

JAVA™ DEVELOPERS' JOURNAL

JavaDevelopersJournal.com
Volume: 3 Issue: 10, 1998

Why Superman Works Alone

by Sean Rhody pg. 5

When Do We Reach Nirvana?

by Thomas Murphy pg. 7

Widget Factory JComponentTree

by Claude Duguay pg. 22

Straight Talking Beautiful Things Come in Small Bundles

by Alan Williamson pg. 27

Product Reviews Vision Jade 4.0

by Jim Milbery pg. 34

Emblaze Audio/Video

by David Jung pg. 30

IMHO

Java Development Tools in Transition

by Paul Petersen pg. 60

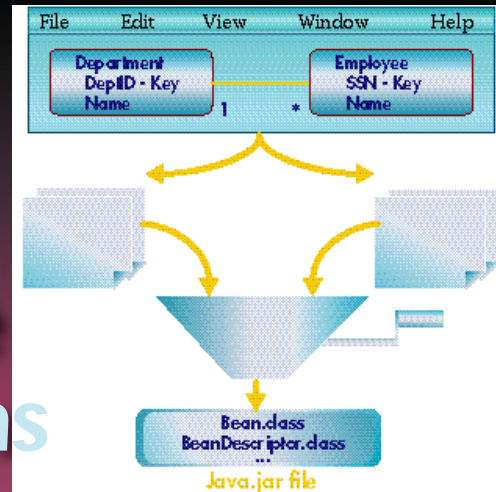
The Grind

SQL Results from an Application Server

by Java George pg. 66

RETAILERS PLEASE DISPLAY UNTIL DECEMBER 10, 1998

Graphical Development Process in Enterprise JavaBeans



Cover Story: Creating a Custom Layout Manager *Dealing with problems when your manager can't* Joseph Cozad 8

What AWT Version Do You Use? *Force components to use 1.1 and discover those that use 1.0* Andrei Cioroianu 16

Case Study: Large Scale Software Development in Java *Issues and solutions* Sriram Sankar 38

Active Menus Without Graphics *For Web pages using cascading style sheets, layers and JavaScript* Ken Jenks 43

Under the Sun: Handling the Load *Peak application performance with JavaLoad load testing software* Matt Evans 48

Persistence in Enterprise JavaBeans *Bean- and container-managed EntityBean persistence and their merit* Patrick Ravenel 50

Software Engineering in Startups Part 2 *The life-cycle stages and tracking of requirements* Juergen Brendel 56

RogueWave StudioJ

<http://www.roguewave.com/ad/studioj>

ProtoView JSuite

<http://www.protoview.com>

Schlumberger Cyberflex

<http://www.cyberflex.slb.com>

EDITORIAL ADVISORY BOARD

Ted Coombs, Bill Dunlap, David Gee, Arthur van Hoff,
Brian Maso, Miko Matsumura Kim Polese,
Sean Rhody, Rick Ross, Richard Soley, George Paolini

Editor-in-Chief: Sean Rhody
Art Director: Jim Morgan
Executive Editor: Scott Davison
Managing Editor: Anita Hartzfeld
Senior Editor: M'lou Pinkham
Editorial Assistant: Brian Christensen
Technical Editor: Bahadır Karuv
Visual J++ Editor: Ed Zebrowski
Visual Café Pro Editor: Alan Williamson
Product Review Editor: Jim Mathis
Games & Graphics Editor: Eric Ries
Tips & Techniques Editor: Brian Maso

WRITERS IN THIS ISSUE

Juergen Brendel, Andrei Cioroiaru, Joseph Cozad
Claude Duguay, Matt Evans, Ken Jenks, David Jung,
George Kassabgi, Jim Milberry, Thomas Murphy, Paul Petersen,
Patrick Ravenel, Sean Rhody, Sriram Sankar, Alan Williamson

SUBSCRIPTIONS

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department

Subscription Hotline: 800 513-7111

Cover Price: \$4.99/issue.

