open and view information in the Application Module Editor:

1 To open the **Application Module Editor**, click on the application module link (for example, Eclipse Monkey DOM Plugin, Book Store, Pet Store, E-commerce Application, JSF Data-Aware Application, Spring MVC Data-Aware Application, or Struts 2 Data-Aware Application links) in the **Application Factory Explorer** view.

2 Click the **Preview** tab and click on any image to view screenshots for the application. Each of the larger-sized images can be grabbed and repositioned for different views of the screenshots.

3 Click on the **Diagram** tab to view a snapshot of the Application Diagram for the module.

4 Click on the **Tags** tab to view a snapshot of the tags for the module

5 Click on the **License** tab to view licenses for the module.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Working with Application Modules
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Consuming Application Modules

Application Modules can be imported into a workspace using the **Application Factory Explorer** and the **Application Module Editor**. Note that Application Modules can be imported into a workspace only if the workspace has an existing Application Factory project.

The **Application Module Editor** creates an instance of the Application Module. The **Application Module Editor** can be opened by clicking on an application module in the **Application Factory Explorer**. Opening the **Application Module Editor** for a module imports the Application Factory project and any other projects included in the application module. It also invokes any application creation script defined to perform application configuration for the module.

To open the Application Factory Explorer view:

1 Switch to the **Application Factory Repository Exploring** perspective (the default perspective). The **Application Factory Explorer** view is opened by default in this perspective.

2 Or, specifically open the **Application Factory Explorer** view in your current perspective by selecting one of the following paths:

- **Window** ▸ **Show View** ▸ **Other** ▸ **Application Factory** ▸ **Application Factory Explorer**
- **Window** ▸ **Show View** ▸ **Other** and enter `Application Factory Explorer` in the TYPE FILTER TEXT field.

To create (import) an Application Module:

1 Click on the **Create Application** button in any of the **Application Module Editor** tabs.

2 The Application Factory project extracts and invokes the application creation script (if defined) for the application module.

3 An application creation script can be used to perform any application-specific configuration actions. With the included JSF, Spring MVC, or Struts 2 data-aware applications, a multi-page **New {JSF | Spring MVC | Struts 2} Data-Aware Web Application** wizard opens and is completed to configure these applications. Refer to the dialog reference for these specific data-aware web application wizards for more details.

To consume (import) an Add-on Application Module:

1 Open the add-on module from the **Application Factory Explorer** . You can filter for add-on modules in the left-side pane. See the previous subtopic on filtering for add-on modules.

2 Click on **Add Application** in any of the **Application Module Editor** tabs (Preview, Diagram, Tags, or License).

3 The imported contents of the add-on module are subsumed into the current Application Module in the workspace. Once imported as add-on, modules are not available as a separate module but as files in the current Application Module in the workspace.

4 The consumed add-on module is created in a subdirectory of the existing workspace Application Module. The subdirectory is located under the current Application Module's workspace add-on module directory. It has the add-on module's name as the parent directory name. For example, if you import a module named `Test` as an add-on module to your existing Application Module named `FirstModule`, the add-on module is created under the`FirstModule/Add-on/Test` directory.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Working with Application Modules
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
JSF Application Factory Dialogs Reference
Spring MVC Application Factory Dialogs Reference
Struts 2 Application Factory Dialogs Reference

Creating a Application from the Eclipse Monkey DOM Template Application Module

To import an Eclipse Monkey DOM Template Application from the Application Factory Explorer:

1 Open or switch to the **Application Factory Repository Exploring** perspective (the default perspective) by one of the following paths:

- **Window** ▸ **Open Perspective** ▸ **Other** ▸ **Application Factory Repository Exploring**
- Click on the **Open Perspective** icon and select **Other** **Application Factory Repository Exploring**
- If this perspective has been previously opened, you can switch back to it by clicking the **Application Factory Repository Exploring perspective** icon in the toolbar.

2 Select the template module **Eclipse Monkey DOM Plugin Factory 1.0** in the right-hand pane of the Explorer. This opens this Eclipse Monkey DOM template application in the **Application Preview** pane.

3 Click **Create Application** to create a new application. This overwrites any existing Application Factory project in your workspace. Click **Add Application** to add an existing Eclipse Monkey DOM application.

Note: If you do not have an Application Factory project in your workspace, one is created by this template. if you do have an Application Factory project, iti is replace dunless you use the **Add Application** button.

4 If you have previously created a Eclipse Monkey DOM project in your workspace, you may see an error message asking if you want to recreate it. Click **OK.**

5 An**Import Application Module** dialog wizard appears asking if you want to run the application creation **dom.js** application creation script now or later. Click**Now** to immediately launch the **dom.js** script. If you select **Later**, the script can always be launched by:

- Double-clicking on **dom.js** in the **Script—Application Factory** view.
- Right-clicking on **dom.js** in the **Script—Application Factory** view and selecting **Execute Script**.

6 Executing the **dom.js** script opens the **Create DOM Project** dialog . Use the default values, or enter new values in the fields.

- **Project name**: specifies the DOM project name. The default value of this field is `MyDom`. Each DOM project in the workspace must have a unique project identifier.
- **Source directory name**: specify the source direct name for the project. The default value of this field is `src`.
- **Package name**: specify the package name for the project. This default value is `mydom`
- **DOM variable name**:specify the name for DOM variables. The default value is `mydom`.

Note: You must have an Application Factory project in your workspace to enable this wizard.

7 Click **OK** to create the DOM plugin project. The project appears in the **Package Explorer** view list.

To import an Eclipse Monkey DOM Template Application from the Import Application Module wizard:

1 Open JBuilder 2008:

2 Select the menu path **File ▸ Import ▸ Application Factory ▸ Import Application Module**:.

3 This opens the **Import Application Module** wizard. Select **Eclipse Monkey DOM Plugin Factory** in the table. Click **Finish**.

4 An**Import Application Module** dialog wizard appears asking if you want to run the application creation dom.js application creation script now or later. Click**Now** to immediately launch the dom.js script (see following procedure). If you select **Later**, the script can always be launched by:

- Double-clicking on dom.js in the **Script—Application Factory** view.
- Right-clicking on dom.js in the **Script—Application Factory** view and selecting **Execute Script**.

5 Executing the **dom.js** script opens the **Create DOM Project** dialog . Use the default values, or enter new values in the fields.

- **Project name**: specifies the DOM project name. The default value of this field is `MyDom`. Each DOM project in the workspace must have a unique project identifier.
- **Source directory name**: specify the source direct name for the project. The default value of this field is `src`.
- **Package name**: specify the package name for the project. This default value is `mydom`
- **DOM variable name**:specify the name for DOM variables. The default value is `mydom`.

Note: You must have an Application Factory project in your workspace to enable this wizard.

6 Click **OK** to create the DOM plugin project. The project appears in the **Package Explorer** view list.

To deploy your Eclipse Monkey DOM plugin application:

1 Select the menu path **File ▸ Export ▸ Plug-in Development ▸ Deployable plug-ins and fragments**. Click **Next**.

2 Select your project and browse to or enter a destination directory to which to deploy the application (such as / JBuilder 2008/thirdparty/eclipse/).

3 Click **Finish**.

4 To verify deployment, restart your IDE and select **File ▸ New ▸ Other ▸ Application Factory ▸ Script for Application Factory**. Click **Finish**. Check page 2 of this wizard to see that your DOM is listed.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Creating an Application Factory Project

When working with an Application Module, the workspace must contain an existing Application Factory project.

To create a new Application Factory project:

1 Switch to the **Application Factory Producer** perspective. Select **File ▸ New ▸ Application Factory Project**.

2 Or, specifically open the **New Application Factory Factory Project** wizard view in your current Application Factory perspective by selecting one of the following paths:

- **Window ▸ New ▸ Project ▸ Application Factory ▸ Application Factory Project**
- **Window ▸ New ▸ Other ▸ Application Factory ▸ Application Factory Project**

3 Enter a name for the Application Factory project.

4 Check the desired option(s):

- **Import global scripts and templates**
- **Open skeleton readme file**
- **Open skeleton cheatsheet file**

5 Click **Finish**. If all options have been selected in the previous step, an Application Factory project is created in the workspace with a template readme, template cheat sheet, and an empty application and tag diagram.

To create (import) an Application Module:

1 Click on the Create Application button in any of the Application Module editor tabs.

2 The Application Factory project extracts and invokes the application creation script (if defined) for the application module.

3 An application creation script can be used to perform any application-specific configuration actions. With the included JSF, Spring MVC, or Struts 2 data-aware applications, a multi-page **New {JSF | Spring MVC | Struts 2} Data-Aware Web Application** wizard opens and is completed to configure these applications. Refer to the dialog reference for these specific data-aware web application wizards for more details.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Creating and Using Add-on Modules

When an Application Module is exported from Application Factory, the producer of that module has an option to allow that module to be exported as an add-on module. When this module is later consumed (imported) into the IDE, the contents of that module are added to an Application Module that already exists in the workspace.

To publish a module as an Add-on Module:

1 With the module to be exported in your workspace, choose the menu path **File** ▸ **Export** ▸ **Application Factory** ▸ **Export Application Module**.

2 Page 1 of the **Export Application Module** wizard opens. Select the projects to be included in the module archive. Click **Next**.

3 Page 2 of the **Export Application Module** wizard opens. Accept defaults or specify values in the dialog page fields. Refer to the task topicPublishing an Application Module .

4 To publish (export) this module as an add-on module so that it can be consumed (imported) later into an existing module in the workspace, check the **Allow this module to be add-on to other application modules** option.

5 Click **Finish**.

6 Open the **Application Factory Explorer** in the IDE to view the application module that was just published. Refer to the task subtopic Browsing Modules in Application Explorer View for more information on this procedure.

To filter on Add-on modules in the Application Factory Explorer:

1 To apply a filter to see only add-on supporting template applications appear in the right-hand pane of the **Application Factory Explorer** view, scroll to the **Add-on Module** group on the left-side of the**Application Factory Explorer** view .

2 Click **Add-on Module** at the top level to show all template applications in the right-side of the **Application Factory Explorer** view.

3 Click **Add-on** at the sublist level to show all published add-on applications in the right-side of the **Application Factory Explorer** view.

4 Click **Not Add-on Module** at the sublist level to show all non-add-on applications in the right-side of the **Application Factory Explorer** view.

To consume (import) an Add-on Application Module:

1 Open the add-on module from the **Application Factory Explorer** . You can filter for add-on modules in the left-side pane. See the previous subtopic on filtering for add-on modules.

2 Click on **Add Application** in any of the **Application Module Editor** tabs (Preview, Diagram, Tags, or License).

3 The imported contents of the add-on module are subsumed into the current Application Module in the workspace. Once imported as add-on, modules are not available as a separate module but as files in the current Application Module in the workspace.

4 The consumed add-on module is created in a subdirectory of the existing workspace Application Module. The subdirectory is located under the current Application Module's `add-on module` directory. It has the add-on module's name as the parent directory name. For example, if you import a module named `Test` as an add-on module to your existing Application Module named `FirstModule`, the add-on module is created under the`FirstModule/Add-on-modules/Test` directory. The added module's scripts are visible in the scripts view, and its tags are added to the tags of the parent module.

Note: Add-on modules can also be created as regular stand-alone modules in your workspace. They have the added feature of being apart of an existing modules's contents.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Creating and Using RSS/Atom Feeds

When an Application Module is exported, the producer of the module can designate an RSS/Atom feed file to accompany the module for later deployment. The feed file can then be added to the consumer's file import location preferences. Later, a consumer can import the module using the RSS/Atom feed file reference. Application Factory supports generating and reading both RSS and Atom feed type files.

To publish an RSS/AtomApplication Factoryfeed file as a module deployment location:

1. From the **Application Factory Producer** perspective, publish (export) your workspace application project by selecting **File ▸ Export ▸ Application Factory ▸ Export Application Module**.
2. Page 1 of the **Export Application Module** wizard opens. Select the projects to be included in the published project's module archive. Click **Next**.
3. Page 2 of the **Export Application Module** wizard opens. Accept the dialog field defaults or enter other values as detailed in Publishing an Application Module.
4. To create an RSS/Atom feed file to accompany the published module for later importation and deployment, expand the **RSS/Atom Feed** section.

   Note: You must physically deploy both the created RSS/Atom feed file and the associated module archive (.mar file) to the location from which you want them accessed by the consumer.

   Once expanded, complete the following fields in the **RSS/Atom Feed File** area:

   - **Create feed file**—check to allow creating an RSS/Atom feed file.
   - **Feed type**—select the type of feed you would like to include from the dropdown menu RSS 2.0 is the default value.
   - **Feed file**—specify the name of (or browse to) the feed file location. The default location is *exportdirectoryname*/*modulename*.rss (or atom, depending on the feed type selected). Any spaces in the default module name are replaced with underscores (_).
   - **Module URL for feed**— specify the URL of the module archive to be stored in the RSS feed file for subsequent deployment. This should be the location from where the archive is later available for the consumer from a URL. You need to physically deploy the archive and the feed file to this location.

5. Click **Finish**. The module is published with an RSS/Atom feed. See the subtasks below for details on deploying the RSS/Atom feed file and associated module archive and adding it to the module search/export directory information.

To deploy an RSS/Atom feed file and associated module archive:

1. To deploy both the created RSS/Atom feed file and the associated module archive (.mar file) to the location specified in the **Export Application Module** wizard, move the files to the URL location specified in the wizard.
2. After physical deployment has occurred, add the RSS/Atom feed file URL to the **Module Search/Export Directories Preferences** page. See the following subtask.

To add RSS/Atom feed file URL to the Module Search/Export Directory path:

1. After physical deployment has occurred (see previous subtask), add the RSS/Atom feed file URL to the **Module Search/Export Directories Preferences** page.
2. Open the **Module Search/Export Directories Preferences** page by following the path **Window ▸ Preferences ▸ Application Factory ▸ Modules Search/Export Directories**.

3 At the bottom of the **Module Search/Export Directories Preferences** page, click **Add Feed**.

4 The **Specify RSS Feed URL** dialog opens. In **Feed Url**field of this dialog, enter the RSS/Atom URL feed address that was specified when you published the module (in the **Module URL for feed** field on page 2 of the **Export Application Module** wizard).

5 To check if this is a valid RSS/Atom feed file for use during Application Module import, press the **Test** button in the **Module Search/Export Directories Preferences** while the RSS/Atom feed location is selected in the list. This checks the format of the file plus the basic URL validity and indicates if it is a valid location. For instance, if the feed file is named *myModule.rss*, and you have physically deployed the feed file to a place that can be referenced by http://myhost:port:/rssLocation/myModule.rss to the feeds list, then this link is used to search for the application module referenced in the feed file when an import is requested. The module itself is not loaded until an import is requested. The feed file contains information about the referenced application module and allows importing when required.

6 The specified RSS/Atom feed can now be read as a location for importing an Application Module from the **Application Factory Explorer**, or from the **Import Application Module** wizard.

To filter on RSS/Atom feed locations in the Application Factory Explorer:

1 If the published RSS/Atom feed file location has been added in the **Module Search/Export Directories Preferences** page, and refers to a valid Application Module RSS/Atom referencing feed file, you can filter the **Application Factory Explorer** view to show only RSS/Atom import location modules.

2 To apply a filter for the RSS/Atom feed import location to the applications that appear in the right-hand pane of the **Application Factory Explorer** view, select the **Import Location** item on the left-side of the**Application Factory Explorer** view.

3 Click **Import Location** at the top level to show all locations for module import in the right-side of the **Application Factory Explorer** view.

4 Click any of the sub-items of the **Import Location** list to narrow the focus of import locations that are shown in the right-side pane of the **Application Factory Explorer** view.

5 Click **RSS/Atom Modules** at the sub-list level to show all modules that can be imported from RSS/Atom feeds in the right-side of the **Application Factory Explorer** view.

To consume (import) a module with an RSS/Atom feed location:

1 If a RSS/Atom feed location has been properly published, deployed and added as an import location in the **Module Search/Export Directories Preferences**, it can be used as an import location when consuming the module. (See preceding subtasks.)

2 The RSS/Atom feed URL is read as an Application Module import location when importing a module in the **Application Factory Explorer**, or importing using the **Import Application Module** wizard.

- To import a module from the **Application Factory Explorer** view, double-click on the project in the right-side pane.
- To import a module using **Import Application Module** wizard, select the menu path **Window ▶ Import ▶ Application Factory ▶ Import Application Module**. Select the RSS/Atom feed referenced module from the list. Click **Finish**.

3 The module opens the preview pane in the **Application Module Editor**. Select **Create Application** or **Add Application** to load the remotely deployed Application Module.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Editing an Application Module Cheat Sheet

A template cheat sheet for an application module can be included when creating a new Application Factory project.

To edit an Application Module template cheat sheet:

1. If there is not an Application Factory project, create a new Application Factory project using the procedure Creating an Application Factory Project. Select the option **Open skeleton cheatsheet file**.
2. If not currently the active perspective, switch to the **Application Factory Producer** perspective. If this perspective has already been open, you can switch to it by clicking on the appropriate icon in the toolbar.
3. Double-click on the cheat sheet XML (cheetsheet.xml) file in the root of the Application Factory project to open it using the Simple Cheat Sheet Editor. Through this editor, you can modify, remove or add steps to the cheat sheet template.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Working with Application Modules
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Editing an Application Module Readme

A template readme file or an Application Module can be included when creating a new Application Factory project.

To edit an Application Module template readme:

1. If there is not an Application Factory project, create a new Application Factory project using the procedure Creating an Application Factory Project. Select the option **Open skeleton readme file**.
2. If not currently the active perspective, switch to the **Application Factory Producer** perspective. If this perspective has already been open, you can switch to it by clicking on the appropriate icon in the toolbar.
3. Double-click on the readme HTML (readme.html) file in the root of the Application Factory project to open it using the HTML Editor. Through this editor, you can modify the HTML code to remove, add or modify information in the readme template.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Working with Application Modules
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Editing Application Modules

Application Modules can be edited using the **Application Module Editor**. Application modules are editable either after creating an instance of the application module using the **Application Module Editor** or by creating a new Application Factory project.

To edit Application Module properties:

1. Double-click on the application.adex file at the root of the Application Factory project to open the **Application Module Editor**.
2. Click the **Preview** tab. Drag and drop screenshot images onto the green plus (+) sign to add screenshots to the module.
3. Click the **Diagram** tab to automatically generate a snapshot of the application diagram.
4. Click the **Tags** tab to automatically generate a snapshot of tags.
5. Click the **License** tab. Select the desired standard license or include a custom license.
6. Click the **Save** icon in the toolbar to save all changes.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Working with Application Modules
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Publishing an Application Module

Application Modules can be published (exported) using the **Application Module Editor**. The **Application Module Editor** is opened by double clicking on the application.adex file in the root of the Application Factory project.

To publish an Application Module:

1. If there is not an Application Factory project, create a new Application Factory project using the procedure Creating an Application Factory Project.
2. Double-click the application.adex file in the root of the Application Factory project to open the **Application Module Editor**.
3. Click the **Preview** tab. Drag and drop screenshot images onto the green plus (+) sign to add screenshots to the module.
4. Click the **Diagram** tab to automatically generate a snapshot of the application diagram.
5. Click the **Tags** tab to automatically generate a snapshot of tags.
6. Click the **License** tab. Select the desired standard license or include a custom license.
7. Click the **Save** icon in the toolbar to save all changes.
8. Click the **Export Module** button from any of the **Application Module Editor** tabs. This opens the 2–page **Export Application Module** wizard. This wizard can also be invoked using the menu option **File ▸ Export ▸ Application Factory ▸ Export Application Module**.
9. Select projects that are to be included in the Application Module archive (.mar). All projects in the workspace are included by default.
10. Click **Next**. Specify the archive name, description, application kind and framework in the appropriate fields.
11. Use the default application creation script or click the **Browse** button next to the **application creation script** field to select a script from the Application Factory project. The application creation script is invoked when the module is created (consumed).
12. Click the **Configure export directory** link. This opens the **Module Search/Export Directories** Preferences page, where changes can be made for the default export directory location for the Application Module archive. Refer to the task subtopic Setting an Application Module Directory for more information on this procedure.
13. Click **Include source directories** to include all source directories along with the module for exportation.
14. Click **Allow this module to be add-on to other application modules** to allow the module you are publishing (exporting) to be consumed (imported) as an add-on module. If a module is exported as an add-on module, this module can later be imported into the IDE and its contents added to an Application Module that already exists in the workspace.

    Refer to the Creating and Using Add-on Modules.
15. Click the **RSS/Atom Feed** link to create an RSS/Atom feed file to accompany the module for later deployment. Note that you must physically deploy both the created RSS/Atom feed file and the associated module archive (.mar file) to the location specified in the **Export Application Module** wizard. Once this deployment has occurred, you can add an RSS feed URL to the **Module Search/Export Directories Preferences** page. This causes the specified RSS feed to be read as a location for importing an Application Module from the **Application Factory Explorer**, or from the **Import Application Module** wizard.

    Once expanded, complete the following fields in the **RSS/Atom Feed File** area:
    - **Create feed file**—check to want to create an RSS/Atom feed file as a location for later deployment
    - **Feed type**—select the type of feed you would like to include from the dropdown menu RSS 2.0 is the default value.
    - **Feed file**—specify the name of (or browse to) the feed file location. The default location is *exportdirectoryname*/*modulename*.rss. Any spaces in the default module name are replaced with underscores (_).

- **Module URL for feed**— specify the URL of the ultimately deployed module archive to be stored in the RSS feed file.

Refer to the Creating and Using RSS/Atom Feeds.

16. Click **Finish** to complete the export action. This creates the Application Module archive (.mar) in the specified export directory.
17. Open the **Application Factory Explorer** in the IDE to view the application module that was just created. Refer to the task subtopic Browsing Modules in Application Explorer View for more information on this procedure.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Working with Application Modules
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Setting an Application Module Search or Export Directory

Application Modules can be published (exported) using the **Application Module Editor**. The default export directory for the application module archive (.mar) can be set using the **Preferences** dialog. The Preferences dialog also allows the configuration of multiple application module search directories in which to search for Application Modules in the **Application Factory Explorer** view.

To change the export directory or to add a module search directory:

1. Click **Window** ▸ **Preferences**.
2. Expand the Application Factory node.
3. Click **Module Search/Export Directories** to open that Preferences page.
4. Click on **Add Directory** to add a search directory to the list.
5. Check any directory in the list to make it the default directory for exporting (publishing) Application Modules.

Related Concepts

Application Factory Overview
Workbench Features of Application Factory
Application Factory Modules
Application Factory Concepts

Related Tasks

Working with Application Modules
Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

TeamInsight Procedures

This section describes how team members configure their client machines to enable the TeamInsight development tools. Team members use the TeamInsight tools and the TeamInsight Viewer in the IDE to create projects, assign tasks, monitor bugs, control source code versions, and integrate builds into the development process.

In This Section

Planning Your Work: Creating Tasks in XPlanner
Describes how to create a user story in XPlanner, and describes the attributes that the Administrator must assign to each team member.

Querying Bugzilla for Bug Reports
Describes how to log in, change the password and use TeamInsight Bugzilla to report, track and fix product software bugs.

Tracking Your Time and Completing Tasks in XPlanner
Describes how to examine tasks, track your time devoted to tasks, and complete your tasks in XPlanner.

Using Continuum/Maven for Continuous Integration Builds
Describes how Maven/Continuum provides continuous builds.

Using the Subversion Viewer for Browsing the Project Repository
Describes how to use the Subversion Viewer to browse the Subversion repository. One of the TeamInsight tools is Sventon, a read-only repository browser.

Adding Mylyn Repositories for Bugzilla and XPlanner

JBuilder 2008 enables you to include the Bugzilla repository bugs and XPlanner repository tasks in the Eclipse **Task List** view, and to use Mylyn to define queries against those repositories. The Mylyn plugin offers task-focused user capabilities for JIRA, Bugzilla, Trac, and XPlanner. After you activate a task, Mylyn remembers the context of your subsequent work, such as the files associated with the active task. Later when you return to the task, the preserved context enables you to work more efficiently.