Domestic: \$49/yr. (12 issues) *Canada/Mexico:* \$69/yr.
Overseas: Basic subscription price plus air-mail postage
(U.S. Banks or Money Orders). *Back Issues:* \$12 each

Publisher, President and CEO: Fuat A. Kircaali
Vice President, Production: Jim Morgan
Vice President, Marketing: Carmen Gonzalez
Advertising Manager: Claudia Jung
Advertising Assistants: Robin Forma
Jaclyn Redmond
Accounting: Ignacio Arellano
Graphic Designers: Robin Groves
Alex Botero
Webmaster: Robert Diamond
Senior Web Designer: Corey Low
Customer Service: Sian O'Gorman
Paula Horowitz
Online Customer Service: Mitchell Lowe
Customer Service Interns: Angela Frasco
Ann Marie Millio

EDITORIAL OFFICES

SYS-CON Publications, Inc.
39 E. Central Ave., Pearl River, NY 10965
Telephone: 914 735-1900 Fax: 914 735-3922
Subscribe@SYS-CON.com

JAVA DEVELOPER'S JOURNAL (ISSN#1087-6944) is
published monthly (12 times a year) for \$49.00 by SYS-CON
Publications, Inc., 39 E. Central Ave., Pearl River, NY 10965-2306.
Application to mail at Periodicals Postage rates is pending at
Pearl River, NY 10965 and additional mailing offices.

POSTMASTER: Send address changes to:

JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc.,
39 E. Central Ave., Pearl River, NY 10965-2306.

© COPYRIGHT

Copyright © 1998 by SYS-CON Publications, Inc. All rights reserved. No part of this
publication may be reproduced or transmitted in any form or by any means, electronic
or mechanical, including photocopy or any information storage and retrieval system,
without written permission. For promotional reprints, contact reprint coordinator.
SYS-CON Publications, Inc. reserves the right to revise, republish and authorize
its readers to use the articles submitted for publication.

ISSN # 1087-6944

**Worldwide Distribution by
Curtis Circulation Company**

739 River Road, New Milford NJ 07646-3048 Phone: 201 634-7400

Java and Java-based marks are trademarks or registered trademarks
of Sun Microsystems, Inc. in the United States and other countries.
SYS-CON Publications, Inc. is independent of Sun Microsystems, Inc.
All brand and product names used on these pages are trade names,
service marks or trademarks of their respective companies.

**SYS-CON
PUBLICATIONS**

Sean Rhody, Editor-in-Chief



Why Superman Works Alone

Toward the end of the last Batman movie, when Robin is giving Batman a hard time, George Clooney gets fed up and says, "This is why Superman works alone." While I'm often tempted to think along the same lines, the reality of our business is that we work in teams. This leads to the topic of this month's diatribe: team development.

Large-scale software development is a complex process. The majority of it takes place in a corporate environment that requires rigor and process. The most familiar of these processes is usually the task of obtaining the blessing of the DBA for your database schema, or for changes you need to make to it. Due to the nature of the corporate database, the DBA process is almost an island unto itself. I'll rant about that some other time because right now I'm interested in team development and its partner - configuration management.

It's my belief that the companies creating source code control tools today are missing the big picture. You'll notice that I said "configuration management" in the previous paragraph and "source code control" in this one. In my mind there's a big difference between them; while source code control is vital to any project and is indeed part of configuration management, it is by no means the whole picture. Mention configuration management to a typical project team and you'll get an answer like, "Oh yeah, we use..." (which can be filled in with any of the current generation of source code control tools).

Unfortunately, this misses the point of configuration management because it addresses only a small portion of the entire task - the part relating to developers. The typical configuration-management dilemma encompasses tasks such as unit, system and integration testing; software builds and deployment; and, in the era of multitiered Web applications, distribution and registration issues.

We need to establish a set of standards for these practices so they can be automated and controlled. It's a complex issue, but that's the whole point of paying a good deal of money to a vendor to come up with a solution. Every time we begin a new project we have to tackle the same issues over and over. This is an opportunity for an up-and-coming ISV to create a new product line and dominate the industry.