Using Mylyn, you can:

- Connect to task- or bug-tracking repository
- Define a query against the repository so that bugs or tasks are represented as Mylyn tasks in the development environment
- Define tasks related to the repository
- View task or bug reports locally or in an embedded browser
- Activate tasks and focus on the active task
- Save task context, including files and file hierarchy
- Work with tasks offline and resynchronize with the repository at a later time

Note: If you are using a source repository that supports Mylyn (currently CVS and Subversion), and the Eclipse plugin for it, you can commit your source changes based upon a change context associated with a Mylyn task. Mylyn automatically creates a comment that includes the task description

To use Mylyn with a Bugzilla or XPlanner repository:

1. Click Window ▸ Configure Mylyn and select either the Bugzilla or the XPlanner repository.
2. On the **Configure Mylyn** dialog box, enter your password for the repository you selected. Click **OK**.

   The **Task List** and the **Task Repositories** views open, displaying a repository entry, and your tasks or bugs query for the repository you selected.
3. On the **Task Repositories** view, you can verify your logon, if necessary, with the appropriate server by double-clicking the repository icon and clicking **Validate Settings**.
4. On the **Task List** view, right-click the repository icon to display the Mylyn context menu. Either select **Open** to open the predefined query for the repository, or select **New Query** to create a new query for a selected repository.
5. If you chose to edit the existing query, on the **Edit Repository Query** dialog box, select the entities (bugs, tasks, user stories, or project iterations) for which you want to see information. Click **Finish**.

   If you select Tasks in the Grouping field, a single query node is created in the Task List view, with all the applicable tasks underneath it. If you select User Stories in the Grouping field, then you get a query node for each selected user story in the Task List view, and each of the query nodes has task children that are associated with the parent user story.

   Additionally, you can control the scope of the tasks that are created – if you select All in the Scope group, all tasks from selected XPlanner entities are added to the query results. If you select My, then only your own tasks are added to the query results

   The Task List displays the selected tasks or bugs from the repository.

   If you choose to edit the Bugzilla query through the Mylyn Connector view, you see a dialog with options similar to those you would get for a Bugzilla query through a web page view.
6. To open the task or bug in an editor window, double-click the item in the **Task List**.

If you are opening an XPlanner task, a detailed editor appears that allows you to change the task name, description, and estimated time. You can also switch to the Browser tab to see the native XPlanner task or the Bugzilla bug editing web page.

7 To activate a task, click the icon in the left-hand column of the **Task List**. To focus Mylyn on the current task in the Task List, click the **Focus on workweek** button. To focus Mylyn on the current task in the Package Explorer or the Outline View, click the **Focus on workweek** button.

8 To open a task or bug with a known ID, select **Navigate ▸ Open Repository Task** from the main Eclipse menu. Type in the ID of the task or bug you want information about in the dialog. You must select the repository associated with the task or bug ID. If you want the task/bug to get added to your **Task List** view, check the **Add to Task List category** field on the dialog and indicate the category where the task/bug should be added (Root is the default).

Related Concepts

ProjectAssist and TeamInsight Overview
Mylyn Concepts

Related Tasks

Configuring Your TeamInsight Client

Related Reference

External Documentation for Mylyn from Eclipse.org
External Documentation about Mylyn Connectors to Repositories
External Article: Task-Focused Programming with Mylyn

Adding Mylyn Repositories for StarTeam Change Requests or Task Planning

(JBuilder 2008 Enterprise Edition JBuilder 2008 enables you to add StarTeam repository change requests and StarTeam repository tasks to the Eclipse **Task List** view, and to use Mylyn to define queries against those repositories. The Mylyn plugin adds task-focused user capabilities to the IDE. After you activate a task, Mylyn remembers the context of your subsequent work, such as the files associated with the active task. Later when you return to the task, the preserved context enables you to work more efficiently.

Using Mylyn, you can:

- Connect to repositories for tasks or change requests
- Define a query against the repository so that change requests or tasks are represented as Mylyn tasks in the development environment
- Define tasks related to the repository
- View task or change request reports locally or in an embedded browser
- Activate tasks and focus on the active task
- Save task context, including files and file hierarchy
- Work with tasks offline and resynchronize with the repository at a later time

To use Mylyn with an assimilated StarTeam installation repository:

1. Click Window ▸ Configure Mylyn and select a project configuration that references a StarTeam repository.
2. On the **Configure Mylyn Repository for StarTeam :Change Requests/Tasks** dialog box, enter your password for the repository you selected. Click **Validate** to validate the user name/password and click **OK**.

   The **Task List** and the **Task Repositories** views open, displaying a repository entry, and your tasks or change requests query for the repository you selected.
3. On the **Task Repositories** view, verify your logon, if necessary, with the appropriate server by double-clicking the repository icon. This opens the **Task Repository Settings:StarTeam Repository Settings**dialog. Click the **Validate Settings**button to validate the server and login settings for this repository.
4. A StarTeam repository can be defined to show tasks, change requests, or both entities through the **Task Repository Settings:StarTeam Repository Settings** dialog or the **Add Task Repository:StarTeam Repository Settings** dialog. Check the Change Request and/or the Task boxes in the **StarTeam Repository Type** area of the dialog page. Based upon this selection, either tasks and/or change requests will appear as selectable items in the **New StarTeam Query** dialog. If you select **All my tasks** and **All my change requests** in this dialog, you will see two queries created in the **Task List**, one for "my tasks" and one for "my change requests".
5. On the **Task List** view, right-click the repository icon to display the Mylyn context menu. Select New Query to create a new query for a selected repository. This opens the **New Repository Query**dialog . Click **Next** to go to **New StarTeam Query** dialog, which allows you to name the query, chose all or selected tasks and change requests, and the scope.
6. If you chose to edit the existing query, on the **Edit Repository Query** dialog box, select the entities you want to see (tasks, or change requests). Click **Finish**.

   . If you select Tasks in the **Type** field, a single query node is created in the Task List view, with all the applicable tasks sublisted. If you select Change Requests in the **Type** field, a single query node is created in the **Task List** view, with all the applicable change requests sublisted. Selecting both generates two lists in the **Task List** view. (See item 4.)

Additionally, you can control the scope of the tasks that are created – if you select All in the **Scope** group, all tasks or change requests from selected StarTeam entities are added to the query results. If you select My, then only your own tasks or change requests are added to the query results

The Task List displays the selected tasks or change requests from the repository.

7 To open a task or change request in an editor window, double-click the task/change request in the **Task List**.

This opens a detailed **Edit Repository Query** dialog window that allows you to change the selected tasks or change requests in the query, and the type or scope of the query.

8 To activate a task or change request, click the icon in the left-hand column of the Task List. To focus Mylyn on the current task in the Task List, click the **Focus on workweek** button. To focus Mylyn on the current task in the Package Explorer or the Outline View, click the **Focus on workweek** button.

Related Concepts

ProjectAssist and TeamInsight Overview
Mylyn Concepts

Related Tasks

Adding Mylyn Repositories for Bugzilla and XPlanner
Configuring Your TeamInsight Client

Related Reference

StarTeam Repository Settings
New StarTeam Query
External Documentation for Mylyn from Eclipse.org
External Documentation about Mylyn Connectors to Repositories
External Article: Task-Focused Programming with Mylyn

Adding Team Members in XPlanner (Administrator Task)

Two administrators have overlapping functions in XPlanner:

The **ProjectAssist Administrator** adds users for each of the TeamInsight tools during installation. Similarly, when new users need to be added to XPlanner, the ProjectAssist Administrator uses the system console to add users.

The **XPlanner Administrator** can add people as team members for any specific project in XPlanner. However, if the XPlanner Administrator adds a new user to the list of People in XPlanner, the user is not thereby added to TeamInsight. This situation can cause problems with the TeamInsight project tools.

Tip: To maintain the consistency of the TeamInsight project, only the ProjectAssist Administrator should add users.

To add a person in XPlanner (outside of TeamInsight):

1. On the ProjectAssist system console, log on as Administrator.
2. Enter XPlanner by selecting **Window** ▶ **Open TeamAssist Viewer** ▶ **XPlanner**.
3. On the top ((**XPlanner Projects**) page, click **People**.
4. On the **People** page, click **Add Person**.
5. On the **Create Profile** page:

- Enter the user's name, such as Joe Bloggs.
- Create a user ID – either a user number such as 183 or a user name such as jbloggs.
- Enter the user's initials (such as jb), E-mail address, and phone number. XPlanner uses a person's initials to indicate the person associated with a given task or story. Phone number and E-mail address are contact information for the team.
- The **Hide?** box controls whether the person is listed in the People list available to developers.
- Enter and confirm a temporary password for the new user.
- Select the appropriate role for the new user: None, Viewer, Editor, Admin.
- Select the project

6. Click **Add**.

To add a team member to a project in XPlanner

1. Start XPlanner by selecting **Window** ▶ **Open TeamAssist Viewer** ▶ **XPlanner**.
2. On the **Project** page, click **People**.
3. On the **People on Project** page, do something else??
4. If the following statement is true, then this procedure is not necessary.

Team members listed in the **People** page are available to work on any project.

Related Concepts

XPlanner: Project and Team Management
Creating and Starting Project Iterations in XPlanner (Administrator Task)

Related Tasks

Planning a Product Feature: Creating a User Story in XPlanner
Planning Your Work: Creating Tasks in XPlanner

Administering the Liferay Portal

The Liferay Administrator can use the **Admin** tab in the Liferay portal to customize the portal to match the needs of the project team. The **Admin** tab is only displayed immediately after the initial logon by the Liferay Administrator.

There is also a **Setup** tab within a TeamInsight component's portlet allows the Liferay Administrator to reconfigure that portlet. This can be useful for resetting passwords for portlets that were changed or to change other configuration settings that have been invalidated after the install.

Changes made through the **Setup** tab take effect immediately with no restart of the server needed. Some changes are localized to the portlet of a particular project (for example, the Build Status portlet setting that contains the ID used by Continuum to identify a particular project). However, most settings are common to all instances of a portlet regardless of the project (for example, the URL used to access the application).

Note: To configure the Liferay portal, you must be the Liferay Administrator. See your ProjectAssist Administrator for permissions.

To open the Liferay Administrator (Admin) page:

1 Select **Window** ▶ **OpenTeamInsight Viewer** ▶ **Liferay**.

The **Liferay Portal** opens, displaying the sign-in command in the upper right corner of the Liferay portal.

2 Click **Sign-in** to display the sign-in dialog.

3 Enter your user name (typically your email address) and your password. Then click **Sign in**.

The Liferay portal displays the **Admin** page. This is the Administrator page, which is only available immediately after the Liferay Administrator logs on to Liferay.

4 On the **Admin** page, you can configure the Liferay portlet using the following tabs:

- Server
- Auto Deploy
- Enterprise
- Portlets
- Users
- Live Sessions
- Default Groups and Roles
- Reserved Users
- Mail Host Names
- Emails

To access the Liferay Administrator's Setup tab:

1 Select **Window** ▶ **OpenTeamInsight Viewer** ▶ **Liferay**.

The **Liferay Portal** opens, displaying the sign-in command in the upper right corner of the Liferay portal.

2 Click **Sign-in** to display the sign-in dialog.

3 Enter your Administrator's user name (typically your email address) and your password. Then click **Sign in**.

The Liferay portal displays the **Admin** page. Other tabs are shown for **Sample Projects** and **Configurations**. The **Setup** tab is in the user interface for the individual portlets that are provided through Liferay when you are logged on as the Liferay administrator.

4 Click on the **Sample Projects** or **Configurations** tab to see the individual portlets for the TeamInsight components. The **Setup** tab is on the individual portlet pages.

5 Change any field on the **Setup** tab as desired. Changes made through the **Setup** tab take effect immediately with no restart of the server needed.

Related Concepts

ProjectAssist and TeamInsight Overview
Liferay: The TeamInsight Project Portal

Related Tasks

Configuring Your TeamInsight Client
Changing Your Passwords for the TeamInsight Tools

Changing Your Passwords for the TeamInsight Tools

The first time you log on to the Liferay project portal and the TeamInsight tools, you use a universal, temporary password that you receive in E-mail from the ProjectAssist Administrator.

To change your password in each of the tools, navigate to the locations described in this topic.

To find the password change fields in the TeamInsight tools:

1 Click **Window** ▸ **Open TeamInsight Viewer** ▸ **Open All**.

2 Navigate to the appropriate location in each of the TeamInsight tools:

- **Bugzilla**: The Bugzillla home page contains both the change password and logon commands. You can search the Bugzilla database without logging on.
- **Continuum**: Only the ProjectAssist Administrator can change passwords for Continuum.
- **Liferay**: On the Liferay project portal, click **My Account**. Then click the **Password** tab.
- **Subversion Viewer**: No password is required for the Subversion Viewer (read-only).
- **Subversion Repository**: On the Liferay project portal, click the **Configuration** tab at the top of the page. (For a Subversion repository assimilated into JBuilder 2008, use the password mechanism of the original Subversion version control system.)
- **XPlanner**: On any XPlanner page, click **Me**. On your personal profile page, click **Edit**.

3 Enter your new password in the appropriate location.

Related Concepts

ProjectAssist and TeamInsight Overview
Subversion: Source Code Repository

Related Tasks

Opening the TeamInsight Viewer and the Liferay Portal

Checking Out a Project, Making Changes, and Checking Your Changes Into the Repository

Your typical work pattern with Subversion is **Edit-Update-Commit**. Start by checking out a local copy of the repository into your workspace. Edit the files in your workspace. Typically, you **update** your files (merge the changes from the repository into your working files). Finally, you **commit** your changes to the repository (check your files into the repository). An Edit-Update-Commit workflow is the way to synchronize the repository and your workspace.

To check out a local copy of the project repository:

1. Click **Project** ▸ **Checkout Project** and select the repository.

   If the repository is not configured, select **Configure** and locate the TeamInsight.ticx file that defines your Subversion repository. Your ProjectAssist Administrator distributes the TeamInsight.ticx file.
2. The repository is displayed in the **Navigator** view.
3. Expand the tree structure to locate the files that you want to open. Double-click a file to open it in the **Editor**.

To update your local working files:

1. In the **Navigator** view, right-click the file or files for which you want to update the local working copy.
2. Select **Update**.

   Subversion merges the changes from the repository into your selected files.

To commit your local files to the repository:

1. In the **Navigator** view, right-click the file or files you want to commit to the repository.
2. Select **Commit**.

   Subversion checks your working copy into the repository, creating a new version.

For more information about using Subclipse (the Subversion plug-in for Eclipse), see the Subclipse online help inside the help for Eclipse.

Tip: To preview the changes between your local working copy and the files in the repository, you can perform a “diff” operation using the Subversion read-only browser. To examine differences between the local copy and the repository, use the **Synchronize** view.

Related Concepts

Subversion: Source Code Repository

Related Tasks

Using the Subversion Viewer for Browsing the Project Repository

Configuring Your TeamInsight Client

TeamInsight team members import a configuration file to their local workstations to configure and setup the TeamInsight tools. If the ProjectAssist Administrator changes the project configuration at a later time, team members must import a changed configuration file.

Note: Only the ProjectAssist Administrator can install the server-side software and distribute the configuration file to enable the TeamInsight tools. The TeamInsight tools can be used on the following platforms:

- Microsoft® Windows® XP Professional (SP2)
- Microsoft® Windows® Vista 32-bit
- Red Hat Enterprise Linux
- Macintosh® OS X

To configure the TeamInsight client and access the TeamInsight tools:

1 After the ProjectAssist Administrator installs the server side, verify that the following commands are present:

Window ▸ **Configure Mylyn** contains a Configure command and lists No installed configurations.

Window ▸ **Open TeamInsight Viewer** contains a Configure command and lists No installed configurations.

Project ▸ **Checkout Project** contains a Configure command and lists No installed configurations.
If these commands are not present, your TeamInsight tools are not correctly installed. You might need to reinstall the software. See your Administrator for help.

2 Locate the **TeamInsight.ticx** file for your project and copy the file to your local system.

Your Administrator sends you an email with an attached TeamInsight.ticx file. The email gives your user ID and temporary password, and it also includes the configuration file as an attachment.

The TeamInsight.ticx file enables your access to all the TeamInsight tools, including the Subversion repository.

Note: After you have configured TeamInsight for the first time, you can import a new TeamInsight.ticx file by opening the **Configuration** page at the top of the Liferay portal. The .ticx file available from the **Configuration** page provides a configuration specifically for the team member logged into the Liferay portal. Therefore, sending a copy of this configuration file to another team member might not be appropriate unless that team member has access to the same applications.

3 Click any one of the three Configure commands for the TeamInsight tools:

- **Window** ▸ **Configure Mylyn** ▸ **Configure**
- **Window** ▸ **Open TeamInsight Viewer** ▸ **Configure**
- **Project** ▸ **Checkout Project** ▸ **Configure**

4 On the **TeamInsight Configuration File** dialog box, navigate to the location of your project's **.ticx** file.

5 Click **Open**.

JBuilder 2007 displays a message confirming that menu configurations for the three commands have been imported successfully. You are now ready to use the TeamInsight tools for software development.

Note: If the confirmation message does not appear, or if an error is displayed, see your Administrator for help.

6 To verify that the TeamInsight client is correctly configured, click the following menus:

- **Window** ▸ **Configure Mylyn** lists Bugzilla, XPlanner, or StarTeam.
- **Window** ▸ **Open TeamInsight Viewer** lists the TeamInsight web components that were selected during the ProjectAssist server installation (such as Bugzilla, Continuum, Liferay, Subversion Viewer, XPlanner, CVS, or Borland's ALM StarTeam).
- **Project** ▸ **Checkout Project** is present.

By following any of these menu paths, you should see the URLs of the servers for the various TeamInsight tools that were configured or assimilated (such as Bugzilla, XPlanner, Continuum, CVS, StarTeam, and so forth).

Note: After installing and configuring the TeamInsight tools, open the TeamInsight Viewer, open each applicable TeamInsight tool, and change your temporary password in each of the tools.

To specify URL favorite links Inside TeamInsight Viewer:

1 You can add URLs of your choice to the TeamInsight Viewer through the JBuilder 2008 **Window** menu.

2 Go to **Window** ▸ **Open TeamInsight Viewer** ▸ **Edit Favorite Links** to add any favorite URL links that will be accessible from inside your TeamInsight Viewer. A tab for each favorite link added appears at the bottom of the TeamInsight Viewer.

3 You can follow the **Window** ▸ **Open TeamInsight Viewer** ▸ **Edit Favorite Links** to edit or remove any favorite URLs from your TeamInsight Viewer.

To delete configuration through Window menu selection:

1 You can delete a TeamInsight configuration through the JBuilder 2008 **Window** menu.

2 Go to **Window** ▸ **Open TeamInsight Viewer** ▸ **Delete Configuration** to remove an imported TeamInsight configuration.

Related Concepts

ProjectAssist and TeamInsight Overview
Liferay: The TeamInsight Project Portal

Related Tasks

Opening the TeamInsight Viewer and the Liferay Portal
Changing Your Passwords for the TeamInsight Tools
Adding Mylyn Repositories for Bugzilla and XPlanner

Creating and Starting Project Iterations in XPlanner (Administrator Task)

Any XPlanner user can create and start iterations for projects in XPlanner. Iterations are typically short, only a few weeks. Projects typically have only one iteration started at a time (the current iteration).

Warning: Do not create projects from inside XPlanner if you want the project to be connected to the TeamInsight tools. Only the ProjectAssist Administrator can create projects that share the TeamInsight tools.

To create an iteration in XPlanner:

1. Enter XPlanner either by selecting **Window** ▸ **Open TeamInsight Viewer** ▸ **XPlanner** or by selecting **Window** ▸ **Add Mylyn Repository** ▸ **XPlanner**.
2. On the **Top (XPlanner Projects)** page, click the appropriate **project name**.
3. On the **Project** page, click **Create Iteration**.
4. On the **Create Iteration** window, supply a **Name** for the iteration (such as Sprint 1 or Backlog), a **Start Date**, an **End Date**, and a **Description** of the iteration.
5. Click **Create**. To clear the fields on the **Create Iteration** window, click **Reset**.

To start an iteration in XPlanner

1. In XPlanner, navigate to the page of the specific iteration.
2. Click **Start**.

Team members listed in the **People** page are available to work on any project.

Related Concepts

XPlanner: Project and Team Management
Mylyn Concepts

Related Tasks

Planning a Product Feature: Creating a User Story in XPlanner
Monitoring Iteration Metrics in XPlanner
Adding Mylyn Repositories for Bugzilla and XPlanner

Creating or Generating Bug Reports in Bugzilla

The Bugzilla component of TeamInsight is loaded during the ProjectAssist server installation or when the Administrator adds a new project. The ProjectAssist Administrator initially creates the access for individual project team members to the TeamInsight Bugzilla tool and all users are assigned the same password, which should be changed by the user. Any user can file a Bugzilla report that can be viewed by the team.

To create a new bug report in Bugzilla:

1. After you reach the **Bugzilla Main Page** window, click on the **Enter a new bug report** link or **Actions ▸ New** to generate a new bug/defect report.
2. Select the product to report the bug against in the **Bugzilla Enter Bug** page. Click on appropriate link to report the bug against that product.
3. Complete the requested information about your bug report. Refer to The Bugzilla Guide for further information on completing these fields. All the members of your TeamInsight group are listed in the bug notification message. You can assign this bug to the appropriate development and QA person. All the members of the team receive notification of the new bug.
4. Click **Commit** to commit the bug into the repository.

To generate a bug report from the error log:

1. Bug reports can be created directly from the error log in Bugzilla. You may want to keep your error log open on the JBuilder 2007 main page. To open the error log on the main page, go to **Window ▸ Show View ▸ Error Log**.
2. With the error log open, you can right-click on any error and select **Report as Bug**.

Related Concepts

Bugzilla: Defect Tracking System
Mylyn Concepts

Related Tasks

Logging in to TeamInsight Bugzilla
Querying Bugzilla for Bug Reports
Managing Bug Reports in Bugzilla
Adding Mylyn Repositories for Bugzilla and XPlanner

Related Reference

Bugzilla Resources and Documents
The Bugzilla Guide

Logging in to TeamInsight Bugzilla

The Bugzilla component of TeamInsight is loaded during the ProjectAssist server installation or when the Administrator adds a new project. The ProjectAssist Administrator initially creates the access for individual project team members to the TeamInsight Bugzilla tool and all users are assigned the same password. You can search the Bugzilla database without logging on to Bugzilla.

To initially login to TeamInsight Bugzilla and change your password:

1. Enter TeamInsight Bugzilla either by selecting **Window ▸ Open TeamInsight Viewer ▸ Bugzilla** or by selecting **Window ▸ Add Mylyn Repository ▸ Bugzilla**. You can also select to load all TeamInsight components by selecting **Open All** in either one of these paths.
2. Select the **Bugzilla** TeamInsight Viewer by clicking on the **Bugzilla** tab at the bottom of the viewer.
3. Click on **Actions ▸ Login** to login in using your Administrator-assigned password.
4. After you reach the **Bugzilla Main Page** window, which shows the work flow for a bug report in Bugzilla, click on the **Change password or user preferences** to update to a more secure user password.
5. Enter the requested information to change your password in the **Bugzilla User Preferences** page on the **Account Preferences** tab. When done, click **Submit Changes** button.

Related Concepts

Bugzilla: Defect Tracking System
Mylyn Concepts

Related Tasks

Creating or Generating Bug Reports in Bugzilla
Querying Bugzilla for Bug Reports
Managing Bug Reports in Bugzilla
Adding Mylyn Repositories for Bugzilla and XPlanner

Related Reference

Bugzilla Resources and Documents
The Bugzilla Guide

Managing Bug Reports in Bugzilla

The Bugzilla component of TeamInsight allows the user to generate bug reports in views different from the standard bug report output. Reports can also be generated in graphical, tabular or chart views according to a variety of criteria.

To create bug report graphical and chart displays:

1. Along with the standard bug list, Bugzilla can generate two additional views of the bugs. These view include reports and charts. Reports give different views of the current database state. Charts plot the changes in sets of bugs over a specified time.
2. After you reach the **Bugzilla Main Page** window, click on the **Summary reports and charts** link or **Actions ▸ Reports**, from either the main page or a bug list search result page, to generate an alternate bug report view or chart.
3. The **Bugzilla Reporting and Charting Kitchen** page opens. From this page, you can select 3 types of report views and 1 type of chart view.
4. If you are interested in generating report views, click on one of the following links:

- **Search** takes you to the **Advanced Search** tab of the **Bugzilla Query** page. The generates the same report as a standard advanced search bug query.
- **Tabular reports** generates tables of bugs counts. You choose one or more fields as your axes, and then refine the set of bugs by completing the remainder of the fields on the **Bugzilla Generate Tabular Report** form. Click on **Generate Report** to view the report. Once the report appears, you can switch between Bar, Line, Table and CSV displays by clicking on the appropriate line at the end of your report.
- **Graphical reports** generates line graphs, bar and pie charts. You choose one or more fields as your axes, and then refine the set of bugs by completing the remainder of the fields on the **Bugzilla Generate Graphical Report** form. Click on **Generate Report** to view the report. Once the report appears, you can switch between Pie, Bar, Line, Table and CSV displays by clicking on the appropriate line at the end of your report.