What we need is a meta tool - one that's language-independent but can interact with any of the major development environments. It could start with source code control. Most organizations I've worked with don't allow developers to directly create executables -

as part of the QA process the testing team usually creates them. And they typically suffer from a lack of expertise in doing just that because their main focus is on testing, not building. The first step would be automating the build process, then representing it in a graphical manner that makes it easy to define and automate. Please don't tell me about nmake or other command line utilities - it's been my experience that people who are typically charged with the build process aren't the kind of heads-down developers who can make that kind of tool workable. A simple graphic tool is needed.

The next step is to establish standards for code deployment and registration. Most multitiered environments use some sort of application server, be it MTS, Jaguar CTS, Tuxedo/M3 or one of the CORBA Orbs. As code is produced, and as it changes, redeployment and registration of components within these servers become an issue. The establishment of a common mechanism for supporting this process would enable our new meta tool to automate this process. I spent a great deal of time on a recent project ensuring that registration and CLSIDs were current. It was largely a mechanical process that didn't need to eat up a lot of time if a tool had been available.

Hand in hand with the application server deployment is the replication of environments. In most cases the code above goes to more than one application server (development, testing, various production systems, etc.). Perhaps a publish-and-subscribe paradigm, with the meta tool being the publisher, would be appropriate.

Finally, deployment and synchronization of the client code is still an issue. Fortunately, we've had more time to think about this one, thanks to client/server, and there are tools available to help. Unfortunately, none that I'm aware of tie into a more global model, so even with these tools you have some type of routine setup and maintenance.

I'm not holding my breath waiting for this meta tool. Having wrestled with these issues for years, I know how difficult it is to establish a set of vendor-neutral standards and get the industry to adhere to them. But I do know we need them. In the meantime, I'm going to take my Kryptonite laptop somewhere quiet and do some real coding. Up, up and away. ☺

About the Author

Sean Rhody is editor-in-chief of *Java Developer's Journal*. He is also a senior consultant with *Computer Sciences Corporation* where he specializes in application architecture, particularly distributed systems. You can contact Sean at sean@sys-con.com.

Zero G InstallAnywhere

<http://www.zerog.com/jdj>

CALL FOR SUBSCRIPTIONS

1 800 513-7111

International Subscriptions
& Customer Service Inquiries
914 735-1900

or by fax: 914 735-3922

E-Mail: Subscribe@SYS-CON.com
<http://www.SYS-CON.com>

MAIL All Subscription Orders or
Customer Service Inquiries to:

JAVA DEVELOPER'S JOURNAL

Java Developer's Journal
<http://www.JavaDeveloperJournal.com>

PowerBuilder DEVELOPER'S JOURNAL

PowerBuilder Developer's Journal
<http://www.PowerBuilderJournal.com>

COLD FUSION DEVELOPER'S JOURNAL

Cold Fusion Developer's Journal
<http://www.ColdFusionJournal.com>

VRML DEVELOPER'S JOURNAL

VRML Developer's Journal
<http://www.VRMLDevelopersJournal.com>

POWERBUILDER 6.0

Secrets of the PowerBuilder Masters
<http://www.PowerBuilderBooks.com>

EDITORIAL OFFICES

Phone: 914 735-7300
Fax: 914 735-3922

ADVERTISING & SALES OFFICE

Phone: 914 735-0300
Fax: 914 735-7302

CUSTOMER SERVICE

Phone: 914 735-1900
Fax: 914 735-3922

DESIGN & PRODUCTION

Phone: 914 735-7300
Fax: 914 735-6547

WORLDWIDE DISTRIBUTION by
Curtis Circulation Company

739 River Road, New Milford, NJ 07646-3048
Phone: 201 634-7400

DISTRIBUTED in the USA by
International Periodical Distributors

674 Via De La Valle, Suite 204
Solana Beach, CA 92075
Phone: 619 481-5928

SYS-CON PUBLICATIONS

Thomas Murphy



When Do We Reach Nirvana?

In some form or another it seems that this question is on the minds of every programmer, manager and CIO in the world. Each day another hundred monkeys introduce the new savior of the world, with the promise of previously unknown productivity and an end to all programming problems.

Will you be saved by JINI? Find your place in the sun with JavaSpaces? Or will HotSpot be your ticket to programming paradise? I would venture that none will do the trick because they're just technologies. Yes, they may make aspects of the job better but they won't put you into a fundamentally new state of programming happiness. To really move forward requires process and knowledge.

Just as information is the most important asset of a company, it's the most important asset you possess as a programmer. As you may have learned watching *Schoolhouse Rock* on Saturday mornings, "Knowledge is power." You can have the most expensive rod and a tacklebox filled with a hundred flies but if you don't know how to cast you won't catch a fish. Even if you can cast, without studying the streambed and looking for other important indicators you'll only end up with tired arms and sore feet.

The Gartner Group elucidates Java's main problem that no technology can solve: "A severe shortage of skilled Java developers is placing new Java projects in jeopardy in more than 30% of AD organizations making a transition to the new language and platform" (Application Development and Management Strategies research note, 28 May 1998). It points out that it really doesn't make any difference if you are a COBOL, VisualBasic or C++ developer. It doesn't matter, per se, that you've been fishing before and know how to cast with tackle and worms - distributed object computing, or fly-fishing, is a completely different art.

Like fly-fishing, you can't simply read a book and walk out to the stream an expert. To really learn and excel you must work with another person who knows the stream and the craft. I've always pushed back on the notion of software engineering. It's not that I don't believe in the use-

fulness of modeling tools, good documentation and coding practices; it's more that I don't think you can learn all you need to know about programming by simply reading a book and learning the standard templates. These are the things that come with experience and, in effect, are best passed down through an "oral tradition."

While the technological advances in Java will make it easier to do things like create distributed applications, the developer still has to think through the application architecture. A good example, ease of use, that many have seen leading to poor results is in the area of threads. Creating multithreaded applications is a snap with Java. However, spawning threads without thought will lead to failure, and new issues like transactional integrity and non-uniform VM implementations will arise.

Fortunately, the road we travel is one that's been traveled before. Distributed CORBA-based Smalltalk systems are running major business systems around the world. CORBA technologies have evolved through several iterations to become portable and cover core architecture, design and base business objects. There are good books on fundamental patterns, tools that help you understand object relationships and full-spectrum training to build your core techniques. The waters have been fished before and there are great mentors who can amplify this book knowledge with the wisdom of experience.

Now is a great time to gain experience and learn to read the waters. While mainstream IS fixes and tests Y2K problems, you can learn good distributed architecture. Find a guide who knows the waters and has seen the flies that work and the ones that don't. Then your tacklebox will not only be filled with great-looking tools, but you'll be able to apply them with skill - and you'll have entered a higher programming plane. ♦