5. Charts generate a view of the bug database state over time. If you are interested in generating chart views, click on the **Old Charts** link on the **Bugzilla Reporting and Charting Kitchen** page. The **Bugzilla Bug Charts** page opens. Select your product and one or more data sets that you want to chart. Click the **Continue** button and your resulting chart is displayed.

Related Concepts

Bugzilla: Defect Tracking System

Related Tasks

Logging in to TeamInsight Bugzilla
Creating or Generating Bug Reports in Bugzilla
Querying Bugzilla for Bug Reports

Related Reference

Bugzilla Resources and Documents
The Bugzilla Guide

Monitoring Iteration Metrics in XPlanner

Three XPlanner commands produce useful statistics about iterations: **Metrics**, **Charts**, and **Accuracy**.

To display statistics about an iteration

1. Enter XPlanner either by selecting **Window** ▶ **Open TeamInsight Viewer** ▶ **XPlanner** or by selecting **Window** ▶ **Add Mylyn Repository** ▶ **XPlanner**.
2. Navigate to the iteration you want to monitor. This can be any iteration, started or not.
3. On the **Iteration** page, click one of the following:

- **Metrics** to compare hours worked by team members, both solo and paired, as well as hours accepted by developers.
- **Charts** to display graphs and pie charts. The graphs represent both iteration progress (hours completed over time) and iteration burn down (remaining hours over time). The pie charts represent progress by task and by hour.
- **Accuracy** to display statistics about the accuracy of time estimates in the iteration.

Related Concepts

ProjectAssist and TeamInsight Overview
XPlanner: Project and Team Management
Mylyn Concepts

Related Tasks

Adding Mylyn Repositories for Bugzilla and XPlanner
Planning a Product Feature: Creating a User Story in XPlanner
Adding Team Members in XPlanner (Administrator Task)
Creating and Starting Project Iterations in XPlanner (Administrator Task)
Planning Your Work: Creating Tasks in XPlanner
Tracking Your Time and Completing Tasks in XPlanner
Moving or Continuing a Story or Task in XPlanner

Related Reference

XPlanner Documentation Available from XPlanner.org

Moving or Continuing a Story or Task in XPlanner

If a story or task is not completed in the original iteration, you can either move the story to a different iteration or move the task to a different story.

To move or continue a story:

1 Enter XPlanner either by selecting **Window** ▸ **Open TeamInsight Viewer** ▸ **XPlanner** or by selecting **Window** ▸ **Add Mylyn Repository** ▸ **XPlanner**.

2 Navigate to the **Iteration** page.

3 Do either of the following:

- Click the **Move/Continue** icon next to the story you want to move.
- Click the **ID** of the story you want to move. Then on the **Story page**, click the **Move/Continue** command, located at the bottom of the screen.

4 On the **Move/Continue Story** page, click the drop-down list of iterations, and select the destination for the story.

5 Click **Move** or **Continue** to move the story to the selected iteration. (To cancel the move, click the browser's Back button.)

To move or continue a task:

1 In XPlanner, navigate to the **Story page**.

2 Do either of the following:

- Click the **Move/Continue icon** next to the task you want to move.
- Select the **ID** of the task you want to move. Then on the **Task page**, click the **Move/Continue** command, located at the bottom of the screen.

3 On the **Move/Continue Task** page, click the drop-down list of stories, and select the destination for the task.

4 Click **Move** or **Continue** to move the task to the selected story. (To cancel the move, click your browser's Back button.)

Related Concepts

ProjectAssist and TeamInsight Overview
XPlanner: Project and Team Management
Mylyn Concepts

Related Tasks

Adding Mylyn Repositories for Bugzilla and XPlanner
Adding Team Members in XPlanner (Administrator Task)
Creating and Starting Project Iterations in XPlanner (Administrator Task)
Planning a Product Feature: Creating a User Story in XPlanner
Planning Your Work: Creating Tasks in XPlanner
Tracking Your Time and Completing Tasks in XPlanner
Monitoring Iteration Metrics in XPlanner

Related Reference

XPlanner Documentation Available from XPlanner.org

Opening the TeamInsight Viewer and the Liferay Portal

After you configure your local workstation for TeamInsight, you can access the Liferay project portal. The Liferay portal is a TeamInsight client tool that displays summary statistics and reports from plugins such as Kosmos, XPlanner, Continuum, Bugzilla, and QALab. You can also open any one TeamInsight tool or all of the tools at once in the TeamInsight Viewer.

To open one or all of the TeamInsight client tools in the TeamInsight Viewer:

1 Configure your workstation as a TeamInsight client.

2 Select **Window** ▸ **OpenTeamInsight Viewer** and do either of the following:

- Click the name of the tool you want to open (CVS, Bugzilla, Continuum, Liferay, Subversion Viewer, StarTeam or XPlanner).
- Click **Open All** to open all TeamInsight tools.

The **TeamInsight Viewer** opens and displays either the one tool you chose or a window with a tab for each of the tools. It will also contain tabs for any favorite URLs that you have added through the **Windows** ▸ **Open TeamInsight Viewer** ▸ **Edit Favorites Links** path. Depending on your recent logons, a TeamInsight tool might also display its logon window.

To open the Liferay project portal:

1 Make sure your workstation is configured as a TeamInsight client. (Clicking **Window** ▸ **OpenTeamInsight Viewer** displays the URLs of the TeamInsight tools.)

2 Select **Window** ▸ **OpenTeamInsight Viewer** ▸ **Liferay**.

3 If you are not logged in to use the TeamInsight tools, the **Sign In** window appears. Enter your user ID (typically your Email address) and your password, and click **Sign In**.

The Liferay portal displays portlets for any installed tools that provide project information and links as follows:

- Current status report from JBoss Labs Subversion repository monitor, including the most recent activity
- CVS repository information for project repositories
- Burn down chart and Current iteration details from XPlanner
- Build status from Continuum/Maven and a **Project Health** link for more information
- Bugzilla status (pages for bugs organized by Important, Newest, Severity, Assignee, and Trends)
- QALab Summary and QALab Classes giving results from the open-source Cobertura and PMD plugins
- StarTeam Task, Bugs and/or StarTeam version control repository information

The Liferay portal is a tabbed window that contains pages for all configured projects as well as a tabs labeled Configuration and Setup. The **Configuration** page contains:

- A portlet that links to the TeamInsight.ticx file for the current project
- The password change mechanism for a Subversion repository

The **Setup** tab within a TeamInsight component's portlet allows the Liferay Administrator to reconfigure that portlet. This can be useful for resetting passwords for portlets that were changed or to change other configuration settings that have been invalidated after the install. Changes made through the **Setup** tab take effect immediately with no restart of the server needed. Some changes are localized to the portlet of a particular project (for example, the Build Status portlet setting that contains the ID used by Continuum to identify a particular project). However, most settings are common to all instances of a portlet regardless of the project. (for example, the URL used to access the application).

Related Concepts

ProjectAssist and TeamInsight Overview
Liferay: The TeamInsight Project Portal

Related Reference

External Liferay Documentation

Planning a Product Feature: Creating a User Story in XPlanner

User stories describe features planned for a given project. The tasks inside a story represent the work required to complete the feature described in the story. Any user can create a user story and associated tasks in XPlanner. Each story has a **Customer** and a **Tracker** associated with the story. Typically, the Customer is the person who requires the feature represented in the story, and the Tracker is the person who is responsible for the completion of the story.

To create a user story in XPlanner:

1. Enter XPlanner by selecting **Window** ▸ **Open TeamInsight Viewer** ▸ **XPlanner** or by selecting **Window** ▸ **Add Mylyn Repository** ▸ **XPlanner**.
2. On the **Top (XPlanner Projects)** page, click the ID of your project, such as Sprint 3 or Backlog.
3. On the **Project** page, click the ID of the iteration where you want to add a story.
4. On the **iteration** page, click **Create Story**.
5. On the **Define Story** page, complete the fields as follows:

- **Name**: Enter a descriptive name for the story, such as New Font Widget.
- **Duration**: Enter the hours you have worked on this task.
- **Disposition**: Select from Planned, Carried Over, or Added.
- **Customer**: Enter the name of the person who requires or uses the product of the story.
- **Tracker**: Enter the name of the person who is responsible for completing the story.
- **Status**: Select from Draft, Defined, Estimated, Planned, Implemented, Verified, or Accepted.
- **Priority**: Enter an arbitrary number indicating relative priority of this story.
- **Estimated Hours**: Enter the number of hours you are estimating to complete the work for the story (such as 40 or 3.5).
- **Description**: Enter a description of the purpose and end result of the feature represented in this story. For example, “Add a new widget to the application that allows user to select the font displayed on the screen.”

6. To create a story using the parameters you have entered, click **Create**. To reset the fields and start over, click **Reset**.

Related Concepts

XPlanner: Project and Team Management
Mylyn Concepts

Related Tasks

Planning Your Work: Creating Tasks in XPlanner
Tracking Your Time and Completing Tasks in XPlanner
Adding Mylyn Repositories for Bugzilla and XPlanner

Planning Your Work: Creating Tasks in XPlanner

Any user can add tasks to a project in XPlanner. Each task has an **Acceptor** associated with the task. Typically, the Acceptor is the person who is assigned to complete the work in the task.

To create a task in XPlanner:

1. Enter XPlanner either by selecting **Window** ▸ **Open TeamInsight Viewer** ▸ **XPlanner** or by selecting **Window** ▸ **Add Mylyn Repository** ▸ **XPlanner**.
2. Navigate to your project and to a specific iteration in the project.
3. On the **Iteration** page, click the ID of a user story.
4. On the **Story** page, click **Create Task**.
5. On the **Define Task** page:
   - In the **Name** field, enter a name that summarizes the task. This is the only required field. Other fields can be easily changed later.
   - In the **Type** drop-down list, select the type of task (Feature, Defect, Debt, FTest, ATest, or Overhead).
   - In the **Disposition** drop-down list, select the a disposition (Planned, Discovered, Added, or Carried Over).
   - Assign a person from the **People** list as **Acceptor**. Select the person who is to perform the task.
   - In the **Estimated Hours** field, enter the number of hours to finish the task.
   - In **Description**, enter a description of the task, including necessary details to complete the task.
6. Click **Create**. To clear the fields on the **Define Task** page, click **Reset**.

Related Concepts

XPlanner: Project and Team Management
Mylyn Concepts

Related Tasks

Planning a Product Feature: Creating a User Story in XPlanner
Tracking Your Time and Completing Tasks in XPlanner
Adding Mylyn Repositories for Bugzilla and XPlanner

Querying Bugzilla for Bug Reports

The Bugzilla component of TeamInsight is loaded during the ProjectAssist server installation or when the Administrator adds a new project. The ProjectAssist Administrator initially creates the access for individual project team members to the TeamInsight Bugzilla tool and all users are assigned the same password.

To query Bugzilla for bug reports:

1. After you reach the **Bugzilla Main Page** window, click on the **Searching existing bug reports** link or **Actions ▸ Search** to search for existing bug reports, comments or patches. The **Bugzilla Query** page opens. You can select either the **Find a Specific Bug** tab or the **Advanced Search**tab.
2. By selecting the **Find a Specific Bug** tab, you can find a specific bug by entering words that describe it. Bugzilla searches bug descriptions and comments for the specified words and returns a list of matching bugs sorted by relevance. Select the appropriate choice from the **Status:** and **Product:** drop-down lists and enter your word search criteria in the **Words:** field. Click on the **Search** button to initiate your query.
3. By selecting the **Advanced Search** tab, you can narrow the search criteria by specifying a number of fields or options. Bugzilla searches bug descriptions and comments for the specified words and returns a list of matching bugs sorted by relevance. Select the appropriate choice from the following page items:

- **Summary:** includes a drop-down box to specify the type of string matching and an area to enter the search string text
- **Product:** is a drop-down list for selecting the product to which the bug will be applied
- **Component:** is a drop-down list for selecting the product component to which the bug is applicable
- **Version:**is a drop-down list for selecting the product version
- **A Comment:** is a drop-down list for selecting the search criterion and an area to enter the search string text
- **The URL:** is a drop-down list for selecting the search criterion and an area to enter the search string text
- **White Board:** is a drop-down list for selecting the search criterion and an area to enter the search string text
- **Keywords:** is a drop-down list for selecting the search criterion and an area to enter the search string text
- **Status:** is a drop-down list for selecting a search by bug status
- **Resolution:** is a drop-down list for selecting a search by bug resolution
- **Severity:** is a drop-down list for selecting a search by the bug severity
- **Priority:** is a drop-down list for selecting a search by assigned bug priority
- **Hardware:** is a drop-down list for selecting your computer hardware
- **OS:** is a drop-down list for selecting your operating system
- **Email and Numbering:** allows searching by email recipients or bug numbers according to the specified strings
- **Bug Changes:** allows searching by a specified date range for any of the selected change types selected in the drop-down list
- **Sort results by:** specifies the sort of for returned search values
- **Advanced Searching Using Boolean Charts:** allows searching based on boolean values

4. Click on the **Search** button to initiate your query
5. Once you have run a search, the **Bugzilla Bug List** page appears. You can save your search for by entering a name in the **as** field and clicking on the **Remember search** button. All saved searches are listed after the **Saved Searches:** field.

Related Concepts

Bugzilla: Defect Tracking System
Mylyn Concepts

Related Tasks

Managing Bug Reports in Bugzilla
Logging in to TeamInsight Bugzilla
Creating or Generating Bug Reports in Bugzilla
Managing Bug Reports in Bugzilla
Adding Mylyn Repositories for Bugzilla and XPlanner

Related Reference

Bugzilla Resources and Documents
The Bugzilla Guide

Tracking Your Time and Completing Tasks in XPlanner

To examine your tasks in XPlanner:

1 Enter XPlanner either by selecting **Window** ▸ **Open TeamInsight Viewer** ▸ **XPlanner** or by selecting **Window** ▸ **Add Mylyn Repository** ▸ **XPlanner**

2 Do either of the following:

- Click the **Me** command, available in the upper right corner of most pages in XPlanner, to display the **Person** page. Your **Person** page lists all your planned and completed tasks, as well as the user stories where you are the customer or tracker. On your **Person** page, you can edit the content of your tasks and record the time you have devoted to tasks. To delete or move tasks, however, you must first click on the task name to open the **Task** page.
- Navigate to your project. Then from the **Project** page, navigate to the relevant iteration, to the user story, and finally the **Task** page. On the Task page, you can manage your tasks as described in the following procedure.

To manage tasks (Edit, Delete, Move/Continue, Edit Time, Export):

1 Navigate from the **Project** page to the **Iteration** page, to the **Story** page, and finally to the **Task** page.

2 On the **Task** page, you can perform several actions:

- **Edit** opens the **Edit Task window** in which you can add or change details about the task.
- **Delete** deletes the task from the story and project.
- **Move/Continue** allows you to select the destination and then move the task to another story or iteration.
- **Edit Time** displays **Start Time** and **End Time** fields, as well as **Duration** and **Person** fields. Enter time you have spent on the task by using either **Duration** or a combination of **Start Time** and **End Time**.
- **Export** exports the task as a PDF or as a JRPDF.
- **History** displays the current XPlanner hierarchy, from project to story, and task.
- **Print** prints the task.

To enter and track time devoted to tasks

1 Navigate through XPlanner from the **Project** page to the **Iteration** page to the **Story** page and then to the **Task** page.

2 On the **Task** page, enter the time you have spent on the task by doing **either** of the following:

- In **Duration**, enter the number of hours spent on the task, such as 32 or 2.5.
- In **Start Time** and **End Time**, enter the time of day when you started and ended work on the task. Use the format YYYY-MM-DD HH:MM. Click **Enter Time** to automatically enter the current time in either of these fields.

3 If you return to the **Story** page, you will see the time decremented on the progress field of the task.

To complete a task in XPlanner

1 On the **Task** page, verify that all the hours spent on the task have been entered.

2 Click the **Complete Task** button.

3 Navigate to the **Story** page. The **Progress** field should be filled with a different color from that used for tasks still in progress.

Related Concepts

XPlanner: Project and Team Management
Mylyn Concepts

Related Tasks

Adding Mylyn Repositories for Bugzilla and XPlanner
Planning a Product Feature: Creating a User Story in XPlanner
Planning Your Work: Creating Tasks in XPlanner
Moving or Continuing a Story or Task in XPlanner

Using Continuum/Maven for Continuous Integration Builds

As part of the ProjectAssist install, Continuum is installed on a server or assimilated from a previously existing installation. Continuum allows for continuous builds during the software development cycle. By default, two build definitions are automatically configured by ProjectAssist when the Continuum component is installed. One build definition runs hourly and does a clean and install. The other build definition runs once a day. This daily build performs the more lengthy site generation, which includes running reports.

The Continuum administrator can add users, change user passwords and perform other administrative tasks. To most users, the continuous build process appears seamless. They only need to go to the Continuum server if they wanted to force an immediate build.

Note: Only the Continuum administrator can change user passwords. Users cannot change their own passwords in Continuum.

To schedule additional builds (administrator):

1. Go to the Continuum component from either the TeamInsight Viewer or through your web browser directly.
2. Login in with your Continuum administrator username and password.
3. Click **Submit** to authenticate your login.
4. The **Continuum Projects** page opens. The portal displays project information about all projects. More information about the project can be obtained by clicking on the project link. A list on the left-hand side of the page links to **Continuum** information, **Add Project** tasks, **Administration** tasks and a **Legend** displaying the meanings of the various icons.
5. In the **Administration** task section, click on **Schedules**. This brings up the **Schedules** page, which lists the schedules installed with the Continuum server component. You can edit these schedules by clicking on the edit icon on the right of the schedule. To add a new schedule, click **Add** and complete the requested information.

To perform other administrative tasks:

1. From links on the **Continuum Projects** page, the Continuum administrator can edit the general configuration information on the **General Configuration** page, manage user groups rights and privileges on the **Group Management** page, and add/edit users and user passwords on the **User Management** page.
2. The **Continuum Projects** page has several **Add Projects** links that allow the administrator to add new projects according to project type (Maven 2.0, Maven 1.x , Ant and Shell). However, the ProjectAssist Continuum component currently supports only Maven 2.0 projects.

To force an immediate build:

1. From the **Continuum Projects** page, a build can be forced immediately on any listed project.
2. With all current project listed, go to the icons on the right-hand side next to the project name.
3. Click the **Build Now** icon to generate an immediate build of the code. Refer to the **Legend** area on the left-side of the **Continuum Projects** page if you want to know the meanings of the various icons.

Refer to the following documentation links for more information on Continuum and Maven.

Related Concepts

ProjectAssist and TeamInsight Overview
Continuum/Maven: Continuous Build System
Subversion: Source Code Repository
CVS: Source Code Repository
StarTeam: Source Code Repository, Change Request Tracking, and Task Provider

Related Tasks

Checking Out a Project, Making Changes, and Checking Your Changes Into the Repository
Using the Subversion Viewer for Browsing the Project Repository

Related Reference

Continuum Online Resources and Documents
Maven Online Resources and Documents

Using the Subversion Viewer for Browsing the Project Repository

TeamInsight provides the Sventon read-only browser for viewing the Subversion repository. This topic describes how to use the Subversion Viewer (Sventon) to:

- View the Subversion repository
- Download a file from the repository
- Flatten the directory
- Display the log of changes or the current file locks
- Diff a selected file to the previous version

To open the Subversion Viewer and browse the repository:

1. Click **Window** ▸ **Open TeamInsight View** ▸ **Subversion Viewer**.
2. On the TeamInsight portal, navigate to the **Subversion** browser.

   The Subversion Viewer displays the directory containing the central repository for your development project.

Note: To check out a local copy of the repository into your workspace, click **Project** ▸ **Checkout Project** and select your project.

Note: To open the **SVN Repository** view for browsing the Subversion repository, click **Window** ▸ **Open Perspective** ▸ **Other** ▸ **SVN Repository Exploring**.

To download a file from the repository:

1. On the **Subversion Viewer**, navigate through the tree structure and open the file you want to download.
2. On the **Show file** window, click **Download**.
3. On the **File Download** dialog box, click **Save**.
4. On the **Save As** dialog box, locate the directory to contain the copy of the file and click **OK**.

Note: Downloading a file from the Subversion Viewer does not place the file in your JBuilder 2008 workspace. To check out a local copy of the repository into your workspace, click **Project** ▸ **Checkout Project**.

To flatten the directory:

1. On the **Subversion Viewer**, click **Flatten dir**. The viewer flattens the directory by displaying the repository as if all files were in one directory.

   Note: In a large project, flattening the directory can take time and files might be difficult to locate.

2. To return the browser view to its original nested status, click **go!** on **Go to path**.

To display the log of changes or the current file locks:

1. On the **Subversion Viewer**, click **Show log** or **Show locks**.

   The viewer displays the log of changes to the repository or the list of current file locks.

2 To return to the browser view, click **Show directory**.

To diff files in the repository:

1 On the **Subversion Viewer**, navigate through the tree structure and double-click the file for which you want to display historical differences.

2 On the **Show file** window, click **Diff to previous**.

The viewer displays a table listing the differences between the current and the previous versions of the file.

3 To return to the browser view, click **Show directory**.

Related Concepts

Subversion: Source Code Repository

Related Tasks

Configuring Your TeamInsight Client
Checking Out a Project, Making Changes, and Checking Your Changes Into the Repository

Peer to Peer Collaboration

The JBuilder 2008 peer to peer subsystem allows you to collaborate with peers on the same local area network (LAN) as you are. You can chat with peers and share data with peers. You can also share projects through a repository.

In This Section

Chatting with Peers

To chat with peers:

1 Open the Peers view (**Window** ▶ **Show View** ▶ **Other** ▶ **Peer to Peer** ▶ **Peers**) and set your status to Available.

2 Double-click the name of the peer or contact group you want to chat with or use multiple selection of peers, right-click and select **Open Session**.

The Collaboration pane is opened on the right of the Peers view. The connection is displayed in the chat area. The chat is recorded on your machine, if chat logging is enabled.

3 Type a message into the text field at the bottom of the Collaboration pane.

4 Press ENTER to send the message.

The message is displayed in the chat area of the Collaboration pane, both on your machine and on the peer's machine(s).

To view and delete the chat log:

1 In the Peers pane on the left of the Peers view, select the name of the peer with whom you have chatted.

Note: Each member of the collaboration has a copy of the chat session in the member's individual log.

2 To view the chat log, right-click and choose **View Chat Log**.

The chat log is displayed in the editor as a text file. It is UTF-8 encoded.

3 To delete the chat log, right-click and choose **Delete Chat Log**.

The chat log is deleted for that peer.

You set the chat log file location on the **Peer to Peer** page of the **Preferences** dialog box (**Window** ▶ **Preferences** ▶ **Peer to Peer**).

Related Concepts

Peer to Peer Collaboration

Related Tasks

Setting Collaboration Preferences
Enabling Peer to Peer Collaboration
Opening a Peer to Peer Session
Sharing Team-Enabled Projects with Peers
Sending Data To Peers
Managing Contact Groups

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Enabling Peer to Peer Collaboration

To use the peer to peer subsystem, you need to enable it and set up your identity. As you work, you can change your status from Available to Away or Offline.

To enable collaboration and create your identity:

1 Open the **Peer to Peer** page of the **Preferences** dialog box (**Window** ▸ **Preferences** ▸ **Peer to Peer**).

2 Check the **Enable Peer To Peer Subsystem** option.

3 Enter your user name in the **Name** field. This defaults to your user logon.

Note: Your user name is displayed in the Peers pane on your peers' machines.

4 Enter an optional description in the **Description** field. This description can help identify you to peers.

Note: The description is displayed in a tooltip in the Peers pane on your peers' machines.

5 Enter the name of an image file in the **Image** field. The image helps identify you to other peers in a collaboration session. You can use the **Browse** button to browse to the image file location.

Note: The image is displayed in a tooltip in the Peers pane on your peers' machines. Any icon you use is automatically resized to 48 x 48 pixels. The image may be distorted if resized.

6 Click **Apply** and **OK** to apply and save your identity settings.

The peer to peer subsystem is enabled and the Peers view is opened, with your status set to Available. You will see any other peers that are available on your LAN. Peers should be able to see you as an available peer.

To set your status

1 Open the **Status** drop-down list. The drop-down list box is located at the top of the Peers pane on the left side of the Peers view.

2 Choose a status from the list.

- Available — You are available for collaboration. Your name, description, status, selected image, and IP address are displayed in the Peers list on your peers' computers.
- Offline — You are offline. This terminates the active session, terminates the LAN connection, and removes your name from the Peers list on your peers' computers.
- Away — You are away from your desk. This status is displayed next to your name in the Peers list on your peers' computers.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Setting Collaboration Preferences
Opening a Peer to Peer Session
Managing Contact Groups
Chatting with Peers
Sharing Team-Enabled Projects with Peers
Sending Data To Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Managing Contact Groups

A contact group is a group of peers. You manage contact groups in the Peers pane.

To add a contact group:

1. Right-click the Peers pane and choose Add Contact Group.
   The **New Contact Group** dialog box is displayed.
2. Enter the name of the group in the **Group Name** field and click **OK**.
   The name of the group is added to the **Contact Groups** list in the Peers pane.

To add peers to a contact group:

1. In the **Available Local Peers** list of the Peers pane, right-click the name of the peer you want to add to the group.

   Tip: You can select more than one peer at a time to add to a group.

2. Choose Add Peer(s) To Contact Group.
3. Choose the group from the drop-down menu.
   The peer is added to the selected group and displayed in the **Contact Groups** list in the Peers pane.

To remove a peer from a contact group:

1. Open the group in the **Contact Groups** list in the Peers pane.
2. Right-click the name of the peer to remove from the group.

   Tip: You can select more than one peer at a time to remove from a group.

3. Choose Remove Peer(s) From Contact Group.

To remove a contact group

1. Right-click the name of the group you want to remove.
2. Choose Remove Contact Group(s).

   Tip: You can select more than one group at a time to remove.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Setting Collaboration Preferences
Enabling Peer to Peer Collaboration
Opening a Peer to Peer Session
Chatting with Peers
Sending Data To Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Opening a Peer to Peer Session

To open a session with a peer or contact group:

1. If the Peers view is not already open, open it with the **Window ▸ Show View ▸ Other ▸ Peer to Peer ▸ Peers** command.
2. Right-click a peer or contact group you want to collaborate with in the Peers pane.
3. Choose **Open Session** to open the session.

> **Note:** You can also double-click the peer's name to open a session. You can multi-select peers to open a session with more than one peer.

A tab is added to the Collaboration pane on the right side of the Peers view. The Collaboration pane displays the peer or peers with whom you are connected and the chat area. Once you start a chat or send a file, the Collaboration pane tab on the peers' machine opens and displays information.

Tip: You can also drag a file from the **Package Explorer** or the **Navigator** directly only to the peer or contact group name in the Peers pane. A chat session is opened if one is not already open.

To close a session with a peer or contact group:

1. Click the **X** on the tab of the session you wish to close in the Collaboration pane.
2. A message indicating that you have left the session is displayed in the Collaboration pane on the peer's machine.

To close all sessions:

1. Click the **Close All Collaboration Sessions** button (the **X**) on the Collaboration pane toolbar.
2. A message indicating that you have left the session is displayed in the Collaboration pane on all peer machines. All sessions are closed and the Collaboration pane on your machine is closed.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Setting Collaboration Preferences
Enabling Peer to Peer Collaboration
Sharing Team-Enabled Projects with Peers
Managing Contact Groups
Chatting with Peers
Sending Data To Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Sending Data To Peers

You can send files, diagrams, web links, and stack traces to peers in a chat session.

To send a file or diagram in a chat session:

1. Open a session with a peer or contact group.
2. Click the **Send Files To Peers In Collaboration** button on the **Collaboration** pane toolbar.
3. Browse to the file or diagram you want to send in the **Open** dialog box and click **Open**.

   The file is sent to the peer(s) in the chat session. A message appears in the peer's chat area and the file name is automatically downloaded to the folder specified in the **Workspace Folder** field in the **File Transfer** area on the **Peer to Peer** page of the **Preferences** dialog box (**Window** ▸ **Preferences** ▸ **Peer to Peer**).

   Note: If the **Automatic Receive Enabled** option is off on the **Peer to Peer** page of the **Preferences** dialog box, the file is displayed as a link in the chat area. You need to click the link to open a **Save As** dialog box and save the file.

Tip: You can also drag a file or diagram from the **Unified Navigator**, **Package Explorer**, or **Navigator** directly only to the peer or contact group in the Peers pane. A chat is opened and the file is sent.

To send a line of text from an external file:

1. Open a session with a peer or contact group.
2. Open the application and file that you want to send text from.
3. Select the text in the file and drag it to the chat area for an open session or drop it on the peer in the Peers pane.

   Tip: Dragging text deletes it from the original file unless you hold down the CTRL key.

4. Press ENTER to send the line of text.

   The text is sent to the peer(s) in the chat session as a message.

To send a web link in a chat session:

1. Open a session with a peer or contact group.
2. Click the **Send Web Link To Peers In Collaboration** button on the **Collaboration** pane toolbar.
3. Enter the URL in the **Send Web Link** dialog box and click **OK**.

   The URL is sent to the peer(s) in the chat session. A message appears in the chat area and the URL is displayed as a link.

4. Click the URL to open the link in a web browser. Your peer(s) can do the same.

To send a stack trace in a chat session

1. Copy the contents of a stack trace into the Clipboard.
2. Open a session with a peer or contact group.
3. Click the **Send Stack Trace To Peers In Collaboration** button on the **Collaboration** pane toolbar.

4 Paste the stack trace into the **Stack Trace** dialog box and choose **Send**.

The stack trace is sent to the peer(s) in the chat session. A message appears in the chat area and the stack trace is displayed as a link.

5 Click the link to open the stack trace in the **Console** view using the Java Stack Trace Console. Your peer(s) can do the same.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Setting Collaboration Preferences
Enabling Peer to Peer Collaboration
Opening a Peer to Peer Session
Managing Contact Groups
Chatting with Peers
Sharing Team-Enabled Projects with Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Setting Collaboration Preferences

Most collaboration preferences are set on the **Peer to Peer** page of the **Preferences** dialog box. If you modify settings during a open session, you may be prompted to restart your connection to apply the changes.

To set filtering for multiple adapters:

1. Open the **Peer to Peer** page of the **Preferences** dialog box (**Window** ▸ **Preferences** ▸ **Peer to Peer**).

   Note: You can also right-click the Collaboration pane and choose **Preferences** to display the **Preferences** dialog box.

2. Choose the adapter you want to use in the **Filtering** drop-down list, or choose NONE to not use an adapter.
3. Click **Apply** to apply the settings. Click **OK** if you're done.

To set chat preferences:

1. Open the **Peer to Peer** page of the **Preferences** dialog box (**Window** ▸ **Preferences** ▸ **Peer to Peer**).
2. Select the **Log Chat Messages** option to turn on logging for chat messages.

   Note: The chat log is UTF-8 encoded.

3. Enter the name of the directory where you want messages saved in the **Workspace Directory** field.
4. Click the **Incoming Message Color** box to set the color for incoming messages.
5. Click the **Outgoing Message Color** box to set the color for outgoing messages.
6. Click the **Status Message Color** box to set the color for status messages.
7. Click **Apply** to apply the settings. Click **OK** if you're done.

To automatically transfer files into the workspace:

1. Select the **Automatic Receive Enabled** option to automatically transfer files when you're chatting with peers.

   Note: If you turn this option off, you will need to click a link to the file when you receive it in order to download it into a directory. You must be in an active chat session.

2. Enter the name of the directory you want files automatically downloaded to in the **Workspace Directory** field.
3. Click **Apply** to apply the settings. Click **OK** if you're done.

To set audio feedback:

1. Select the **Audio Feedback Enabled** option.
2. Adjust the volume slider.
3. Click **Apply** and **OK** to apply and save the preferences.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Enabling Peer to Peer Collaboration
Opening a Peer to Peer Session
Managing Contact Groups
Chatting with Peers
Sharing Team-Enabled Projects with Peers
Sending Data To Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Sharing Team-Enabled Projects with Peers

Projects are shared through a repository. When projects are shared, the **Navigator** or **Package Explorer** displays the project repository and location.

To share projects with a peer:

1. Check out the project from the repository.
2. Right-click the project and choose **Send VCS Link to Peer.**

   The **Select Peer** dialog box is displayed.
3. Choose the name of the peer you want to send the link to and click **Select**.

   The project is sent as a VCS link to the selected peer. The message **Sending VCS link for project "<Project Name>"** is displayed in your chat area.

Note: To send the link to more than one peer, you need to choose the **Send VCS Link to Peer** command for each peer.

To receive a link to a shared project:

1. Click the VCS link you received in the chat area.
2. Log onto the server if you are not already logged on.
3. Navigate the VCS check out dialog boxes.

   The project is checked out locally. The **Navigator** or **Package Explorer** displays the project repository and location.

Note: JBuilder 2008 and JBuilder 2006 projects are not compatible for the project sharing feature.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Setting Collaboration Preferences
Enabling Peer to Peer Collaboration
Managing Contact Groups
Chatting with Peers
Sending Data To Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Reference

IDE Reference

This section lists all of the dialog/wizards information provided through your CodeGear software.

In This Section

JBuilder or JGear Perspectives

This section lists some of the additional perspectives provided with various JBuilder or JGear feature sets. A Perspective is available through the Workbench and defines and controls the views, editors and actions within a window.

The JBuilder or JGear products add several perspective depending upon the feature set:

In This Section

Application Factory Producer Perspective

Use one of the following paths to open the **Application Factory Producer** perspective in your workspace.

Window ▸ Open Perspective ▸ Other ▸ Application Factory Producer

Open Perspective icon (upper-right of workspace by default) ▸ ▸ Other ▸ Application Factory Producer

Use the **Application Factory Producer** perspective in your workspace to focus on functionality for producers of Application Factory modules. Once the perspective has been opened you can switch between it and other open perspectives using the appropriate icons next to the **Open Perspective** toolbar icon. By using the context menu from the associated perspective icon, you can customize, save, reset, or close that perspective. You can also change the docking location for all the perpective toolbar icons. You can surface additional views in this perspective by using the **Window ▸ Show View ▸ Other** and selecting from or filtering on views from the provided list.

Default Views in the Application Factory Producer Perspective	Description
Package Explorer	Contains the open **Package Explorer** view for file browsing.
Hierarchy	Contains the hiearchy of the element.
Scripts — Application Factory	Shows the scripts view.
Outline	
Tags	
Problems	Contains details of any problems, warnings, errors or other information.
Javadoc	Contains any available Javadoc.
Declaration	Contains any code declarations.
Commit History	Contains the commit history view for the item

You can customize the default **Application Factory Producer** perspective to your specific requirements by right-clicking on the perspective icon and choosing **Customize**.

Right-click on the workspace icon for this perspective and select **Reset** to reset the perspective to its defaults.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Application Factory Repository Exploring Perspective

Use one of the following paths to open the **Application Factory Repository Exploring** perspective in your workspace.

Window ▸ Open Perspective ▸ Other ▸ Application Factory Repository Exploring

Open Perspective icon (upper-right of workspace by default) ▸ ▸ Other ▸ Application Factory Repository Exploring

Use the **Application Factory Repository Exploring** perspective in your workspace to focus on functionality for consumers of Application Factory projects and modules. Once the perspective has been opened you can switch between it and other open perspectives using the appropriate icons next to the **Open Perspective** toolbar icon. By using the context menu from the associated perspective icon, you can customize, save, reset, or close that perspective. You can also change the docking location for all the perpective toolbar icons. You can surface additional views in this perspective by using the **Window ▸ Show View ▸ Other** and selecting from or filtering on views from the provided list.

Default Views in the Application Factory Repository Exploring Perspective	Description
Application Factory Explorer	Contains the data-aware module types that can be created.
Package Explorer	Contains the open **Package Explorer** view for file browsing.
Hierarchy	Contains the hiearchy of the element.
Scripts — Application Factory	Shows the scripts view.
Problems	Contains details of any problems, warnings, errors or other information.
Javadoc	Contains any available Javadoc.
Declaration	Contains any code declarations.

You can customize the default **Application Factory Repository Exploring** perspective to your specific requirements by right-clicking on the perspective icon and choosing **Customize**.

Right-click on the workspace icon for this perspective and select **Reset** to reset the perspective to its defaults.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Application Factory Modeling Perspective

Use one of the following paths to open the **Application Factory Modeling** perspective in your workspace.

Window ▸ Open Perspective ▸ Other ▸ Application Factory Modeling

Open Perspective icon (upper-right of workspace by default) ▸ ▸ Other ▸ Application Factory Modeling

Default Views in the Application Factory Modeling Perspective	Description
Model Navigator	Contains the tree for the Application Factory project.
Package Explorer	Contains the open **Package Explorer** view for file browsing.
Navigator	contains the open **Navigator** view for file navigation
Application Diagram	Shows the **Application Diagram** view.
Scripts — Application Factory	Shows the scripts view.
Tags	Shows a view of all tags and the weight of each tag.
Properties	Contains any available properties.
Data Source Explorer	Contains the explorer for data sources.
Servers	Contains a list of servers
Problems	List any problems that have occurred.

You can customize the default **Application Factory Modeling** perspective to your specific requirements by right-clicking on the perspective icon and choosing **Customize**.

Right-click on the workspace icon for this perspective and select **Reset** to reset the perspective to its defaults.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

ProjectAssist Perspective

Use one of the following paths to open the ProjectAssist perspective in your workspace.

Window ▸ Open Perspective ▸ Other ▸ ProjectAssist ▸

Open Perspective icon (upper-right of workspace by default) ▸ ▸ Other ▸ ProjectAssist

Use the **ProjectAssist** perspective in your workspace to focus on ProjectAssist and TeamInsight project functionality. Once the perspective has been opened you can switch between it and other open perspectives using the appropriate icons next to the **Open Perspective** toolbar icon. By using the context menu from the associated perspective icon, you can customize, save, reset, or close that perspective. You can also change the docking location for all the perpective toolbar icons.You can surface additional views in this perspective by using the **Window ▸ Show View ▸ Other** and selecting from or filtering on views from the provided list.

Default Views in the ProjectAssist Perspective	Description
Navigator	Contains the open **Navigator** explorer view for file browsing.
.pacx file	Contains the configured ProjectAssist file (.pacx) with Stacks, Users, Projects and Source tabs.
Problems	Contains details of any problems, warnings, errors or other information.

You can customize the default **ProjectAssist** perspective to your specific requirements by right-clicking on the perspective icon and choosing **Customize**.

Right-click on the workspace icon for this perspective and select **Reset** to reset the perspective to its defaults.

Related Concepts

ProjectAssist and TeamInsight Concepts

Related Tasks

TeamInsight Procedures

Related Reference

ProjectAssist and TeamInsight Dialogs

Project Import Dialogs

This section describes the dialogs/wizards information for importing legacy projects into JBuilder on Eclipse projects.

In This Section

Java EE Project Import from Legacy JBuilder
Imports a Java EE project from a legacy JBuilder. jpx file

Java Project Import from Legacy JBuilder
Imports a Java project from a legacy JBuilder. jpx file

Java EE Project Import from Legacy JBuilder

File ▸ New ▸ Other ▸ Legacy JBuilder ▸ Java EE Project from Existing JBuilder .jpx Project

Use this dialog box to import a legacy JBuilder project into a Java EE project for JBuilder on Eclipse.

Item	Description
JBuilder project file	The name and path to an legacy JBuilder project. The file extension must be `.jpx`.
Browse	Displays the **Open** dialog box where you can browse to the legacy JBuilder project file.
Project name	The name of a valid legacy JBuilder project. It is filled in automatically when the project name is selected.
User home	The default user home directory. This is based on the default installation directory for JBuilder and defines the default search directory for libraries. You can browse to any directory desired.
Directories to search for missing libraries	The directories to search for JBuilder libraries for the selected project.
Add	Displays the **Browse for Folder** dialog box, where you can browse to the library search directory you want to add to the search list.
Remove	Removes the selected directory from the search list.
Libraries Not Yet Found	Libraries required by the selected project but not yet located. Locate the library directory for each library and add it to the **Library Search Directories** list.
Finish	Imports or checks out the project.

Related Concepts

Legacy JBuilder Project Migration Overview
Migrating from Legacy Versions of JBuilder

Related Tasks

JBuilder Project Migration

Java Project Import from Legacy JBuilder

File ▶ New ▶ Other ▶ Legacy JBuilder ▶ Java Project From Existing JBuilder.jpx Project

Use this dialog box to import a project from legacy JBuilder into a Java project for JBuilder on Eclipse project.

Item	Description
JBuilder project file	The name and path to an legacy JBuilder project. The file extension must be `.jpx`.
Browse	Displays the **Open** dialog box where you can browse to the legacy JBuilder project file.
Project name	The name of a valid legacy JBuilder project. Filled in automatically when the project name is selected.
Enable VCS plugin for this project	Checks out the project into the JBuilder on Eclipse workspace. If the project is under source control in JBuilder and you want the project checked out, you may need to log onto the server to check out the project.
User home	The default user home directory. This is based on the default installation directory for JBuilder and defines the default search directory for libraries. You can browse to any directory desired.
Directories to search for missing libraries	The directories to search for JBuilder libraries for the selected project.
Add	Displays the **Browse for Folder** dialog box, where you can browse to the library search directory you want to add to the search list.
Remove	Removes the selected directory from the search list.
Libraries Not Yet Found	Libraries required by the selected project but not yet located. Locate the library directory for each library and add it to the **Library Search Directories** list.
Finish	Imports or checks out the project.

Related Concepts

Legacy JBuilder Project Migration Overview
Migrating from Legacy Versions of JBuilder

Related Tasks

JBuilder Project Migration

Application Factory Dialogs and Preferences Reference

This section lists all of the dialog/wizards and preferences information provided for the Application Factory functionality of your JBuilder or JGear product.

In This Section

Application Factory Preferences

Window ▸ Preferences ▸ Application Factory

Use the **Application Factory Preferences** dialog to define the general preferences for the Application Factory project .

Item	Description
Show information dialog when launch script	Check to show an information dialog when a script is launched.
Warn of unresolved changes when launch script	Check to receive warnings of unresolved changes when a script is launched.
Script behavior when commiting file changes	Select the **Ask before commit** button if you want to be asked prior to any script changes being committed. Select the **Show changes before commit** button if you want to view any script changes prior to those changes being committed. Select the **Commit automatically (only if no errors)** button if you want to commit any script changes automatically (this occurs only if there are no errors).

Click **Apply** to go to apply all changes.

Click **Restore Defaults** to return to the default selections.

Click **OK** to exit the Preferences dialog.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Data-Aware Web Application Settings Preferences

Window ▸ Preferences ▸ Application Factory ▸ Data-Aware Web Application Settings

Use the **Data-Aware Web Application Settings Preferences** dialog to set your preferred mail and CRUD settings.

Item	Description
Mail Settings	Provides settings for mail delivery. This can also be set after application creation through the IDE path **Window ▸ Prefererences ▸ Application Factory ▸ Data-Aware Web Application Settings**. Complete the following information: ■ **Mail from**—the mail address from which to send messages. ■ **Transport protocol**—the type of mail transport protocol used to send mail messages, usually SMTP (simple mail transfer protocool) ■ **Host**—the name of the host mail server. ■ **User name**—the user name for mail access. ■ **Password**—the password for the mail access user name.
CRUD settings	Allows you to select or deselect the **Use Generic Manager classes** option. This option is checked by default.

Click **OK** to enter your preferences settings.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory
Using Scripts

Related Reference

Application Factory Dialogs and Preferences Reference

Identity Preferences

Window ▸ Preferences ▸ Application Factory ▸ Identity

Use the **Identity Preferences** dialog to change your unique identifier.

Item	Description
Your unique identifier	Displays your current identifier and allows you to change the identifier.

Click **Apply** to apply any changes.

Click **Restore Defaults** to return to the default selections.

Click **OK** to apply changes.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Module Search/Export Directories Preferences

Window ▸ Preferences ▸ Application Factory ▸ Module Search/Export Directories

Use the **Module Search/Export Directories** preferences page to choose or specify the directories that are searched for modules or to whichApplication Factory modules are exported.

Item	Description
Directories	Lists all the directories to search for application modules when searching for modules to import or lists the directories to which modules are exported (published) . Click the **Add Directory** button to specify a new directory to search/export. Click **Remove** to remove any unwanted directories.
Feeds	Lists the RSS/Atom feeds that can be used as a location from which you can import a module. This RSS/Atom feed file is then included on the list of importable application modules shown to the consumer in the **Application Factory Explorer** or in the **Import Module** wizard. Click the **Add Feed** button to specify a new feed file to make available as a location from which you can import a module. The**Specify RSS Feed URL** dialog opens. In **Feed Url**field of this dialog, enter the RSS/Atom URL feed address that was specified when you published the module (in the **Module URL for feed** field on page 2 of the **Export Application Module** wizard). Click **Edit** to edit the feed name. Click **Remove** to remove any unwanted items. Click **Test** to verify that the specified feed location is valid.

Click **Apply** to apply any changes.

Click **Restore Defaults** to return to the default selections.

Click **OK** to apply changes.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Template Appearance Preferences

Window ▸ Preferences ▸ Application Factory ▸ Template Appearance

Use the **Template Appearance Preferences** dialog to define the templates appearance.

Item	Description
Directive	Selects the color highlighting for directives in the template.
Interpolation	Selects the color highlighting for areas of the template where sections are replaced with a calculated value in the output.
Text	Selects the color highlighting for text areas of the output.
Comment	Selects the color highlighting for comments in the template.
String	Selects the color highlighting for any strings in the template.
HTML/XML Highlighting	Selects whether HTML/XML highlight is done in the template. This box is checked by default.
HTML/XML Tag	Selects the color highlighting for HTML/XML tags in the template.
HTML/XML Comment	Selects the color highlighting for HTML/XML comments in the template.

Click **Apply** to apply any changes.

Click **Restore Defaults** to return to the default selections.

Click **OK** to apply changes.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

New Application Factory Project Dialog

File ▸ New ▸ Other ▸ Application Factory ▸ Application Factory Project ▸ Next

Use the **New Application Factory Project** dialog to create a new Application Factory project in your workspace.

Item	Description
Name	Specify the name of the Application Factory project. Application Factory is selected by default.
Import global scripts and templates	Check to import all global scripts and templates.
Open skeleton readme file	Check if you want to open a skeleton cheat sheet for this project.
Open skeleton cheat sheet file	Check if you want to open a skeleton cheat sheet for this project.

Click **Finish** to create an Application Factory project in your workspace.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Import Application Module Dialog

File ▸ Import ▸ Application Factory ▸ Import Application Module

Use the **Import Application Module** dialog to import a completed Application Factory module.

Item	Description
Application Modules	Provides a table of the Application Modules available for import. Select the module you want to import into the workspace.

Click **Finish** to import the selected project.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Export Application Module Dialog (page 1 of 2)

File ▸ Export ▸ Application Factory ▸ Export Application Module

Use page 1 of **Export Application Module** wizard to select the Application Factory projects that are to be included in the exported module archive.

Item	Description
Select projects to include in the archive	Specify the projects you want to include in the module archive (.mar) file. Individually select desired projects or use the **Select All** or **Unselect All** buttons to globally designate selections.

Click **Next** to proceed to page 2 of the **Export Application Module** wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Script Recipe for Application Factory Dialog Reference

File ▸ New ▸ Other ▸ Application Factory ▸ Script Recipe for Application Factory

Use the **Script Recipe for Application Factory** dialog to create a new Application Factory script using code archeology.

Item	Description
Name	Specify a name for your script.
Project task	Designates whether to add a project task when opening the new script recipe. You can also specify a project template.

Click **Finish** to create the specified script.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Create DOM Project

Application Factory Explorer ▸ Eclipse Monkey DOM Plugin Factory 1.0

Use the **Create DOM Project** to create an Eclipse Monkey DOM plugin project module to supplement your own script accessible API.