About the Author

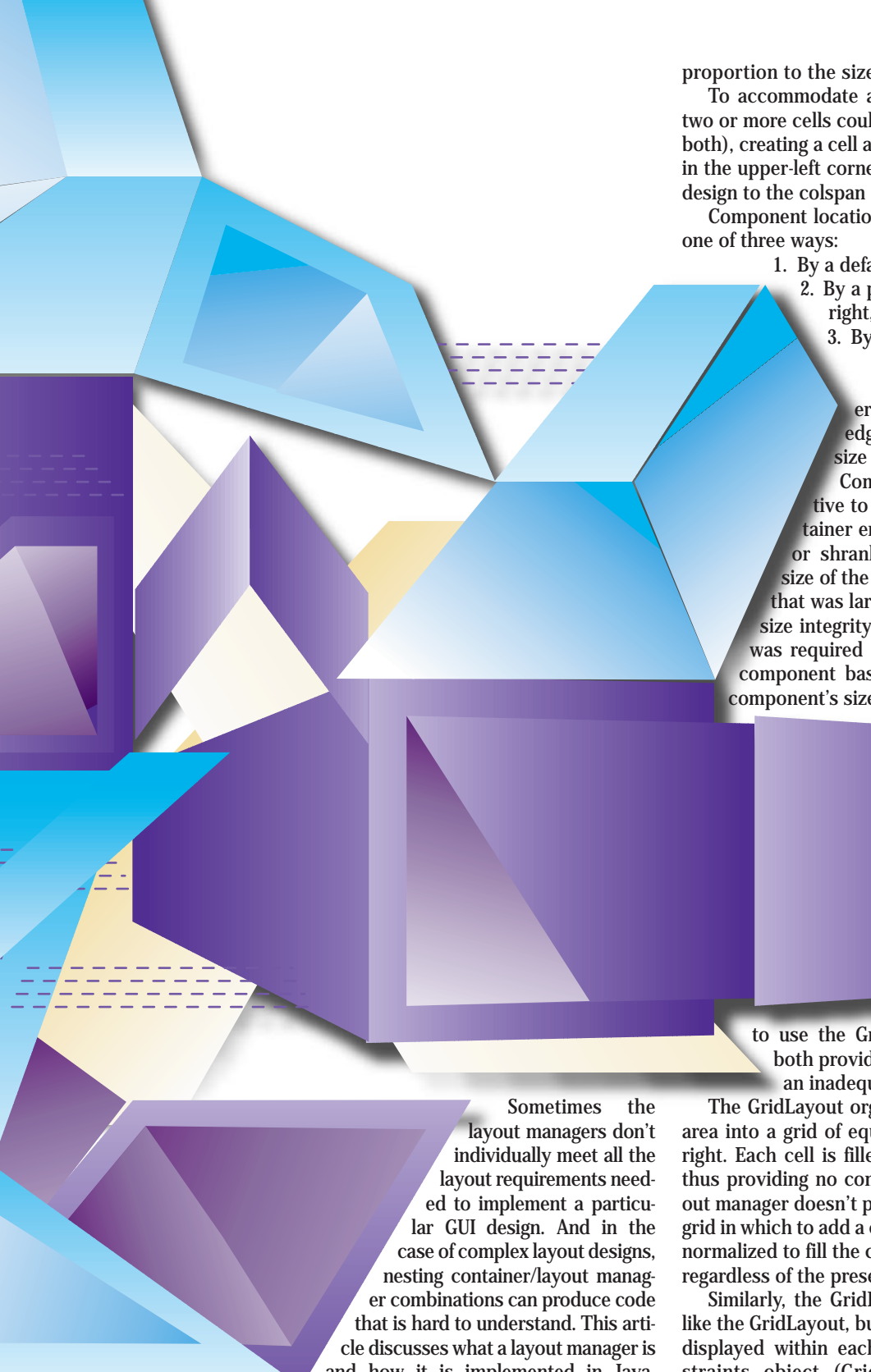
Thomas Murphy is the director of marketing for ObjectShare. He's acted as the product manager for ObjectShare's PARTS for Java and Visual Smalltalk product lines. You can e-mail him at tmurphy@objectshare.com.

Creating a Custom Layout Manager

*Dealing with specific
problems when your
layout manager can't*

by Joseph Cozad

A layout manager is an object that positions and resizes components within a display area according to a specific algorithm. The Java 1.2 AWT package provides 10 layout manager classes that can be used to accomplish this task. Each has a defined set of behaviors that organize components in a container. Each Java container instance is associated with an instance of one of these layout managers. By nesting one container/layout manager combination within another one, complex screen layouts can be implemented.



proportion to the size change.

To accommodate a component larger than an individual cell, two or more cells could be combined horizontally or vertically (or both), creating a cell area addressed by the identity of the first cell in the upper-left corner of the area. This functionality is similar in design to the colspan and rowspan attributes in HTML tables.

Component locations within a cell area need to be specified in one of three ways:

1. By a default value (the center of the cell area)
2. By a predefined general location, such as top left, right, bottom right
3. By a user-defined point within the cell area

In addition, components within the container could not lose their positions relative to the edges of the container when the container's size changed.