You need an Application Factory project in your workspace to enable this wizard.

Item	Description
Project name	Specify the DOM project name. This name appears at the top level of the directory tree in the **Package Explorer** view. The default value of this field is `MyDom`. Use this value or enter another value. If you want to create multiple projects in the same workspace using this wizard, each project must have a unique project name.
Source directory name	Specify the source directory name for the project. The default value of this field is `src`. Use this value or enter another value.
Package name	Specify the package name for the project. This default value is `mydom`. Use this value or enter another value.
DOM variable name	Specify the name for DOM variables. The default value is `mydom`. Use this value or enter another value.

Click **OK** to create the DOM plugin project. The project appears in the **Package Explorer** view list.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

PetStore Template Dialogs Reference

This section lists the dialog/wizards for module development using the PetStore Template of the Application Factory.

In This Section

Setup.js: Select Glassfish Server Runtime
Use the **Select Glassfish Server Runtime** dialog page of the **Setup.js** script to select a Glassfish Server runtime for a new PetStore application module.

Setup.js: Select Domain
Use the **Select Domain** dialog page of the **Setup.js** script to select a domain for the Glassfish server.

Setup.js: Select Database Connection
Use the **Select Database Connection** dialog page of the **Setup.js** script to select a database connection for your Pet Store application.

03_Add_Maps.js: Enter Google Maps API Key
Use the **03_Add_Maps.js: Enter Google Maps API Key** dialog page to add support for map functionality in your Pet Store module.

Add_RSS_Bar.js: Enter RSS feed XML URL
Use the **Add_RSS_Bar.js: Enter RSS feed XML URL** dialog page to add support for an RSS feed bar in your Pet Store module.

Setup.js: Select Glassfish Server Runtime

Application Factory Explorer ▸ PetStore (right-side pane) ▸ Setup.js

Use the **Select Glassfish Server Runtime** dialog page of the **Setup.js** script to select a Glassfish Server runtime for a new PetStore application module.

Item	Description
Glassfish Runtime	Specify a Glassfish runtime server either by selecting an existing server from the dropdown menu or clicking **New**, which opens a series of dialogs to install a new server runtime.
Show all runtimes	Specifies that all available runtimes are shown in the dropdown menu.

Click **OK** to open the **Select Domain** dialog page.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Setup.js: Select Domain

Application Factory Explorer ▸ PetStore (right-side pane) ▸ Setup.js ▸ Select Glassfish Server Runtime ▸ OK

Use the **Select Domain** dialog page of the **Setup.js** script to select a domain for the Glassfish server.

Item	Description
Domain	Specify the Glassfish runtime server domain.

Click **OK** to open the **Select Database Connection** dialog page.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Setup.js: Select Database Connection

Application Factory Explorer ▸ PetStore (right-side pane) ▸ Setup.js ▸ Select Glassfish Server Runtime ▸ Select Domain ▸ OK

Use the **Select Database Connection** dialog page of the **Setup.js** script to select a database connection for your Pet Store application.

Item	Description
Domain	Select the database connection for your application.

Click **OK** to finish your Pet Store application setup.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

03_Add_Maps.js: Enter Google Maps API Key

Script—Application Factory ▸ 03_Add_Maps.js

Use the **03_Add_Maps.js: Enter Google Maps API Key** dialog page to add support for map functionality in your Pet Store module.

Item	Description
Key	Specifies the Google Maps API Key.

Click **OK** to finish adding Google map unctionality to your Pet Store application.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Add_RSS_Bar.js: Enter RSS feed XML URL

Script—Application Factory ▸ Add_RSS_Bar.js

Use the **Add_RSS_Bar.js: Enter RSS feed XML URL** dialog page to add support for an RSS feed bar in your Pet Store module.

Item	Description
URL	Enter any valid RSS feed URL. This field defaults to a CodeGear blog feed.

Click **OK** to finish adding RSS feed bar functionality to your Pet Store application.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference

Script for Application Factory

This section describes the 4-page **Script for Application Factory** wizard.

In This Section

Script for Application Factory: Create Application Factory Script File (page 1 of 4)

If an Application Factory project exists in workspace: **File ▸ New ▸ Script for Application Factory**

or

File ▸ New ▸ Other ▸ Application Factory ▸ Script for Application Factory

Use the **Script for Application Factory** wizard to create anApplication Factory script in the workspace. This is page 1 of a 4-page wizard.

Note: An Application Factory project must exist in your workspace prior to using the **Script for Application Factory** wizard.

Item	Description
Script Preview	Previews the current state of the Application Factory script you are creating. Select or deselect the **Generate code to report input values**, as desired. Click **Test Script** button at any time to test your script in its current state.
Enter or select the parent folder	Contains the name of the Application Factory project in your workspace. You can also specify an alternate name in this field. A tree structure for your Application Factory project name appears in the box below this field.
File name	Specify a name for your script file.
Advanced	Click the **Advanced** button to select the **Link to file in the file system** box.
Link to file in the file system	To link to an existing file in the file system, click the **Advanced** button and check the **Link to file in the file system** box. Click the **Browse** button to browse to a file location. Click the **Variables** button to select a path variable.

Click **Next** to go to the **Define your APIs and Other Metadata** page (page 2) of this wizard to specify the metadata to include

Click **Finish** to create the specified script.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory
Creating Scripts

Related Reference

Script for Application Factory: Define your APIs and Other Metadata (page 2 of 4)
Script for Application Factory: Add a User Interface to your Script (page 3 of 4)
Script for Application Factory: Add Code to Change Workspace Files (page 4 of 4)
Application Factory Dialogs and Preferences Reference
FreeMarker Template Engine Overview

Script for Application Factory: Define your APIs and Other Metadata (page 2 of 4)

If an Application Factory project exists in workspace: **File ▸ New ▸ Script for Application Factory ▸ Create Application Factory Script File (page 1 of 4) ▸ Next**

or

File ▸ New ▸ Other ▸ Application Factory ▸ Script for Application Factory ▸ Create Application Factory Script File (page 1 of 4) ▸ Next

Use the **Script for Application Factory** wizard to create an Application Factory script in the workspace. This is page 2 of a 4-page wizard.

Item	Description
Script Preview	Shows the current state of the Application Factory script you are creating. Select or deselect the **Generate code to report input values**, as desired and depending on whether you want feedback that your UI is delivering what you expect from your input when you test it. Click **Test Script** button at any time to test your script in its current state.
Author	Specifies the name of the script author.
Description	Describes the script to be generated. This is useful information that is displayed several ways. For example: it is used as a tooltip in the Scripts — Application Factory view.
Installed DOMs	Includes a list of the DOMs that are installed, with check boxes. Check or uncheck as desired to include with your script. As you check or uncheck boxes, the DOMs selected will show in your **Script Preview** window. Click **Install New DOM**. This opens the **Open Update Manager** dialog where you can specify the URL for an update site to install an additional DOM. The **Variables** column of the **Installed DOMs** table shows you the names of variables that are available to your script at runtime when selecting that particular DOM. All these variables are references to classes that each surface and APT that you can easily access from your script.

Click **Next** to go to the **Add an User Interface to your Script** page of this wizard to specify code to generate for user input in your script.

Click **Finish** to create the specified script.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory
Creating Scripts

Related Reference

Script for Application Factory: Create Application Factory Script File (page 1 of 4)
Script for Application Factory: Add a User Interface to your Script (page 3 of 4)
Script for Application Factory: Add Code to Change Workspace Files (page 4 of 4)
Application Factory Dialogs and Preferences Reference
FreeMarker Template Engine Overview

Script for Application Factory: Add a User Interface to your Script (page 3 of 4)

File ▸ New ▸ Script for Application Factory ▸ Create Application Factory Script File (page 1 of 4) ▸ Define your APIs and Other Metadata (page 2 of 4) ▸ Next

or

File ▸ New ▸ Other ▸ Application Factory ▸ Script for Application Factory ▸ Create Application Factory Script File (page 1 of 4) ▸ Define your APIs and Other Metadata (page 2 of 4) ▸ Next

Use the **Script for Application Factory** wizard to create anApplication Factory script in the workspace. This is page 3 of a 4-page wizard.

Item	Description
Script Preview	Shows the current state of theApplication Factory script you are creating. Select or deselect the **Generate code to report input values** , as desired. Click **Test Script** button at any time to test your script in its current state.
Name	Selects the name field to be included in your UI.
Title	Selects the dialog title to appear in bold in the title area of the dialog box.
Description	Selects the dialog short description to appear under the title.
UI Elements	Shows the selected UI elements. You can add new UI elements by clicking **New**. You can perform this action multiple times. This opens the **New UI Element** dialog. Select from Workspace resources, file system resources, data entry and selection resource, and template resources. (you must specify the template file path if you select template .resources). Click **OK**. The code for the UI elements selected shows in your **Script Preview** window. When selecting each variable in the UI elements list, different fields for control-specific properties appear below the list: Each field contains a default value. The default value can be used or changed, as desired.

Click **Next** to go to the **Add Code to Change Workspace Files** page of this wizard to generate code that can create/delete files and change text in existing files.

Click **Finish** to create your script.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory
Creating Scripts

Related Reference

Script for Application Factory: Create Application Factory Script File (page 1 of 4)
Script for Application Factory: Define your APIs and Other Metadata (page 2 of 4)
Script for Application Factory: Add Code to Change Workspace Files (page 4 of 4)
Application Factory Dialogs and Preferences Reference
FreeMarker Template Engine Overview

Script for Application Factory: Add Code to Change Workspace Files (page 4 of 4)

File ▸ New ▸ Script for Application Factory ▸ Create Application Factory Script File (page 1 of 4) ▸ Define your APIs and Other Metadata (page 2 of 4) ▸ Add an User Interface to your Script (page 3 of 4) ▸ Next

or

File ▸ New ▸ Other ▸ Application Factory ▸ Script for Application Factory ▸ Create Application Factory Script File (page 1 of 4) ▸ Define your APIs and Other Metadata (page 2 of 4) ▸ Add a User Interface to your Script (page 3 of 4) ▸ Next

Use the **Script for Application Factory** wizard to create anApplication Factory script in the workspace. This is page 4 of a 4–page wizard.

Script Preview	Shows the current state of theApplication Factory script you are creating. Select or deselect the **Generate code to report input values**, as desired. Click **Test Script** button at any time to test your script in its current state.
Code Snippets	Click on **New** to add a new code snippet.
Operation type	Choose the type of operation from the dropdown menu to create a new file, modifyand existing file, or delete a file.
File change description	Describes the change you are making to the file. This is displayed in the Script Learning/Resolve/Commit and Archeology views.
Tags to apply	Click on **...** to select tags to apply to the file that this code snippet modifies or creates.
Select project	Select and enter (or browse to) the **Project workspace name** or select the **Project reference variable**.
Select project file	Select and enter (or browse to) the **Project-relative path** or select the **Project file reference variable**.
Configure insert/replace indicator	Specifies the expression where to find a location in the file and then the operation to perform, relative to that location.
Select source	Specifies whether the source of the input text is: ■ a string (that you insert in the text area). If it is a string, you can check **Treat string as template**. ■ a template file (for which you can specify a name or browse to for selection).

Click **Finish** to create the specified code in the script.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory
Creating Scripts

Related Reference

Script for Application Factory: Create Application Factory Script File (page 1 of 4)
Script for Application Factory: Define your APIs and Other Metadata (page 2 of 4)
Script for Application Factory: Add a User Interface to your Script (page 3 of 4)
Application Factory Dialogs and Preferences Reference
FreeMarker Template Engine Overview

Creating Script for Task

This section describes the 2-page Script for Creating Script for Task.

In This Section

Creating Script for Task: Add a User Interface to your Script (page 1 of 2)

File ▸ New ▸ Script for Application Factory ▸ Create Script for Task

or

File ▸ New ▸ Other ▸ Application Factory ▸ Script for Application Factory ▸ Create Script for Task

Use the 2-page Create Script for Task wizard to create a script in the workspace. This is page 1 of a 2-page wizard and specifies the code to generate for your application's user interface. This is page 1 of a 2-page wizard.

Item	Description
Script Preview	Shows the current state of theApplication Factory script you are creating. Select or deselect the **Generate code to report input values** , as desired. Click **Test Script** button at any time to test your script in its current state.
Name	Selects the name field to be included in your UI.
Title	Selects the dialog title to appear in bold in the title area of the dialog box.
Description	Selects the dialog short description to appear under the title.
UI Elements	Shows the selected UI elements. You can add new UI elements by clicking **New**. You can perform this action multiple times. This opens the **New UI Element** dialog. Select from Workspace resources, file system resources, data entry and selection resource, and template resources. (you must specify the template file path if you select template .resources). Click **OK**. The code for the UI elements selected shows in your **Script Preview** window. When selecting each variable in the UI elements list, different fields for control-specific properties appear below the list: Each field contains a default value. The default value can be used or changed, as desired.

Click **Next** to go to the **Add Code to Change Workspace Files** page of this wizard to generate code that can create/delete files and change text in existing files.

Click **Finish** to create your script.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory
Creating Scripts

Related Reference

Creating Script for Task: Add Code to Change Workspace Files (page 2 of 2)
Application Factory Dialogs and Preferences Reference
FreeMarker Template Engine Overview

Creating Script for Task: Add Code to Change Workspace Files (page 2 of 2)

File ▸ New ▸ Script for Application Factory ▸ Create Script for Task ▸ Next

or

File ▸ New ▸ Other ▸ Application Factory ▸ Script for Application Factory ▸ Create Script for Task ▸ Next

Use the Create Script for Task wizard to create anApplication Factory script in the workspace. This is page 2 of a 2–page wizard.

Script Preview	Shows the current state of theApplication Factory script you are creating. Select or deselect the **Generate code to report input values**, as desired. Click **Test Script** button at any time to test your script in its current state.
Code Snippets	Click on **New** to add a new code snippet.
Operation type	Choose the type of operation from the dropdown menu to create a new file, modifyand existing file, or delete a file.
File change description	Describes the change you are making to the file. This is displayed in the Script Learning/Resolve/Commit and Archeology views.
Tags to apply	Click on **...** to select tags to apply to the file that this code snippet modifies or creates.
Select project	Select and enter (or browse to) the **Project workspace name** or select the **Project reference variable**.
Select project file	Select and enter (or browse to) the **Project-relative path** or select the **Project file reference variable**.
Configure insert/replace indicator	Specifies the expression where to find a location in the file and then the operation to perform, relative to that location.
Select source	Specifies whether the source of the input text is: ■ a string (that you insert in the text area). If it is a string, you can check **Treat string as template**. ■ a template file (for which you can specify a name or browse to for selection).

Click **Finish** to create the specified code in the script.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory
Creating Scripts

Related Reference

Creating Script for Task: Add a User Interface to your Script (page 1 of 2)
Application Factory Dialogs and Preferences Reference
FreeMarker Template Engine Overview

JSF Application Factory Dialogs Reference

This section lists the dialog/wizards for JSF web application development through the Application Factory functionality.

In This Section

New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
Use the **Web Application Settings** dialog page of the **New JSF Data-Aware Web Application** wizard to specify web settings for a new JSF web application created through Application Factory.

New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
Use the **Persistence Frameworks and Database Settings** dialog page of the **New JSF Data-Aware Web Application** wizard to define the persistence framework and database settings for a new JSF web application through Application Factory.

New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
Use the **New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)** wizard to specify the AppFuse settings for a new JSF web application created through Application Factory.

New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
Use the **Modules Settings (page 4 of 4/5)** dialog page of the **New JSF Data-Aware Web Application** wizard to specify module parameters for a new JSF web application created through Application Factory.

New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
Use the **New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)** wizard to generate entities from an existing table in a new JSF web application created through Application Factory.

New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)

Application Factory Explorer ▸ ▸ JSF Data-Aware Web Application ▸ Create Application (in Application Preview window)

Use the **New JSF Data-Aware Web Application: Web Application Settings** wizard to specify web settings when a new JSF web application is created through Application Factory. This is page 1 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Item	Description
Project name	Specify a name for the JSF data-aware project
Project contents	Specifies the default directory for the project content. You can specify your own directory name if you uncheck the **Use default** box and specify or browse to the desired directory.
Target runtime	Specifies the target runtime server for the application. Either select an installed runtime server from the drop down list or add one by clicking on **New.**
Default server	Specifies the default server. Either select a default server from the drop down list or add one by clicking on **New.**
Existing sources	Chooses what existing sources to use for project creation. You can choose from: ■ **Create new project in workspace** ■ **Create new project from existing JPA project's sources** You can select JPA project from dropdown menu. ■ **Create project from database schema**—this uses the schema from an existing database to create the application. If this option is checked, a fifth page is added to the **New JSF Data-Aware Web Application** wizard that defines the table entities to use from this database. Table data can always be imported after the project is created by right-clicking the project and selecting **Import Entities from the Database**.
Disable validators	If checked, disables any code validators. When not disabled, the workbench validates your files automatically after any build. Validators can also be reset after project creation by selecting the project as the active project in the workbench, and choosing **Window ▸ Preference ▸ Validation** and indicating the validators you want to enable or disable.
Switch off autobuild option for the workspace	If checked, switches off the autobuild option for the workspace. Automatic workspace builds can also be switched on and off after application creation by checking or unchecking the **Project ▸ Build Automatically** option for the active project in your workspace.

Click **Next** to go to **Persistence Frameworks and Database Settings (page 2 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)

New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)

Application Factory Explorer ▸ ▸ JSF Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ Next

Use the **Persistence Frameworks and Database Settings** dialog page of the **New JSF Data-Aware Web Application** wizard to define the persistence framework and database settings for a new JSF web application through Application Factory. This is page 2 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Note: You define your database settings on this wizard page. You must have an active database connection defined to complete this **New JSF Data-Aware Web Application** wizard.

Item	Description
Application Frameworks	Select the persistence framework from the dropdown menu that is to be used for this JSF web application (for example, JPA or Hibernate).
Database Settings	Specify the database settings for the application's database. You can either select an available database connection, schema and dialect from the dropdown list, or select **Add connection** to add a new database connection. This will lead you through dialogs to define the new database. ■ Specify the database connection settings in the **Database Settings** area. Any active database connection appear in the drowdown menu of the **Connection** field . You can add a database connection by clicking **Add connection**. This walks you through a wizard to add a new database connection. You must have a database connection established before you are allowed to finish the **New JSF Data-Aware Web Application** wizard. ■ Specify the database schema settings in the **Database Settings** area. Any active database schema appear in the dropdown menu of the **Schema** field . ■ Specify the dialect of the interaction with underlying database in the **Dialect** field via the dropdown menu. A database dialect controls the behavior of the database objects and deals with DDL statements (metadata) executed against the database Refer to the Eclipse DTP (Eclipse Data Tools Project) documentation for information on configuring a new database connection. This documentation can be found in the Eclipse Help in JBuilder by following the IDE path **Help ▸ Help Contents ▸ Data Tools Platform <document name>**. The database schema required by the User/Login module is created and populated with default data in the selected database/schema.

Click **Next** to go to **AppFuse Settings (page 3 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
Hibernate Documentation
Adopting a Java Persistence Framework: Which, When, and What?
Eclipse Data Tools Platform Project Home Page

New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)

Application Factory Explorer ▸ ▸ JSF Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ Next

Use the **New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)** wizard to specify the AppFuse settings for a new JSF web application created through Application Factory. This is page 3 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Item	Description
Maven Settings	Specify the Maven **Artifact Id/Project Name**, **Group Id/Package**, and **Version** fields in the **Maven Settings** area. These fields are initially populated with default values. All data-aware web application projects are Maven projects that can be compiled and deployed using either WTP or Maven (from the command line).
AppFuse Settings	Specify the AppFuse settings to use during project creation. ■ Check the option **Include AppFuse Framework sources** to include AppFuse sources in the web application. Turning on this option extracts all base AppFuse classes for the persistence, business and front-end layers into the web application project. This option is turned off by default. ■ Check the option **Use generic Manager classes** to use AppFuse DAO and service classes during CRUD code generation. This option is turned on by default. AppFuse provides generic DAO and business classes to perform basic CRUD options from any database table. Unchecking this option generates custom DAO and manager classes during CRUD generation for an entity. This can also be selected or deselected after application creation through the IDE path **Window ▸ Prefererences ▸ Application Factory ▸ Data-Aware Web Application Settings ▸ CRUD settings**
Application Mail Settings	Specify the settings for application mail. Complete the following information: ■ **Mail from**— mail address from which to send messages. ■ **Transport protocol**— type of mail transport protocol used to send mail messages, usually SMTP (simple mail transfer protocool) ■ **Host**—name of the host mail server. ■ **User name**—user name for mail access. ■ **Password**— password for the mail access user name. This can also be set after application creation through the IDE path **Window ▸ Prefererences ▸ Application Factory ▸ Data-Aware Web Application Settings**.

Click **Next** to go to **Modules Settings (page 4 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
AppFuse Home Page
Apache Maven Project Page

New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)

Application Factory Explorer ▸ ▸ JSF Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5) ▸ Next

Use the **Modules Settings (page 4 of 4/5)** dialog page of the **New JSF Data-Aware Web Application** wizard to specify module parameters for a new JSF web application created through Application Factory. This is page 4 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Item	Description
User and Login Module	Specifies that this module includes user management with security implemented using ACEGI. The module also includes a login page. The default user and password for the user/login module is admin/admin. The user/login module is installed by default.
JasperReports Module	Specifies that this module includes generation of JasperReports in standard formats (HTML, CSV, Excel, Word) based on the user module. This module is not installed by default.
Search Module	Specifies whether search capabilities can be added to the web application, based on either Apache Lucene or Compass. This module is not installed by default. The default implementation provided for the Search Module when enabled is Apache Lucene.

Click **Finish** to complete **New JSF Data-Aware Web Application** wizard or **Next** to go to **Generate Entities from Tables (page 5 of 5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
ACEGI Security System for Spring Home Page
JasperReportsHome Page
Apache Lucene Overview and Documentation Page
Lucene Compass Home Page

New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)

Application Factory Explorer ▸ ▸ JSF Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5) ▸ New JSF Data-Aware Web Application: Modules Settings (page 4 of 5) ▸ Next

Use the **New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)** wizard to generate entities from an existing table in a new JSF web application created through Application Factory. This is page 5 of a 5–page wizard. This wizard cannot be accessed through the IDE menu.

Note: This page appears in the **New JSF Data-Aware Web Application** wizard only if you checked the **Create project from database schema** on page 1 (**Web Application Settings**) of the wizard. You can also nclude existing table entries after the web application creation by right-clicking the project and selecting **Import Entities from the Database**.

Item	Description
Package	Specifies the package name of the data-aware web application project.
Tables	Lists the Table name and Entity Name for any existing database tables. You can select those table/ entity pairs that you want to include.

Click **Finish** to complete your **New JSF Data-Aware Web Application** wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)

Spring MVC Application Factory Dialogs Reference

This section lists the dialog/wizards for Spring MVC web application development through the Application Factory functionality

In This Section

New Spring MVC Data-Aware Web Application: Web Application Settings (page 1 of 4/5)

Use the **Web Application Settings** dialog page of the **New Spring MVC Data-Aware Web Application** wizard to specify web settings for a new Spring MVC web application created through Application Factory.

New Spring MVC Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)

Use the **Persistence Frameworks and Database Settings** dialog page of the **New Spring MVC Data-Aware Web Application** wizard to define the persistence framework and database settings for a new Spring MVC web application through Application Factory.

New Spring MVC Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)

Use the **New Spring MVC Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)** wizard to specify the AppFuse settings for a new Spring MVC web application created through Application Factory.

New Spring MVC Data-Aware Web Application: Modules Settings (page 4 of 4/5)

Use the **Modules Settings (page 4 of 4/5)** dialog page of the **New Spring MVC Data-Aware Web Application** wizard to specify module parameters for a new Spring MVC web application created through Application Factory.

New Spring MVC Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)

Use the **New Spring MVC Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)** wizard to generate entities from an existing table in a new Spring MVC web application created through Application Factory.