Components needed to maintain their size relative to the size of the container so that as the container enlarged or shrank, the components enlarged or shrank proportionally. Also, the specified initial size of the component could not be altered to fill a cell that was larger than the component; that is, component size integrity had to be maintained. The layout manager was required to provide a way to calculate the size of a component based on cell area size to assist in setting a component's size relative to the size of the container.

No limitation was placed on the number of components that could be added to a cell area. This meant that components could be overlapped and that overlapping should be done in a back-to-front manner. That is, component B added to the same container cell area as, and after, component A would lie on top of component A.

Design Choices

Because the requirements describe a method for laying out components based on a grid pattern, a designer might be led to use the GridLayout or GridBagLayout classes. While both provide a grid system, their limitations make them an inadequate solution for these requirements.

The GridLayout organizes components by dividing the display area into a grid of equal-sized cells numbered 0 to n from left to right. Each cell is filled in succession with only one component, thus providing no component overlap functionality. The GridLayout manager doesn't provide a way to address a specific cell in the grid in which to add a component. In addition, component sizes are normalized to fill the cell so that each cell maintains an equal size, regardless of the preset component size.

Similarly, the GridBagLayout organizes components into cells like the GridLayout, but provides greater flexibility in how they are displayed within each cell area by associating a separate constraints object (GridBagConstraints) with each component. Through the value manipulation of the GridBagConstraints object, each component can occupy more than one cell at a time. This is not a simple process, however, because it relies on a coordinate system rather than a single-cell ID number. For example, to add a component to a panel that implements a GridBagLayout manager, the programmer sets the instance variables, gridx and gridy, of the associated GridBagConstraints Object.

Although this manager provides a mechanism for a component to span more than one cell, it doesn't provide component overlap or a simple solution for defining where in a cell area the component

Sometimes the layout managers don't individually meet all the layout requirements needed to implement a particular GUI design. And in the case of complex layout designs, nesting container/layout manager combinations can produce code that is hard to understand. This article discusses what a layout manager is and how it is implemented in Java.

Through a real-world example it demonstrates how a layout manager can be created to address any GUI layout situation without the need to nest container/layout manager combinations.

Layout Requirements

The GUI design for the example required that component locations be specified using a grid system of cells in which each cell was easily identifiable. The cells needed to be the same size and immutable, so an individual cell's size could be altered only if the container's overall size were also altered, and then only in

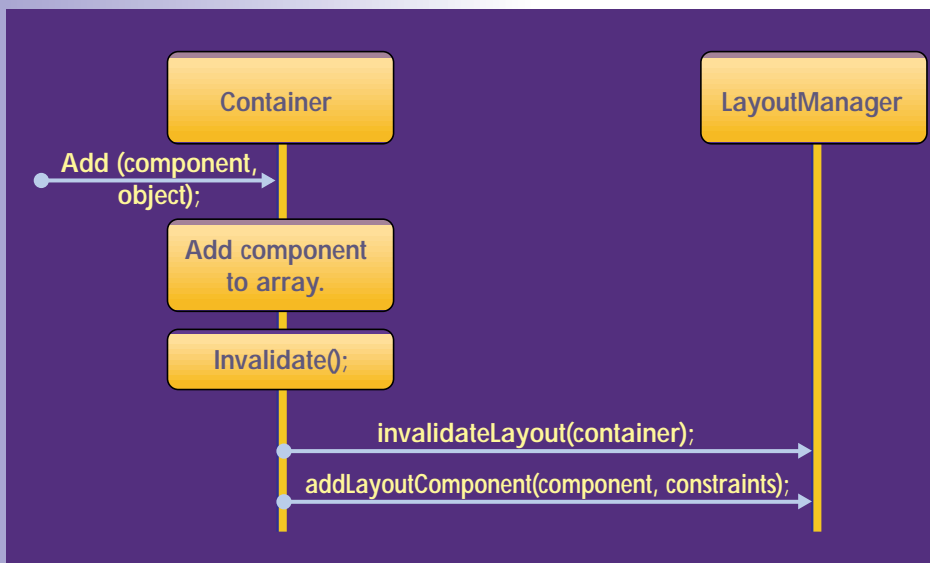


Figure 1: Adding components scenario diagram

should be located. Also, the GridBagLayout resizes components based on a combination of its associated constraints object, the minimum size of the component and the preferred size of the container. Setting a specific size for a button object can be overridden by the size calculated by the layout manager using the constraints object.