New Spring MVC Data-Aware Web Application: Web Application Settings (page 1 of 4/5)

Application Factory Explorer ▸ ▸ Spring MVC Data-Aware Web Application ▸ Create Application (in Application Preview window)

Use the **New Spring MVC Data-Aware Web Application: Web Application Settings** wizard to specify web settings when a new Spring MVC web application is created through Application Factory. This is page 1 of a 4–page or 5–page wizard. spe

Item	Description
Project name	Specify a name for the Spring MVC data-aware project
Project contents	Specifies the default directory for the project content. You can specify your own directory name if you uncheck the **Use default** box and specify or browse to the desired directory.
Target runtime	Specifies the target runtime server for the application. Either select an installed runtime server from the drop down list or add one by clicking on **New.**
Default server	Specifies the default server. Either select a default server from the drop down list or add one by clicking on **New.**
Existing sources	Chooses what existing sources to use for project creation. You can choose from: ■ **Create new project in workspace** ■ **Create new project from existing JPA project's sources** You can select JPA project from dropdown menu. ■ **Create project from database schema**—this uses the schema from an existing database to create the application. If this option is checked, a fifth page is added to the **New Spring MVC Data-Aware Web Application** wizard that defines the table entities to use from this database. Table data can always be imported after the project is created by right-clicking the project and selecting **Import Entities from the Database**.
Disable validators	If checked, disables any code validators. When not disabled, the workbench validates your files automatically after any build. Validators can also be reset after project creation by selecting the project as the active project in the workbench, and choosing **Window ▸ Preference ▸ Validation** and indicating the validators you want to enable or disable.
Switch off autobuild option for the workspace	If checked, switches off the autobuild option for the workspace. Automatic workspace builds can also be switched on and off after application creation by checking or unchecking the **Project ▸ Build Automatically** option for the active project in your workspace.

Click **Next** to go to **Persistence Frameworks and Database Settings (page 2 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)

New Spring MVC Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)

Application Factory Explorer ▸ ▸ Spring MVC Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New Spring MVC Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ Next

Use the **Persistence Frameworks and Database Settings** dialog page of the **New Spring MVC Data-Aware Web Application** wizard to define the persistence framework and database settings for a new Spring MVC web application through Application Factory. This is page 2 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Note: You define your database settings on this wizard page. You must have an active database connection defined to complete this **New Spring MVC Data-Aware Web Application** wizard.

Item	Description
Application Frameworks	Select the persistence framework from the dropdown menu that is to be used for this Spring MVC web application (for example, JPA or Hibernate).
Database Settings	Specify the database settings for the application's database. You can either select an available database connection, schema and dialect from the dropdown list, or select **Add connection** to add a new database connection. This will lead you through dialogs to define the new database. ■ Specify the database connection settings in the **Database Settings** area. Any active database connection appear in the drowdown menu of the **Connection** field . You can add a database connection by clicking **Add connection**. This walks you through a wizard to add a new database connection. You must have a database connection established before you are allowed to finish the **New Spring MVC Data-Aware Web Application** wizard. ■ Specify the database schema settings in the **Database Settings** area. Any active database schema appear in the dropdown menu of the **Schema** field . ■ Specify the dialect of the interaction with underlying database in the **Dialect** field via the dropdown menu. A database dialect controls the behavior of the database objects and deals with DDL statements (metadata) executed against the database Refer to the Eclipse DTP (Eclipse Data Tools Project) documentation for information on configuring a new database connection. This documentation can be found in the Eclipse Help in JBuilder by following the IDE path **Help ▸ Help Contents ▸ Data Tools Platform <document name>**. The database schema required by the User/Login module is created and populated with default data in the selected database/schema.

Click **Next** to go to **AppFuse Settings (page 3 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
Hibernate Documentation
Adopting a Java Persistence Framework: Which, When, and What?
Eclipse Data Tools Platform Project Home Page

New Spring MVC Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)

Application Factory Explorer ▸ ▸ Spring MVC Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New Spring MVC Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New Spring MVC Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ Next

Use the **New Spring MVC Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)** wizard to specify the AppFuse settings for a new Spring MVC web application created through Application Factory. This is page 3 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Item	Description
Maven Settings	Specify the Maven **Artifact Id/Project Name**, **Group Id/Package**, and **Version** fields in the **Maven Settings** area. These fields are initially populated with default values. All data-aware web application projects are Maven projects that can be compiled and deployed using either WTP or Maven (from the command line).
AppFuse Settings	Specify the AppFuse settings to use during project creation. ■ Check the option **Include AppFuse Framework sources** to include AppFuse sources in the web application. Turning on this option extracts all base AppFuse classes for the persistence, business and front-end layers into the web application project. This option is turned off by default. ■ Check the option **Use generic Manager classes** to use AppFuse DAO and service classes during CRUD code generation. This option is turned on by default. AppFuse provides generic DAO and business classes to perform basic CRUD options from any database table. Unchecking this option generates custom DAO and manager classes during CRUD generation for an entity. This can also be selected or deselected after application creation through the IDE path **Window ▸ Preferernces ▸ Application Factory ▸ Data-Aware Web Application Settings ▸ CRUD settings**
Application Mail Settings	Specify the settings for application mail. Complete the following information: ■ **Mail from**— mail address from which to send messages. ■ **Transport protocol**— type of mail transport protocol used to send mail messages, usually SMTP (simple mail transfer protocool) ■ **Host**—name of the host mail server. ■ **User name**—user name for mail access. ■ **Password**— password for the mail access user name. This can also be set after application creation through the IDE path **Window ▸ Preferernces ▸ Application Factory ▸ Data-Aware Web Application Settings**.

Click **Next** to go to **Modules Settings (page 4 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
AppFuse Home Page
Apache Maven Project Page

New Spring MVC Data-Aware Web Application: Modules Settings (page 4 of 4/5)

Application Factory Explorer ▸ ▸ Spring MVC Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New Spring MVC Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New Spring MVC Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ New Spring MVC Data-Aware Web Application: AppFuse Settings (page 3 of 4/5) ▸ Next

Use the **Modules Settings (page 4 of 4/5)** dialog page of the **New Spring MVC Data-Aware Web Application** wizard to specify module parameters for a new Spring MVC web application created through Application Factory. This is page 4 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Item	Description
User and Login Module	Specifies that this module includes user management with security implemented using ACEGI. The module also includes a login page. The default user and password for the user/login module is admin/admin. The user/login module is installed by default.
JasperReports Module	Specifies that this module includes generation of JasperReports in standard formats (HTML, CSV, Excel, Word) based on the user module. This module is not installed by default.
Search Module	Specifies whether search capabilities can be added to the web application, based on either Apache Lucene or Compass. This module is not installed by default. The default implementation provided for the Search Module when enabled is Apache Lucene.

Click **Finish** to complete **New Spring MVC Data-Aware Web Application** wizard or **Next** to go to **Generate Entities from Tables (page 5 of 5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
ACEGI Security System for Spring Home Page
JasperReportsHome Page
Apache Lucene Overview and Documentation Page
Lucene Compass Home Page

New Spring MVC Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)

Application Factory Explorer ▸ ▸ Spring MVC Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New Spring MVC Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New Spring MVC Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ New Spring MVC Data-Aware Web Application: AppFuse Settings (page 3 of 4/5) ▸ New Spring MVC Data-Aware Web Application: Modules Settings (page 4 of 5) ▸ Next

Use the **New Spring MVC Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)** wizard to generate entities from an existing table in a new Spring MVC web application created through Application Factory. This is page 5 of a 5–page wizard. This wizard cannot be accessed through the IDE menu.

Note: This page appears in the **New Spring MVC Data-Aware Web Application** wizard only if you checked the **Create project from database schema** on page 1 (**Web Application Settings**) of the wizard. You can also nclude existing table entries after the web application creation by right-clicking the project and selecting **Import Entities from the Database**.

Item	Description
Package	Specifies the package name of the data-aware web application project.
Tables	Lists the Table name and Entity Name for any existing database tables. You can select those table/ entity pairs that you want to include.

Click **Finish** to complete your **New Spring MVC Data-Aware Web Application** wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)

Struts 2 Application Factory Dialogs Reference

This section lists the dialog/wizards for Struts 2 data-aware application development provided through Application Factory.

In This Section

New Struts 2 Data-Aware Web Application: Web Application Settings (page 1 of 4/5)

Use the **Web Application Settings** dialog page of the **New Struts 2 Data-Aware Web Application** wizard to specify web settings for a new Struts 2 web application created through Application Factory.

New Struts 2 Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)

Use the **Persistence Frameworks and Database Settings** dialog page of the **New Struts 2 Data-Aware Web Application** wizard to define the persistence framework and database settings for a new Struts 2 web application through Application Factory.

New Struts 2 Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)

Use the **New Struts 2 Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)** wizard to specify the AppFuse settings for a new Struts 2 web application created through Application Factory.

New Struts 2 Data-Aware Web Application: Modules Settings (page 4 of 4/5)

Use the **Modules Settings (page 4 of 4/5)** dialog page of the **New Struts 2 Data-Aware Web Application** wizard to specify module parameters for a new Struts 2 web application created through Application Factory.

New Struts 2 Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)

Use the **New Struts 2 Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)** wizard to generate entities from an existing table in a new Struts 2 web application created through Application Factory.

New Struts 2 Data-Aware Web Application: Web Application Settings (page 1 of 4/5)

Application Factory Explorer ▸ ▸ Struts 2 Data-Aware Web Application ▸ Create Application (in Application Preview window)

Use the **New Struts 2 Data-Aware Web Application: Web Application Settings** wizard to specify web settings when a new Struts 2 web application is created through Application Factory. This is page 1 of a 4–page or 5–page wizard. spe

Item	Description
Project name	Specify a name for the Struts 2 data-aware project
Project contents	Specifies the default directory for the project content. You can specify your own directory name if you uncheck the **Use default** box and specify or browse to the desired directory.
Target runtime	Specifies the target runtime server for the application. Either select an installed runtime server from the drop down list or add one by clicking on **New.**
Default server	Specifies the default server. Either select a default server from the drop down list or add one by clicking on **New.**
Existing sources	Chooses what existing sources to use for project creation. You can choose from: ■ **Create new project in workspace** ■ **Create new project from existing JPA project's sources** You can select JPA project from dropdown menu. ■ **Create project from database schema**—this uses the schema from an existing database to create the application. If this option is checked, a fifth page is added to the **New Struts 2 Data-Aware Web Application** wizard that defines the table entities to use from this database. Table data can always be imported after the project is created by right-clicking the project and selecting **Import Entities from the Database**.
Disable validators	If checked, disables any code validators. When not disabled, the workbench validates your files automatically after any build. Validators can also be reset after project creation by selecting the project as the active project in the workbench, and choosing **Window ▸ Preference ▸ Validation** and indicating the validators you want to enable or disable.
Switch off autobuild option for the workspace	If checked, switches off the autobuild option for the workspace. Automatic workspace builds can also be switched on and off after application creation by checking or unchecking the **Project ▸ Build Automatically** option for the active project in your workspace.

Click **Next** to go to **Persistence Frameworks and Database Settings (page 2 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)

New Struts 2 Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)

Application Factory Explorer ▸ ▸ Struts 2 Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New Struts 2 Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ Next

Use the **Persistence Frameworks and Database Settings** dialog page of the **New Struts 2 Data-Aware Web Application** wizard to define the persistence framework and database settings for a new Struts 2 web application through Application Factory. This is page 2 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Note: You define your database settings on this wizard page. You must have an active database connection defined to complete this **New Struts 2 Data-Aware Web Application** wizard.

Item	Description
Application Frameworks	Select the persistence framework from the dropdown menu that is to be used for this Struts 2 web application (for example, JPA or Hibernate).
Database Settings	Specify the database settings for the application's database. You can either select an available database connection, schema and dialect from the dropdown list, or select **Add connection** to add a new database connection. This will lead you through dialogs to define the new database. ■ Specify the database connection settings in the **Database Settings** area. Any active database connection appear in the drowdown menu of the **Connection** field . You can add a database connection by clicking **Add connection**. This walks you through a wizard to add a new database connection. You must have a database connection established before you are allowed to finish the **New Struts 2 Data-Aware Web Application** wizard. ■ Specify the database schema settings in the **Database Settings** area. Any active database schema appear in the dropdown menu of the **Schema** field . ■ Specify the dialect of the interaction with underlying database in the **Dialect** field via the dropdown menu. A database dialect controls the behavior of the database objects and deals with DDL statements (metadata) executed against the database Refer to the Eclipse DTP (Eclipse Data Tools Project) documentation for information on configuring a new database connection. This documentation can be found in the Eclipse Help in JBuilder by following the IDE path **Help ▸ Help Contents ▸ Data Tools Platform <document name>**. The database schema required by the User/Login module is created and populated with default data in the selected database/schema.

Click **Next** to go to **AppFuse Settings (page 3 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
Hibernate Documentation
Adopting a Java Persistence Framework: Which, When, and What?
Eclipse Data Tools Platform Project Home Page

New Struts 2 Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)

Application Factory Explorer ▸ ▸ Struts 2 Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New Struts 2 Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New Struts 2 Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ Next

Use the **New Struts 2 Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)** wizard to specify the AppFuse settings for a new Struts 2 web application created through Application Factory. This is page 3 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Item	Description
Maven Settings	Specify the Maven **Artifact Id/Project Name**, **Group Id/Package**, and **Version** fields in the **Maven Settings** area. These fields are initially populated with default values. All data-aware web application projects are Maven projects that can be compiled and deployed using either WTP or Maven (from the command line).
AppFuse Settings	Specify the AppFuse settings to use during project creation. ■ Check the option **Include AppFuse Framework sources** to include AppFuse sources in the web application. Turning on this option extracts all base AppFuse classes for the persistence, business and front-end layers into the web application project. This option is turned off by default. ■ Check the option **Use generic Manager classes** to use AppFuse DAO and service classes during CRUD code generation. This option is turned on by default. AppFuse provides generic DAO and business classes to perform basic CRUD options from any database table. Unchecking this option generates custom DAO and manager classes during CRUD generation for an entity. This can also be selected or deselected after application creation through the IDE path Window ▸ Preferences ▸ Application Factory ▸ Data-Aware Web Application Settings ▸ CRUD settings
Application Mail Settings	Specify the settings for application mail. Complete the following information: ■ **Mail from**— mail address from which to send messages. ■ **Transport protocol**— type of mail transport protocol used to send mail messages, usually SMTP (simple mail transfer protocool) ■ **Host**—name of the host mail server. ■ **User name**—user name for mail access. ■ **Password**— password for the mail access user name. This can also be set after application creation through the IDE path Window ▸ Preferences ▸ Application Factory ▸ Data-Aware Web Application Settings.

Click **Next** to go to **Modules Settings (page 4 of 4/5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
AppFuse Home Page
Apache Maven Project Page

New Struts 2 Data-Aware Web Application: Modules Settings (page 4 of 4/5)

Application Factory Explorer ▸ ▸ Struts 2 Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New Struts 2 Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New Struts 2 Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ New Struts 2 Data-Aware Web Application: AppFuse Settings (page 3 of 4/5) ▸ Next

Use the **Modules Settings (page 4 of 4/5)** dialog page of the **New Struts 2 Data-Aware Web Application** wizard to specify module parameters for a new Struts 2 web application created through Application Factory. This is page 4 of a 4–page or 5–page wizard. This wizard cannot be accessed through the IDE menu.

Item	Description
User and Login Module	Specifies that this module includes user management with security implemented using ACEGI. The module also includes a login page. The default user and password for the user/login module is admin/admin. The user/login module is installed by default.
JasperReports Module	Specifies that this module includes generation of JasperReports in standard formats (HTML, CSV, Excel, Word) based on the user module. This module is not installed by default.
Search Module	Specifies whether search capabilities can be added to the web application, based on either Apache Lucene or Compass. This module is not installed by default. The default implementation provided for the Search Module when enabled is Apache Lucene.

Click **Finish** to complete **New Struts 2 Data-Aware Web Application** wizard or **Next** to go to **Generate Entities from Tables (page 5 of 5)** of this wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)
ACEGI Security System for Spring Home Page
JasperReportsHome Page
Apache Lucene Overview and Documentation Page
Lucene Compass Home Page

New Struts 2 Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)

Application Factory Explorer ▸ ▸ Struts 2 Data-Aware Web Application ▸ Create Application (in Application Preview window) ▸ New Struts 2 Data-Aware Web Application: Web Application Settings (page 1 of 4/5) ▸ New Struts 2 Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5) ▸ New Struts 2 Data-Aware Web Application: AppFuse Settings (page 3 of 4/5) ▸ New Struts 2 Data-Aware Web Application: Modules Settings (page 4 of 5) ▸ Next

Use the **New Struts 2 Data-Aware Web Application: Generate Entities from Tables (page 5 of 5)** wizard to generate entities from an existing table in a new Struts 2 web application created through Application Factory. This is page 5 of a 5–page wizard. This wizard cannot be accessed through the IDE menu.

Note: This page appears in the **New Struts 2 Data-Aware Web Application** wizard only if you checked the **Create project from database schema** on page 1 (**Web Application Settings**) of the wizard. You can also nclude existing table entries after the web application creation by right-clicking the project and selecting **Import Entities from the Database**.

Item	Description
Package	Specifies the package name of the data-aware web application project.
Tables	Lists the Table name and Entity Name for any existing database tables. You can select those table/ entity pairs that you want to include.

Click **Finish** to complete your **New Struts 2 Data-Aware Web Application** wizard.

Related Concepts

Application Factory Concepts

Related Tasks

Using Application Factory

Related Reference

Application Factory Dialogs and Preferences Reference
New JSF Data-Aware Web Application: Web Application Settings (page 1 of 4/5)
New JSF Data-Aware Web Application: Persistence Frameworks and Database Settings (page 2 of 4/5)
New JSF Data-Aware Web Application: AppFuse Settings (page 3 of 4/5)
New JSF Data-Aware Web Application: Modules Settings (page 4 of 4/5)

Axis Web Service Dialogs Reference

This section lists all of the dialog/wizards information provided for the Axis Web Service through JBuilder 2008.

In This Section

New Dynamic Web Project: New Axis Web Service Project Wizard
Use the **New Dynamic Web Project: New Axis Web Service Project** dialog to create a dynamic web project with Axis Web Service Modeling Support.

New Dynamic Web Project: Project Facets
Use the **New Dynamic Web Project: Project Facets** dialog to change the facet (unit of functionality) for the web project.

New Dynamic Web Project: Web Module
Use the **New Dynamic Web Project: Web Module** page to configure Web module settings.

New Dynamic Web Project: New Axis Web Service Project Wizard

File ▸ New ▸ Project ▸ Web Services ▸ Axis Web Service Project

Use the **New Dynamic Web Project: New Axis Web Service Project** dialog to create a dynamic web project with Axis Web Service Modeling Support. This is a 3–page wizard.

Item	Description
Project name	Specify the new project name.
Project contents	Specify the file system location (the place where the resources you create are stored). When the **Use default** check box is selected, the project is created in the file system location where your workspace resides. To change the default file system location, clear the checkbox and locate the path using the**Browse** button.
Target Runtime	Select the server where you want to deploy the Web project in this field. If you want to define a new server location, click **New** to select a server runtime environment.
EAR Membership	Adds the Web Services project to an enterprise archive (EAR) file. A new or existing EAR project file must be associated with the new Web project to facilitate deployment. The EAR file contains artifacts necessary to build a Web service. Check the **Add project to an EAR** box to add the project to an EAR file. Specify an existing EAR Project Name in the **EAR Project Name** area or use the default value of *Projectname* EAR. You can also click **New** to take you to the **New EAR Application Project** wizard to define a new EAR application project. When your Web project is created, the new EAR file is also created with the specified name. The default name is the Web project name appended with the letters **EAR**.

Click **Next** to go to the **Project Facets** page. The **Project Facets** page of this wizard allow the selection of various project functionality.

Click **Finish** to create the specified Axis Web Services project.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Related Reference

New Dynamic Web Project: Project Facets
New Dynamic Web Project: Web Module

New Dynamic Web Project: Project Facets

File ▸ New ▸ Project ▸ Web Services ▸ Axis Web Service Project ▸ New Dynamic Web Project: New Axis Web Service Project Wizard (page 1) ▸ Next

The **New Dynamic Web Project: Project Facets** dialog is the second page of the **New Dynamic Web Project: New Axis Web Service Project Wizard** wizard. Use the **Project Facets** page to select the various functionalities for the project. You can select facets by the custom configuration or use preconfigured project types.

Item	Description
Configurations	Select the configuration associated with the project. You can select pre-filled configurations or create custom configurations. Configurations can be saved or deleted using the appropriate button.
Project Facet	Select the check boxes next to the facets you want this project to have. Only valid facets for the project are listed. The list of runtimes selected for the project limits the facets shown in the list. Only the facets compatible with all selected target runtimes are shown. The currently selected facets and their version numbers limit the other facets shown in the list to compatible facets. You can find out more about the requirements and limitations for each facet by right-clicking the facet name and then clicking **Show Constraints**. To remove a facet, clear its check box. Not all facets can be removed.
Version	Choose a version number for the facet by clicking the current version number and selecting the version number you want from the drop-down list.
Show Runtimes	Click the **Show Runtimes** button and select the runtimes that you want the project to be compatible with if you want to limit the project compatibility with one or more runtimes.

Click **Next** to go to the **New Dynamic Web Project: Web Module** page. The **New Dynamic Web Project: Web Module** page of this wizard. enables the deployment of the project as a dynamic web module.

Click **Finish** to create the specified Axis Web Services project.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Related Reference

New Dynamic Web Project: New Axis Web Service Project Wizard
New Dynamic Web Project: Web Module

New Dynamic Web Project: Web Module

File ▸ New ▸ Project ▸ Web Services ▸ Axis Web Service Project ▸ New Dynamic Web Project: New Axis Web Service Project Wizard (page 1) ▸ New Dynamic Web Project: Project Facets (page 2) ▸ Next

The **New Dynamic Web Project: Web Module** dialog is the third page of the 3–page **New Dynamic Web Project: New Axis Web Service Project Wizard** wizard. Use the **Web Module** to configure specific Web module settings.

Item	Description
Context Root	The context root is the Web application root. The Web application root is the top-level directory of your application when it is deployed to the Web server. The context root can be changed after you create a project by using the project **Properties** dialog, which is accessed from the project's pop-up menu. The context root is used by the links builder to ensure that your links remain ready to publish as you move and rename files inside your project.
Content Directory	Specifies the content directory of the project.
Java Source Directory	Specifies the Java source directory.

Click **Finish** to create the specified Axis Web Services project.

Related Concepts

Web Services Overview

Related Tasks

Working in the Web Services Designer
Designing a Bottom-Up Web Service Using the Apache Axis Runtime
Designing a Top-Down Web Service Using the Apache Axis Runtime

Related Reference

New Dynamic Web Project: New Axis Web Service Project Wizard
New Dynamic Web Project: Project Facets

Dynamic Web JPA Modeling Dialogs Reference

This section lists the dialog/wizards for Dynamic Web JPA Modeling application development provided through JBuilder 2008.

In This Section

New Dynamic Web Project: New Dynamic Web JPA Modeling Project
Use the **New Dynamic Web Project: New Dynamic Web JPA Modeling Project** dialog to creates a new Dynamic web modeling project with Java Persistence API (JPA) support.

New Dynamic Web Project: Persistence unit settings page
Use the **New Dynamic Web Project: Persistence unit setting page** dialog to set the persistence settings for your new Dynamic web modeling project with Java Persistence API (JPA) support.

New Dynamic Web Project: Project Facets
Use the **New Dynamic Web Project: Project Facets** dialog to change the facet (unit of functionality) for the web project.

New Dynamic Web Project: Web Module
Use the **New Dynamic Web Project: Web Module** page to configure Web module settings for your dynamic Web JPA project.

New Dynamic Web Project: New Dynamic Web JPA Modeling Project

File ▸ New ▸ Project ▸ JPA ▸ Dynamic Web JPA modeling project

Use the **New Dynamic Web Project: New Dynamic Web JPA Modeling Project** wizard to create a dynamic web project with Java Persistence API (JPA) support. This is a 4–page wizard.

Item	Description
Project name	Specify the new dynamic web JPA modeling project name.
Project contents	Specify the file system location (the place where the resources you create are stored). When the **Use default** check box is selected, the project is created in the file system location where your workspace resides. To change the default file system location, clear the checkbox and locate the path using the **Browse** button.
Persistence Provider	Select **Hibernate** , **Toplink**, or **Other** as the persistence manager in the **Manager Name** field. Check the **Add library to the class path** box, if not checked.
Target Runtime	Select the server where you want to deploy the Web project in this field. If you want to define a new server location, click **New** to select a server runtime environment.
EAR Membership	Adds the Web Services project to an enterprise archive (EAR) file. A new or existing EAR project file must be associated with the new Web project to facilitate deployment. The EAR file contains artifacts necessary to build a Web service. Check the **Add project to an EAR** box to add the project to an EAR file. Specify an existing EAR Project Name in the **EAR Project Name** area or use the default value of *Projectname* EAR. You can also click**New** to take you to the **New EAR Application Project** wizard to define a new EAR application project. When your Web project is created, the new EAR file is also created with the specified name. The default name is the Web project name appended with the letters **EAR**.

Click **Next** to go to the **Persistence unit settings page** page of this wizard to specify the persistence settings.

Click **Finish** to create the specified dynamic Web JPA modeling project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Runtime Servers

Related Tasks

Creating a Dynamic Web Java Persistence API (JPA) Modeling Project
Setting Up a Runtime Server

Related Reference

New Dynamic Web Project: Persistence unit settings page
New Dynamic Web Project: Project Facets
New Dynamic Web Project: Web Module
Hibernate Documentation
TopLink Resources

New Dynamic Web Project: Persistence unit settings page

File ▸ New ▸ Project ▸ JPA ▸ Dynamic Web JPA modeling project (page 1) ▸ Next

Use the **New Dynamic Web Project: Persistence unit settings page** of the **New Dynamic Web JPA Modeling Project** wizard to specify your project persistence settings. This is page 2 of a 4–page wizard.

Item	Description
Persistence unit name	Specify the name of the persistence unit.
Transaction type	Choose the transaction type.
Database type	Choose the type of database. This field is limited by the **Manager Name** selected on the previous page of this wizard.
Database	Specifies further information about the selected database, including **Database Connection** and database **Schema** information. The database connection describes the method used to talk with the database server. The database schema describes the structure of the database. An active connection must exist to select a database schema. The database connection information is applicable only for Hibernate and Toplink. Database connection information for other persistence managers has to be specified manually. An active connection must exist to select a database schema. Click on the **Add Connection** link to setup a database connection. The **Schema** dropdown menu is automatically populated depending on the active database connection.
Add Connection or Reconnect	Adds a database connection or reconnects using an existing connection.

Click **Next** to go to the **Project Facets** page. The **Project Facets**page of this wizard allows the specification of project functionalities.

Click **Finish** to create the specified dynamic Web JPA modeling project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Runtime Servers

Related Tasks

Creating a Dynamic Web Java Persistence API (JPA) Modeling Project
Setting Up a Runtime Server

Related Reference

New Dynamic Web Project: New Dynamic Web JPA Modeling Project
New Dynamic Web Project: Project Facets
New Dynamic Web Project: Web Module
Hibernate Documentation
TopLink Resources

New Dynamic Web Project: Project Facets

File ▸ New ▸ Project ▸ JPA Services ▸ Dynamic Web JPA modeling project (page 1) ▸ Persistence unit settings page (page 2) ▸ Next

The **New Dynamic Web Project: Project Facets** dialog is the third page of the **New Dynamic Web Project: New Dynamic Web JPA Modeling Project** wizard. Use the **Project Facets** page to select the various functionalities for the project. You can select facets by the custom configuration or use preconfigured project types.

Item	Description
Configurations	Select the configuration associated with the project. You can select prefilled configurations or create custom configurations. Configurations can be saved or deleted using the appropriate button.
Project Facet	Select the check boxes next to the facets you want this project to have. Only valid facets for the project are listed. The list of runtimes selected for the project limits the facets shown in the list. Only the facets compatible with all selected target runtimes are shown. The currently selected facets and their version numbers limit the other facets shown in the list to compatible facets. You can find out more about the requirements and limitations for each facet by right-clicking the facet name and then clicking **Show Constraints**. To remove a facet, clear its check box. Not all facets can be removed.
Version	Choose a version number for the facet by clicking the current version number and selecting the version number you want from the drop-down list.
Show Runtimes	Click the **Show Runtimes** button and select the runtimes that you want the project to be compatible with if you want to limit the project compatibility with one or more runtimes.

Click **Next** to go to the **New Dynamic Web Project: Web Module** page. The **New Dynamic Web Project: Web Module** page of this wizard enables the deployment of the project as a dynamic web module.

Click **Finish** to create the specifieddynamic Web JPA modeling project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Runtime Servers

Related Tasks

Creating a Dynamic Web Java Persistence API (JPA) Modeling Project

Related Reference

New Dynamic Web Project: New Dynamic Web JPA Modeling Project
New Dynamic Web Project: Persistence unit settings page
New Dynamic Web Project: Web Module
Hibernate Documentation
TopLink Resources

New Dynamic Web Project: Web Module

File ▸ New ▸ Project ▸ JPA Services ▸ Dynamic Web JPA modeling project (page 1) ▸ Persistence unit settings page (page 2) ▸ Project Facets (page 3) ▸ Next

The **New Dynamic Web Project: Web Module** dialog is the fourth page of the 4–page **New Dynamic Web Project: New Dynamic Web JPA Modeling Project** wizard. Use the **Web Module** dialog screen to configure specific Web module settings.

Context Root	The context root is the Web application root. The Web application root is the top-level directory of your application when it is deployed to the Web server. The context root can be changed after you create a project by using the project **Properties** dialog, which is accessed from the project's pop-up menu. The context root is used by the links builder to ensure that your links remain ready to publish as you move and rename files inside your project.
Content Directory	Specifies the content directory of the project.
Java Source Directory	Specifies the Java source directory.

Click **Finish** to create the specified dynamic Web JPA modeling project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Runtime Servers

Related Tasks

Creating a Dynamic Web Java Persistence API (JPA) Modeling Project
Setting Up a Runtime Server

Related Reference

New Dynamic Web Project: New Dynamic Web JPA Modeling Project
New Dynamic Web Project: Persistence unit settings page
New Dynamic Web Project: Project Facets
Hibernate Documentation
TopLink Resources

EJB Modeling Projects from XDoclet Dialogs Reference

This section lists the dialog/wizards information for converting an EJB project to an EJB Modeling project through JBuilder 2008.

In This Section

EJB Modeling Project from XDoclet annotated WTP project

Use the **EJB Modeling Project from XDoclet annotated WTP Project** wizard to convert an EJB XDoclet annotated WTP project to an EJB modeling project.

EJB Modeling Project from XDoclet annotated WTP project

File ▸ New ▸ Project ▸ EJB ▸ EJB Modeling Project from XDoclet annotated WTP project

Use the **EJB modeling project from EJB project: EJB Modeling Project from XDoclet annotated WTP Project** wizard to convert an EJB XDoclet annotated WTP project to an EJB modeling project.

Item	Description
Projects	Lists all available WTP EJB projects in the current workspace. Only WTP EJB projects without the modeling nature are displayed in this list. Click the **Select All** button or the **Deselect All** button to select all deselect all items in the list.

Click **Finish** to create the new EJB modeling project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview

Related Tasks

Setting Up a Runtime Server
Creating an EJB Modeling Project based on WTP XDoclet Project

Related Reference

Creating Enterprise Beans with XDoclet Annotation Support

New EJB Modeling Dialogs Reference

This section lists all of the dialog/wizards information provided through JBuilder 2008.

In This Section

EJB Modeling Project from Java Project

Use the **EJB Modeling Project from Java Project** wizard to create an EJB modeling project from an existing Java project.

EJB Modeling Project from Java Project: Create New EJB Project from Java Project

Use the **EJB Modeling Project from Java Project** wizard to create an EJB modeling project from an existing Java project. This is page 2 of a 3-page wizard.

EJB Modeling Project from Java Project: Project Facets

Use the **EJB Modeling Project from Java Project** wizard to create an EJB modeling project from an existing Java project. This is page 3 of a 3-page wizard.

EJB Modeling Project from Java Project

New ▶ Project ▶ EJB ▶ EJB Modeling Project from Java Project

Use the **EJB Modeling Project from Java Project** wizard to create an EJB modeling project from an existing Java project. This is a 3-page wizard.

Item	Description
Projects	Lists all available Java projects for conversion, including modeling projects, in the selected workspace. Use the **Browse** button to set the workspace directory. Click the **Refresh** button to refresh the list contents.

Click **Next** to go to the **Creates new EJB Modeling Project from Java Project** page. The **Creates new EJB Modeling Project from Java Project** page of this wizard specifies characteristics of the new EJB modeling project.

Click **Finish** to create the specified EJB modeling project from a Java project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Enterprise Java Bean (EJB) Applications Overview

Related Tasks

Importing an EJB Modeling Project from a Java Project
Creating an Enterprise Java Bean (EJB) Modeling Project
Setting Up a Runtime Server

Related Reference

EJB Modeling Project from Java Project: Create New EJB Project from Java Project
EJB Modeling Project from Java Project: Project Facets

EJB Modeling Project from Java Project: Create New EJB Project from Java Project

New ▸ Project ▸ EJB ▸ EJB Modeling Project from Java Project (page 1) ▸ Next

Use the **EJB Modeling Project from Java Project** wizard to create an EJB modeling project from an existing Java project. This is a 3-page wizard. This is page 2 of a 3-page wizard.

Item	Description
Project name	Specify the new project name.
Project contents	Specify the file system location (the place where the resources you create are stored). When the **Use default** check box is selected, the project is created in the file system location where your workspace resides. To change the default file system location, clear the checkbox and locate the path using the **Browse** button.
Target Runtime	Select the server where you want to deploy the EJB modeling project. If you want to define a new server location, click **New** to select a server runtime environment.
EAR Membership	Adds the project to an enterprise archive (EAR) file. A new or existing EAR project file must be associated with the new Web project to facilitate deployment. The EAR file contains artifacts necessary to build an EJB modeling project. Check the **Add project to an EAR** box to add the project to an EAR file. Specify an existing EAR Project Name in the **EAR Project Name** area or use the default value of *Projectname* EAR. You can also click **New** to take you to the **New EAR Application Project** wizard to define a new EAR application project. When your Web project is created, the new EAR file is also created with the specified name. The default name is the Web project name appended with the letters **EAR**.
UML Version	Select the version of the Unified Modeling Language (UML) standard that will be used to build this project. To switch off the autobuild option, check the **Switch off autobuild option for the workspace** box.

Click **Next** to go to the **Project Facets** page. The**Project Facets** page of this wizard allow the selection of various project functionality.

Click **Finish** to create the specified Axis Web Services project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Enterprise Java Bean (EJB) Applications Overview

Related Tasks

Importing an EJB Modeling Project from a Java Project
Creating an Enterprise Java Bean (EJB) Modeling Project
Setting Up a Runtime Server

Related Reference

EJB Modeling Project from Java Project
EJB Modeling Project from Java Project: Project Facets

EJB Modeling Project from Java Project: Project Facets

New ▸ Project ▸ EJB ▸ EJB Modeling Project from Java Project (page 1) ▸ EJB Modeling Project from Java Project (page 2) ▸ Next

Use the **EJB Modeling Project from Java Project** wizard to create an EJB modeling project from an existing Java project. This is a 3-page wizard. This is page 3 of a 3-page wizard.

Item	Description
Configurations	Select the configuration associated with the project. You can select prefilled configurations or create custom configurations. Configurations can be saved or deleted using the appropriate button.
Project Facet	Select the check boxes next to the facets you want this project to have. Only valid facets for the project are listed. The list of runtimes selected for the project limits the facets shown in the list. Only the facets compatible with all selected target runtimes are shown. The currently selected facets and their version numbers limit the other facets shown in the list to compatible facets. You can find out more about the requirements and limitations for each facet by right-clicking the facet name and then clicking **Show Constraints**. To remove a facet, clear its check box. Not all facets can be removed.
Version	Choose a version number for the facet by clicking the current version number and selecting the version number you want from the drop-down list.
Show Runtimes	Click the **Show Runtimes** button and select the runtimes that you want the project to be compatible with if you want to limit the project compatibility with one or more runtimes.

Click **Finish** to create the EJB modeling project from a Java project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview
Enterprise Java Bean (EJB) Applications Overview

Related Tasks

Importing an EJB Modeling Project from a Java Project
Creating an Enterprise Java Bean (EJB) Modeling Project
Setting Up a Runtime Server

Related Reference

EJB Modeling Project from Java Project
EJB Modeling Project from Java Project: Create New EJB Project from Java Project

JPA Modeling Dialogs Reference

This section lists the dialog/wizards used to create new JPA modeling projects provided through JBuilder 2008.

In This Section

New JPA Modeling Project: Create a JPA modeling project
Use the **New JPA Modeling Project: Create a JPA modeling project** wizard to create a Java Modeling project with Java Persistence API (JPA) support.

New JPA Modeling Project: Persistence unit settings page
Use the 3-page New JPA Modeling Project: wizard to create a modeling project with Java Persistence API (JPA) support.

New JPA Modeling Project: Java Settings
Use the 3-page New JPA Modeling Project: wizard to create a modeling project with Java Persistence API (JPA) support.

New JPA Modeling Project: Create a JPA modeling project

File ▸ New ▸ Project ▸ JPA ▸ JPA modeling project

Use the **New JPA Modeling Project: Create a JPA modeling project** dialog to create a Java Modeling project with Java Persistence API (JPA) support. This is a page 1 of a 3–page wizard.

Item	Description
Project name	Specify the new JPA modeling project name.
Persistence manager	Select **Hibernate** , **Toplink**, or **Other** as the persistence manager in the **Manager Name** field. Check the **Add library to the class path** box, if not checked.
Contents	When the **Create new project in workspace** button is selected, the project is created in the file system location where your workspace resides. When the **Create project from existing source** button is selected you can specify the file location of the existing source. Locate the path using the **Browse** button.
JRE	When the **Use default JRE** button is selected, the default Java runtime environment is used. When the **Use a project specific JRE** button is selected you can select a specific project-related JRE from the drop down box. Click on the **Configure JRE** link to obtain a list of installed JREs and to add, edit, copy, remove, or search for other Java runtime environments.
Project layout	When the **Use project folder as root for sources and class files** button is selected, the project folder is used as the root directory for all source and class files. When the **Create separate source and output folders** button is selected, folders are created for the source and output unique from the project folder. Click the **Configure default** link to specify the default build path entries used by wizards when new Java projects are created.

Click **Next** to go to the **Persistence unit settings page** page of this wizard to specify the persistence settings.

Click **Finish** to create the specified new JPA modeling project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview

Related Tasks

Creating a Java Persistence API (JPA) Modeling Project
Setting Up a Runtime Server

Related Reference

New JPA Modeling Project: Persistence unit settings page
New JPA Modeling Project: Java Settings
Hibernate Documentation
TopLink Resources

New JPA Modeling Project: Persistence unit settings page

File ▸ New ▸ Project ▸ JPA ▸ New JPA modeling project (page 1) ▸ Next

Use the **New JPA Modeling Project: Persistence unit settings page** of the **New JPA Modeling Project** wizard to specify your project persistence settings. This is page 2 of a 3–page wizard.

Item	Description
Persistence unit name	Specify the name of the persistence unit.
Transaction type	Choose the transaction type.
Database type	Choose the type of database. This field is limited by the **Manager Name** selected on the previous page of this wizard.
Database	Specifies further information about the selected database, including '**Database Connection** and database **Schema** information. The database connection describes the method used to talk with the database server. The database schema describes the structure of the database. An active connection must exist to select a database schema. The database connection information is applicable only for Hibernate and Toplink. Database connection information for other persistence managers has to be specified manually. An active connection must exist to select a database schema. Click on the **Add Connection** link to setup a database connection. The **Schema** dropdown menu is automatically populated depending on the active database connection.
Add Connection or Reconnect	Adds a database connection or reconnects using an existing connection.

Click **Next** to go to the **Java Settings** page. The **Java Settings** page of this wizard defines the Java build settings..

Click **Finish** to create the new JPA modeling project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview

Related Tasks

Creating a Java Persistence API (JPA) Modeling Project
Setting Up a Runtime Server

Related Reference

New JPA Modeling Project: Create a JPA modeling project
New JPA Modeling Project: Java Settings
Hibernate Documentation
TopLink Resources

New JPA Modeling Project: Java Settings

File ▸ New ▸ Project ▸ JPA ▸ New JPA modeling project (page 1) ▸ New JPA modeling project (page 2) ▸ Next

Use the **New JPA Modeling Project: Java Settings** of the **New JPA Modeling Project** wizard to specify your build path settings for your Java project. This is page 3 of a 3–page wizard.

Item	Description
Source	This tab is where you specify the source location (root) of packages containing .java files. These files are then translated to .class files by the compiler and written to the defined output folder.
Projects	Specifies the required projects on the build path for a new project. This also adds all the classpath entries marked as exported (Order and Export tab) for the required project. These projects are automatically added to the referenced project list. This list is used to determine the build order as a project is built only after all its reference projects have been built.
Libraries	Choose the libraries to add to the build path, including file system (external) JAR files, folders contain class files, workbench-managed (internal) JAR files and predefined libraries.
Order and Export	Allows you to move a selected build path entry up or down in the class path order list for the new project. Entries marked in the list with a check mark are exported to be visible to the projects requiring them. Entries can be selected or deselected for exportation; source folders are always exported.
Details	contains links to additional tasks. Click the Click on **Create new source folder** to add a new Java source folder to your project. Click on **Link additional source** to link to a folder in the file system to use as an additonal source folder. Click on **Configure inclusion and exclusion filters** link to specify patterns for inclusion and exclusion filtering.
Allow output folders for source folders	check this box to permit output folders to be utilized as source folders. Click on **Create new source folder** to add a new Java source folder to your project.
Default output folder	use the default name or click **Browse** to locate a folder to use as the default output folder.

Click **Finish** to create the new JPA modeling project.

Related Concepts

Java EE Applications Overview
Modeling Applications Overview

Related Tasks

Creating a Java Persistence API (JPA) Modeling Project
Setting Up a Runtime Server

Related Reference

New JPA Modeling Project: Create a JPA modeling project
New JPA Modeling Project: Persistence unit settings page
Hibernate Documentation
TopLink Resources

ProjectAssist and TeamInsight Dialogs

This section lists the dialog/wizards information provided through JBuilder 2008 for the ProjectAssist and TeamInsight features.

In This Section

New ProjectAssist File Link
Create a link to an existing ProjectAssist .pacx file.

New ProjectAssist File
Creates a new ProjectAssist stack file (.pacx).

New ProjectAssist File:Select Stack Components
Selects the stack components for a new ProjectAssist stack file (.pacx).

New ProjectAssist file: Choose disk scan paths
Use the **Choose disk scan paths** to specify directory, paths and files to scan for preexisting components prior to the stack file creation.

TeamInsight Viewer
Use the **TeamInsight Viewer** to browse the web pages of the TeamInsight tools.

Edit Repository Query or New XPlanner Query
Queries an XPlanner repository by specifying a requested list of tasks.

TeamInsight User Mail Notification
Through the **User Notification** window, the ProjectAssist Administrator specifies e-mail message text and users to notify of the TeamInsight component availability.

ProjectAssist Mail Preferences
Enables the ProjectAssist Administrator mail account and configures mail preferences.

Passwords for Authorization
Allows the ProjectAssist Administrator to create passwords..

Preinstalled Component Scan: Choose scan type
Use the **Preinstalled Component Scan: Choose scan type** dialog to select the type of scan to perform on the server for preinstalled ProjectAssist components.

ProjectAssist Configuration Editor: Projects
To reach the Stacks, Users, and Projects tabs at the bottom of the ProjectAssist configuration editor, click on the ProjectAssist stack file name (.pacx extension).

ProjectAssist Configuration Editor: Stacks
To reach the Stacks, Users, and Projects tabs at the bottom of the ProjectAssist configuration editor, click on the ProjectAssist stack file name (.pacx extension).

ProjectAssist Configuration Editor: Users
To reach the Stacks, Users, and Projects tabs at the bottom of the ProjectAssist configuration editor, click on the ProjectAssist stack file name (.pacx extension).

Maven Project from Archetype
Specifies a new project using the Maven archetype wizard.

New StarTeam Query
Allows the definition of a new query against a StarTeam task or change request repository.

StarTeam Repository Settings
Specifies the settings for the StarTeam Mylar repository for change requests and/or tasks.

New ProjectAssist File Link

File ▶ New ▶ Other ▶ Team ▶ ProjectAssist File Link

Creates a link to a ProjectAssist file. Open either the full or the ProjectAssist Administrator version of JBuilder 2007. Select **File ▶ New ▶ Other ▶ Team ▶ ProjectAssist File Link**. This opens the **New ProjectAssist File Link** wizard to link to an existing ProjectAssist stack file.

Item	Description
File location	Specify the location of the existing ProjectAssist component stack file (which will have a .pacx file extension).
Browse	Click this button to browse to the file location for the existing ProjectAssist stack file.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

New ProjectAssist File

File ▸ New ▸ Other ▸ Team ▸ ProjectAssist File

To create a new project assist developer stack file, open either the full or the ProjectAssist Administrator version of JBuilder 2007 through the menu path **File ▸ New ▸ Other ▸ Team ▸ ProjectAssist File**. This opens the **New ProjectAssist file** wizard to create a new ProjectAssist stack file.

Item	Description
File name	Specifies the ProjectAssist component stack file name (which will have a .pacx file extension).
Administrator name	Specifies the ProjectAssist server Administrator name.
Initials	Specifies the ProjectAssist Administrator initials.
E-mail	Specifies the e-mail address of the ProjectAssist Administrator.
ID	Specifies the ProjectAssist Administrator alias ID.
Project name	Specifies a project name for the initially generated project.
Description	Provides additional description of the initial project.
ProjectAssist install directory	Provides (default value) or specifies (user-provided) root directory and path for the ProjectAssist installation.
ProjectAssist data directory	Provides (default value) or specifies (user-provided) root directory and path for ProjectAssist data files.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

New ProjectAssist File:Select Stack Components

File ▸ New ▸ Other ▸ Team ▸ ProjectAssist File ▸ Next

To create a new project assist developer stack file, open either the full or the ProjectAssist Administrator version of JBuilder 2007. Select **File ▸ New ▸ Other ▸ Team ▸ ProjectAssist File**. This opens the **New ProjectAssist file** wizard to create a new ProjectAssist stack file. Fill in the appropriate information and select **Next** to move to the **New ProjectAssist file: Select Stack Components** screen.

Note: The selections on this page are determined by your JBuilder 2007 Enterprise Edition. StarTeam choices appear only if you have the JBuilder 2007 Enterprise Edition. Two columns in the **Select Stack Components** screen are titled **New or existing installation** and **Existing installation only**. The software component choices you can install for each function are listed in the appropriate column.

Item	Description
Version Control System	Select the version control system to include component stack file. The choices are CVS, StarTeam and/or Subversion (depending on whether you have the JBuilder 2007 Enterprise or JBuilder 2007 Enterprise Edition).
Continuous Build System	Select the build system for continuous integration to include in the component stack file.
Defect Tracker	Select the defect tracking or change request system to include in the component stack file. The choices are Bugzilla and/or StarTeam (depending on whether you have the JBuilder 2007 Enterprise or JBuilder 2007 Enterprise Edition).
Task Provider	Select the task provider system to include in the component stack file. The choices are and/or StarTeam and/or XPlanner (depending on whether you have the JBuilder 2007 Enterprise or JBuilder 2007 Enterprise Edition).

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

Maven Project from Archetype

File ▸ New ▸ Project or Other ▸ Maven ▸ Maven Project from Archetype

To create a new project based on the Maven archetype, go to **File ▸ New ▸ Project or Other ▸ Maven ▸ Maven Project from Archetype**. This opens the **Maven Project from Archetype** wizard. Fill in the appropriate information and click **Finish** to create your new project.

Note: The ProjectAssist Administrator can also add a project based on the Maven archetype through the **Project** tabs of the **ProjectAssist Configuration Editor** .

Item	Description
Archetype Group Id	Select the Maven archetype group ID from the dropdown list.
Archetype Artifact Id	Select the Maven archetype artifact ID from the dropdown list.
Archetype Version	Select the Maven archetype version from the dropdown list.
Remote Repositories	Enter the URL of a remote repository in which to search for the specified Maven archetype.
Project Group Id	Enter the Maven Group ID for the project to be created.
Project Artifact Id	Enter the Maven artifact ID for the project to be created.
Project Folder	Enter the root project folder name for the new project. You can browse for a current folder by clicking on the **Browse** button.
Project Version	Select the product version from the dropdown menu.
Project Package	Specity the project package name for the project to be created.

Click **Finish** to create the specified project based on the Maven archetype model.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

Related Reference

ProjectAssist Configuration Editor: Projects

New ProjectAssist file: Choose disk scan paths

File ▸ New ▸ Other ▸ Team ▸ ProjectAssist File

Use the **Choose disk scan paths** page of the **New ProjectAssist File** wizard to specify directory, paths and files to scan for preexisting components prior to the stack file creation.

Item	Description
A:\	Choose paths and folders in the A:\ directory to scan (all or individual folders and files) for preexisting components.
C:\	Choose paths and folders in the C:\ directory to scan (all or individual folders and files) for preexisting components. This is the default selection.
D:\	Choose paths and folders in the D:\ directory to scan (all or individual folders and files) for preexisting components.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

TeamInsight Viewer

Window ▸ Open TeamInsight Viewer ▸ <tool-name> or Open All

The TeamInsight Viewer is a custom browser window that opens in the editor pane. You can open any one or all of the TeamInsight tools from the **Window** menu. The TeamInsight window contains its own navigation bar located at the top of the window, including an entry field for URLs.

Item	Description
Navigation Bar	Located at the top of the TeamInsight Viewer, the navigation bar contains several buttons, such as: Home, Back, Forward, Stop, Refresh, Go to URL, and Go to home location for application. The URL field in the navigation bar contains the URL of the web page currently being displayed. The **Go to home location for application** (an icon of a ringed planet) is useful for returning to the configured application after you visit another URL in the TeamInsight Viewer.
Window	The window area on the TeamInsight Viewer is a tabbed window for displaying the web pages of TeamInsight tools. To scroll the TeamInsight Viewer, use the scroll bars at the right-hand side of the viewer. After you open the Liferay portal in the TeamInsight Viewer, you can click links in several of the portlets (such as Continuum and XPlanner) to display the main web page of the application server that generates the portlet.
Tabs	The TeamInsight Viewer window has a tab for each TeamInsight tool that you have opened.

Related Concepts

ProjectAssist and TeamInsight Overview
Liferay: The TeamInsight Project Portal

Related Tasks

Opening the TeamInsight Viewer and the Liferay Portal
Configuring Your TeamInsight Client

Edit Repository Query or New XPlanner Query

<task-list-context-menu> ▸ Open

<task-list-context-menu> ▸ New Query

To edit an existing query (or enter all-new values): In the Task List, either double-click an XPlanner repository or right-click an XPlanner repository and select **Open** from the context menu. The **Edit Repository Query** dialog box is displayed.

To create a new query: In the Task List, right-click anywhere and select **New Query**.

Mylar displays a preliminary **New Repository Query** dialog box ("Add or modify repository query"). On the Mylar dialog box, you can select from the available XPlanner and Bugzilla repositories, as configured in ProjectAssist and displayed in the TeamInsight Viewer, or you can click **Add Task Repository** to connect to another repository. (Mylar connects to several types of repositories.) Then the **New XPlanner Query** dialog box is displayed.

Note: A tree structure of the projects, iterations, and stories in the XPlanner repository appears in both of these dialog boxes. You can select to find either stories or tasks, and you can search a project, an iteration, or a specific story. The repository query finds either all your current tasks or only those tasks that match the query, and lists the resulting tasks in the **Task List**.

Item	Description
Query name	Enter a name to identify this query and its results.
All my current tasks	Finds all your current tasks within the selected XPlanner repository.
Selected tasks	Finds only the tasks that meet the values you have specified in the subfields.
Projects, Iterations, and User Stories	Either select the name of your project or expand the directory listing and select the correct iteration or user story that you want to search.
Grouping	Select the grouping you want: Tasks or User Stories.
Scope	Select either All (meaning all tasks in the selected project, iteration, or user story) or My (meaning only your tasks).

Related Concepts

Liferay: The TeamInsight Project Portal
XPlanner: Project and Team Management

Related Tasks

Opening the TeamInsight Viewer and the Liferay Portal

TeamInsight User Mail Notification

<file>.pacx ▸ Mail icon (upper-right) ▸ Mail configuration preferences (if not enabled) ▸ User Notification

Use this dialog box to select TeamInsight users to notify and send a prepared (or edited) notification message.

Item	Description
Subject text	Fill in your Subject line text or use the default text provided.
Message body	Defaults to prefilled e-mail message text to send (in HTML format). You can edit this to your own message or accept the default text.
Users to be notified	Lists the users to be notified in this mail message. Add or remove users for notification by clicking on the **Add** or **Remove** button.
Configure Mail . . .	Allows you to configure your e-mail system to send this e-mail (if it has not already been done).
Send Notification	Attaches a configuration file for the TeamInsight user to configure the client machine for access to the ProjectAssist component servers. It then sends the notification message to specified recipients.
Cancel	Click this button to cancel sending the notification message.

Note: After each TeamInsight user additions, the ProjectAssist Administrator is asked if the notification message is to be sent. The Administrator then has access to this dialog window if a notification message is to be sent.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

ProjectAssist Mail Preferences

Window ▸ Preferences ▸ Mail

Enabling mail can also be initiated through the **Send Mail Notification** icon on the ProjectAssist configuration designer.

Use the **Preferences Mail** dialog box to enable the ProjectAssist Administrator's mail account prior to sending e-mail notifications to users.

Item	Description
Enable Mail	Click the check box to enable ProjectAssist e-mail settings.
Sender name	Enter the Administrator (sender) name in the text area.
Sender email address	Enter the Administrator (sender) e-mail address in the text area.
SMTP server address	Enter the address of the Simple Mail Transport Protocol (SMTP) server for the Administrator's e-mail.
Use custom SMTP port	Click the check box to use a custom SMTP port.
SMTP server port	If the **Use custom SMTP port** check box is selected, Enter the server information in the **SMTP server port** field.
Server requires authentication	Click the check box if the server requires user authentication.
Name	If the **Server requires authentication** check box is selected, enter the user name for server authentication.
Password	If the **Server requires authentication** check box is selected, enter the password for server authentication.
POP before SMTP required	Check the **POP before SMTP required** to route through a POP server.
Host	If the **POP before SMTP required** check box is selected, enter the POP **Host** identifier in the text box.
Name	If the **POP before SMTP required** check box is selected, type the user **Name** in the text box.
Password	If the **POP before SMTP required** check box is selected, type the user password in the text box.
Send Test Message	Click **Send Test Message** to confirm the configured mail preferences via a test message.
Restore Defaults	Click the **Restore Defaults** button to restore the mail preferences setting to a default state.
Apply	Click the **Apply** button to enable your settings.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

Passwords for Authorization

<file>.pacx ▸ Install ProjectAssist icon

Specifies passwords for the Administrator for all the ProjectAssist server components. The **Password for Authorization** dialog pops up when you initially click on the **Install ProjectAssist** icon on any of the **ProjectAssist Designer** configuration editor windows (Stacks, Users, or Projects).

Item	Description
Use the same passwords when appropriate	Check this box if you want to use the same passwords for all components. All passwords text areas are filled when you type in the initial password. The default value is checked.
Administrator ID	For each component, this area is prefilled with the ProjectAssist Administrator's username for log in to that component. For Bugzilla and Liferay, the **Administrator ID** is *username@somewhere*.com (an E-mail address). For Continuum, Subversion and XPlanner, the **Administrator ID** is *username*.
Password	Specifies the Administrator password for each ProjectAssist component. If the **Use the same password when appropriate** box is checked, the same password is used when you type the initial password into the text area.
Confirm password	Enter the Administrator's password for the component again to confirm.
Validate Passwords	Click to validate that all passwords entered are in the correct syntax.
Install	Click to install the passwords and components.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

Preinstalled Component Scan: Choose scan type

File ▸ New ▸ Other ▸ Team ▸ Project Assist File

Use the **Preinstalled Component Scan: Choose scan type** page of the **New ProjectAssist File** wizard to select the type of scan to perform for any preinstalled ProjectAssist components on the installation server.

Item	Description
Components to scan for	Lists all the components that are included in the preinstallation scan.
Skip system scan	Check this button to skip the system scan.
Minimal system scan (system path, services and running processes)	Check this button to perform a minimal scan of your system.
Thorough system scan (disk, system path, services and running processes)	Check this button to perform a complete scan of your system. This is the default value.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

ProjectAssist Configuration Editor: Projects

<file>.pacx ▸ Projects tab

Click the **Projects** tab to navigate to the **Projects** configuration editor where you add and configure projects for your team. The Sample Project generated during the stack file creation should already be in the Projects list.

Note: The individual TeamInsight component fields in the **Projects** tab are determined by the component selected during the ProjectAssist stack file installation. The descriptions below are for all components; your tab will only have choice for your defined TeamInsight components.

Item	Description
Projects	List the names of projects to be added. The Sample Project generated during the stack file creation should already be in the list.
Clone	Click a project name in the **Projects** area. Click **Clone** to create a project with the same configuration information. user. Replace the generic information with any project specific information.
Add	Click **Add** to create a project with the default values assigned. Replace the generic filler information with user-specific information.
Remove	Select a project in the **Projects** area and click **Remove** to remove the project.
Name	Enter the new project name or use the default name.
Description	Enter a description of the project.
Project Content Provider	Select a project content provider from the dropdown list. The type of projects provided in the Project Content dropdown list is determined by the Teaminsight component tools installed. The following choices may be present: Select **Maven2 Archetype Project** to generate a Maven2 project using a Maven archetype model for quick generation of a Maven project. If a Maven 2 Arechtype Project is selected, the **Project Content** area includes fields to specify aspects of the Maven project you are trying to create. Select **Maven2 Sample Project** to generate a Maven2 sample project with POM files. Select **CodeGear Sample Project** to generate a Maven2 sample project with POM files that enable Optimizeit Code Coverage, Together Code Audits and Metrics, and Together Documentation Generation. Select **Existing Project Directory** to links to an existing project with a pom.xml file on a local directory. Select **Project checked into version control** if you are assimilating an existing CVS or Subversion installation and this project has a pom.xml file. Select **Project checked into Subversion and uploaded to Continuum** if you are assimilating both the Subversion and Continuum installation and you want to specify the name of a project on the Continuum server.

Group Id	Enter the Maven group ID.
Artifact Id	Enter the Maven artifact ID.
JBuilder Home (CodeGear Sample Project)	Select the home directory of the JBuilder installation.
Optimizeit Agent Directory (CodeGear Sample Project)	Select the directory of the Optimizeit Agent (for example: *<JBuilderHome>/eclipse/optimizeit-agent*.
Archetype group Id (Maven archetype project)	Select the Maven archetype group ID from the dropdown list.
Archetype Artifact Id (Maven archetype project)	Select the Maven archetype artifact ID from the dropdown list.
Archetype Version (Maven archetype project)	Select the Maven archetype version from the dropdown list.
Remote Repositories (Maven archetype project)	Enter the URL of a remote repository in which to search for the specified Maven archetype.
Project Group Id ((Maven archetype project)	Enter the Maven Group ID for the project to be created.
Project Artifact Id (Maven archetype project)	Enter the Maven artifact ID for the project to be created.
Project Version (Maven archetype project)	Select the product version from the dropdown menu.
Project Package (Maven archetype project)	Specity the project package name for the project to be created.
SVN Repository path	Enter the repository path information for Subversion.
Path in repository	Enter the CVS repository path field in the **Path in repository** field. This is verified against existing paths and module names. Therefore, on the host machine, the project is in "repository path" plus "path in repository" (for example, "/public/SampleProject").
CVS vendor tag	Enter the required Vendor tag information in the **Vendor tag** when importing a project. An attempt is made to derive the vendor tag from the administrator's email address, but you can change this field to any value. This field cannot be blank.
Bugzilla project (product) name	Enter the name for the related Bugzilla project file.
XPlanner project name	Enter the name for the related XPlanner project file.
Bugzilla project version	Enter the Bugzilla project version.
Bugzilla project component(s)	Enter the name(s) for the Bugzilla project component(s).
StarTeam Version Control	Enter the name of the StarTeam project in the **Project** field. Enter the StarTeam view in the **View** field, or leave empty for the default view. Enter the StarTeam folder name in the **Folder** field, or leave empty for the default folder.
StarTeam Change Requests	Enter the name of the StarTeam project in the **Project** field. Enter the StarTeam view in the **View** field, or leave empty for the default view. Enter the StarTeam folder name in the **Folder** field, or leave empty for the default folder.
StarTeam Tasks	Enter the name of the StarTeam project in the **Project** field. Enter the StarTeam view in the **View** field, or leave empty for the default view. Enter the StarTeam folder name in the **Folder** field, or leave empty for the default folder.
Install Developer Stacks	Click on this icon in the upper-right of the page to install after all projects and users have been added.
Uninstall Developer Stacks	Uninstalls components.
Send Mail Notification	Enables mail (if not previously enabled) and sends a notification message to users that new projects have been added.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

ProjectAssist Configuration Editor: Stacks

<file>.pacx ▸ Stacks tab

Use the **Stacks** configuration editor to configure the individual components installed in the ProjectAssist stack file installation. Click on the component name in the left-side list on this page to bring up configuration information for that component.

There are also other categories on the **Stacks** page such as Shared Components, Settings and so forth although you may not need to change the default settings in these categories.

Item	Description
Install on local machine	Check this box for any component (Subversion, Continuum, Bugzilla, or XPlanner) to install the component on the local ProjectAssist server. This box is grayed out (not selectable) for assimilate-only ProjectAssist components (CVS or StarTeam). This creates a **General** area (for all components) with information on the Name, Description (all components), Installation Directory (Subversion, Continuum, and Bugzilla) and Data Directory (Subversion only). Bugzilla also has an **SMTP server** field that specifies the Bugzilla mail server name and port. Continuum also shows fields specifying **Continuum HTTP port**,**Continuum RPC port** and **Windows service name**.
Refer to an existing installation (local or remote)	Check this box for any component (Subversion, Continuum, Bugzilla, or XPlanner) to assimilate an existing component install (local or remote). For CVS and StarTeam components, this box is checked by default and cannot be changed. On the **Subversion** page, enter the existing component installation location in the **URL** field. The **Admin username** field defaults to the Administrator's name. Enter the Subversion password in the **Password** field. You can enable the ability to add users remotely to Subversion by checking the **Enable add users to remote server** box. On the **Continuum** page, enter the existing component installation location in the **URL** field. Enter the port number in the **Continuum RPC port** field. The **Admin username** field defaults to the Administrator's name. Enter the Continuum password in the **Password** field. On the **Bugzilla** page, enter the existing component installation location in the **URL** field. The **Email Address** field defaults to the Administrator's E-mail address. Enter the Bugzilla password in the **Password** field. On the **XPlanner** page, enter the existing component installation location in the **URL** field. The ProjectAssist Administrator **Admin username** field defaults to the Administrator's name. Enter the XPlanner password in the **Password** field.

Tip: On each component configuration page, click **Test Connection** to validate the configuration information.

Install Developer Stacks	Click on this icon in the upper-right of the page to install the stack components after configuration.
Uninstall Developer Stacks	Click on this icon in the upper-right of the page to uninstall the stack components.
Send Mail Notification	Do not use this icon on the **Stacks** page.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

ProjectAssist Configuration Editor: Users

<file>.pacx ▸ Users tab

Click the **Users** tab to navigate to the **Users** configuration editor to add and configure users for your project and the components. The Administrator created initially should already be in the User List. When you click on the Administrator's name, the **General Information** frame appears with user details and permissions for the Administrator.

Item	Description
User List	The Administrator created initially should already be in the User List. If you click the Administrator name, the **General Information** frame appears with user details and permissions for the Administrator. Names are added to this list as you clone or add more users.
Clone	Click a user name in the **User List** area. Click **Clone** to create a user with the same assigned user roles as the already defined user. Replace the generic filler information with user-specific information. To change the user's role for any TeamInsight component, right-click on the component in the **Roles** list. The roles of Administrator, Developer, or No Access can be assigned for each user according to components.
Add	Click **Add** to create a user with the default roles assigned to all the ProjectAssist components. Replace the generic filler information with user-specific information. The default role assignment for all components (except MySQL) is Developer. The default for MySQL (used by the Bugzilla component) is No Access. To change the user's role for any TeamInsight component, right-click on the component in the **Roles** list. The roles of Administrator, Developer, or No Access can be assigned for each user according to components.
Remove	Select a user in the **User List** area and click **Remove** to remove a user prior to the clicking the **Install Developer Stacks** icon. Users cannot be removed after they have been added by clicking on the **Install Developer Stacks** icon. Be sure the information is correct before installing.
Install Developer Stacks	Click on this icon in the upper-right of the page to install the users after configuration to the compenent servers. Users cannot be removed after they have been added with the **Install Developer Stacks** icon. Be sure the information is correct before clicking this icon.
Uninstall Developer Stacks	Is not applicable to the **Users** page.
Send Mail Notification	Enables mail and sends a notification message to users after they are added.

Related Concepts

ProjectAssist and TeamInsight Overview

Related Tasks

Configuring Your TeamInsight Client

New StarTeam Query

Window ▸ Configure Mylyn ▸ StarTeam project name ▸ change request or task view

The ProjectAssist functionality enables you to add the StarTeam repository change requests and StarTeam repository tasks to the Eclipse **Task List** view, and to use Mylyn to define queries against those repositories.

Item	Description
Query Name	Enter a name for your query.
All my current tasks and change requests	Select this button if you wish to see both repository queries for change requests and tasks.
Selected tasks or change requests	Check this button to select tasks or change requests for specific project/views/folders.
Type	Select whether to include tasks or change requests, or both, in your query. If you select Tasks in the **Type** field, a single query node is created in the Task List view, with all the applicable tasks sublisted. If you select Change Requests in the **Type** field, a single query node is created in the **Task List** view, with all the applicable change requests sublisted. Selecting both generates two lists in the **Task List** view.
Scope	Specifies the scope of control over the type. If you select **All** in this group, all tasks or change requests from selected StarTeam entities are added to the query results. If you select **My**, then only your own tasks or change requests are added to the query results.

Related Concepts

ProjectAssist and TeamInsight Overview
Mylyn Concepts

Related Tasks

Adding Mylyn Repositories for Bugzilla and XPlanner
Configuring Your TeamInsight Client

Related Reference

StarTeam Repository Settings
External Documentation for Mylyn from Eclipse.org
External Documentation about Mylyn Connectors to Repositories
External Article: Task-Focused Programming with Mylyn

StarTeam Repository Settings

Window ▸ Configure Mylyn ▸ StarTeam project repository name

Use the **StarTeam Repository Settings** dialog for Mylyn-based repositories for StarTeam change requests and/or tasks.

Item	Description
Server <address:port>	Select the address of the StarTeam repository server in the dropdown list. The address is in the format *address:port.*
Label	Enter the label for the StarTeam repository
User ID	Enter the User ID for the authorized StarTeam user.
Password	Enter the password for the User ID.
Default location	Enter or browse for the default location of the StarTeam repository. This is the location to be searched for all entities of this repository (the location against which the Mylar query is run).
StarTeam Repository Type	Select the type of repository you would like to establish. If you select Tasks in the **Type** field, a single query node is created in the Task List view, with all the applicable tasks sublisted. If you select Change Requests in the **Type** field, a single query node is created in the **Task List** view, with all the applicable change requests sublisted. Selecting both generates two lists in the **Task List** view.
Character Encoding	Use the default character encoding of UTF-8 or click **Other** and select another encoding method from the dropdown list.
Validate Settings	Click the **Validate Settings** buttons to verify that all your settings are correct.

Related Concepts

ProjectAssist and TeamInsight Overview
Mylyn Concepts

Related Tasks

Adding Mylyn Repositories for Bugzilla and XPlanner
Configuring Your TeamInsight Client

Related Reference

New StarTeam Query
External Documentation for Mylyn from Eclipse.org
External Documentation about Mylyn Connectors to Repositories
External Article: Task-Focused Programming with Mylyn

Peer to Peer Dialogs Reference

This section lists the dialog/wizards information for peer to peer interaction provided through JBuilder 2008.

In This Section

Peer To Peer Preferences

Window ▸ Preferences ▸ Peer To Peer

Use this dialog box to set preferences for collaborating with peers.

Item	Description
Enable Peer to Peer Subsystem	Enables the peer to peer features and opens the Peers view when you click the **Apply** button.
Name	The name you want to display to peers. This defaults to your user name.
Description	An optional description that can help identify you to peers.
Image	An optional icon that helps identify you in a peer to peer collaboration session. The following file types are accepted: `.GIF`, `.JPEG`, and `.PNG`.
Browse	Displays the **Open** dialog box, where you browse to the location of an image to use for identification. Any icon you use is resized to 48 x 48 pixels. This may distort the image.
Filtering	The adapter to use. Select NONE if you have only one adapter or want to be prompted at peer to peer startup for the adapter.
Log Chat Messages	Enables logging of chat messages to a file.
Workspace Directory	The Eclipse workspace folder in which to save chat logs.
Incoming Message Color	The color for incoming messages.
Outgoing Message Color	The color for outgoing messages.
Status Message Color	The color for status messages.
Automatic Receive Enabled	Enables automatic file transfer and allows a file sent from a peer to be automatically received, rather than downloaded manually.
Workspace Directory	The Eclipse workspace folder to save files to when automatic receive is enabled.
Audio Feedback Enabled	Enables audio feedback. There are different sounds for incoming messages and incoming status information.
Slider	Adjusts the audio feedback volume.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Setting Collaboration Preferences
Opening a Peer to Peer Session

Related Reference

Peers View
New Contact Group
Send Stack Trace
Send Web Link
Send VCS Link

Peers View

Window ▸ Show View ▸ Other ▸ Peer to Peer ▸ Peers

The Peers view is where you discover peers, choose the peer(s) you want to chat with, create and manage contact groups, chat with peers, and send data to peers. The Peers view contains the Peers pane on the left and the Collaboration pane on the right.

Peers Pane

Item	Description
Status	Your current status: Available, Away, or Offline.
IP Address	Your IP address; used for identification. The IP address is shown when you are online.
Available Local Peers	The list of available peers.
Contact Groups	Your contact groups. Peers assigned to each contact group are also displayed.

Collaboration Pane

Item	Description
Peer in Session	The name(s) of the peer(s) in the chat session.
Chat Area	The chat.
Message Area	The message input area.
Session Tab	The representation of the session. To close the session, click the X on the tab.

The Collaboration pane toolbar contains buttons for:

- Adding peer(s) to the chat session
- Sending a file to peer(s)
- Sending a web link to peer(s)
- Sending a stack trace to peer(s)
- Closing all chat sessions

Related Concepts

Peer to Peer Collaboration

Related Tasks

Enabling Peer to Peer Collaboration
Opening a Peer to Peer Session
Chatting with Peers
Sending Data To Peers

Related Reference

New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

New Contact Group

Window ▸ Show View ▸ Other ▸ Peer to Peer ▸ Peer ▸ OK ▸ Add Contact Group

Use this dialog box to create a name for a contact group.

Item	Description
Group Name	Enter the contact group name.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Managing Contact Groups

Related Reference

Peers View
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Send Stack Trace

Window ▸ Show View ▸ Other ▸ Peer To Peer ▸ Peers ▸ Send Stack Trace

Use this dialog box to send a stack trace to a peer.

Item	Description
Stack Trace	The stack trace to send. Paste the stack trace from the Clipboard.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Send Stack Trace

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link
Send VCS Link

Send Web Link

Collaboration pane toolbar ▸ Send Web Link to Peers in Collaboration icon

Use this dialog box to specify a web link to send to a peer during a collaboration session.

Item	Description
Web link	The URL of the web link to send. Click **OK** to send.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Sending Data To Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send VCS Link

Send VCS Link

VCS projectname (right-click) ▸ Send VCS Link to Peer

Projects are shared through a repository. When projects are shared, the **Navigator** or **Package Explorer** displays the project repository and location. You can send your peers a link to the VCS project repository by right-clicking on the project name and selecting **Send VCS Link to Peer**. This opens the **Select Peers** dialog.

Item	Description
Available peers	Lists all available peers to whom you can send the VCS link. Click **Select** to send. The project is sent as a VCS link to the selected peer. The message **Sending VCS link for project "<Project Name>"** is displayed in your chat area.

Related Concepts

Peer to Peer Collaboration

Related Tasks

Sharing Team-Enabled Projects with Peers
Sending Data To Peers

Related Reference

Peers View
New Contact Group
Peer To Peer Preferences
Peers View
New Contact Group
Peer To Peer Preferences
Send Stack Trace
Send Web Link