Java 1.2 introduces a new layout manager class called an OverlayLayout, which lays out a component based on the component's specified alignment value. While this class allows components to overlie each other, the programmer must set each component's alignmentX and alignmentY values so that the manager can calculate the component's size and location relative to the size of the enclosing container.

For this example, creating a custom layout manager solution centering around the general functionality of these three layout managers and incorporating the simplicity of HTML tables seemed appropriate. So how is this done? To begin designing a custom layout manager, it is first necessary to understand how it works and when.

Layout Manager Functionality

Java container classes in themselves provide no functionality supporting the positioning of components. Sun has separated this functionality into a set of layout manager classes. In turn, a layout manager controls only *where* components are displayed within the container, not what components are displayed. In effect, the container relies on the layout manager to place the components while the layout manager relies on the container to know what components are to be placed.

In Java, an object becomes a layout manager by implementing the methods declared in the java.awt.LayoutManager or

java.awt.LayoutManager2 interfaces or both. The latter interface is new to Java 1.1, and extends the LayoutManager; no changes were made to these interfaces in Java 1.2. These interfaces describe 10 method signatures (behaviors) necessary for the implementing object to arrange components within a container. These behaviors can be grouped into six types:

- Adding components
- Removing components
- Container layout
- Invalidation
- Size definition
- Location definition

The first three types are the minimum behaviors needed to create a layout manager's functionality. In designing the add and remove behaviors, the most important consideration is that the layout manager needs to maintain and associate layout information on each component object maintained by the container object.

The following describes each group, what the associated methods do and when they are called by a container object. I assume the reader is familiar with how a layout manager is used by a container object, and knows that a Java applet or application never calls any of these implemented interface methods directly, but leaves these calls to be implemented by the container object itself.

Adding Components

When a component is added to a container using one of the container's add methods, the container adds a new component to the end of an array (n-1) of components that it maintains. The container then invalidates itself and calls one of the two registered layout managers' addLayoutComponent(); methods (see Figure 1).

These methods define how the layout manager will track the components that are added to the associated container:

```

public void addLayoutComponent(String position, Component comp);
public void addLayoutComponent(Component comp, Object constraints);
  
```

The layout manager should maintain the information passed by the string or constraints object to use when the container needs to be laid out (see "Container Layout"). This could be done with an array, vector or hash table.

The first method definition takes a String object that acts as a positional modifier indicating how or where the component should be laid out. For example, when a container object implements a BorderLayout object as its layout manager and a component is added to the center region of the container, the method Container.add("Center", comp); uses the string "Center" as a positional modifier.

The other method definition supports constraint-based layout management, which assumes that each component added to the container is associated with a separate constraint object that specifies how the component will be laid out. An example of this is seen in the GridBagLayout and the associated GridBagConstraints classes.

Removing Components

When the container's remove(Component comp); or removeAll(); method is called, the registered layout manager's removeLayoutComponent(); method is called. This method removes the corresponding component reference and any associated modifiers from the layout manager's stored references:

```

public void removeLayoutComponent(Component comp);
  
```

The container also removes the component object from the list of components it maintains. After this method has been called, the container calls the layout manager's invalidateLayout(); method (see "Container Invalidation").

Container Layout

When the container's doLayout(); method is called, the registered layout manager's layoutContainer(); method is called. This method executes the algorithm that sets the x and y coordinates' width and height of each component contained in the targeted container object:

```

public void layoutContainer(Container parent);
  
```

EnterpriseSoft

Report Writer for Java

<http://www.enterprisesoft.com>

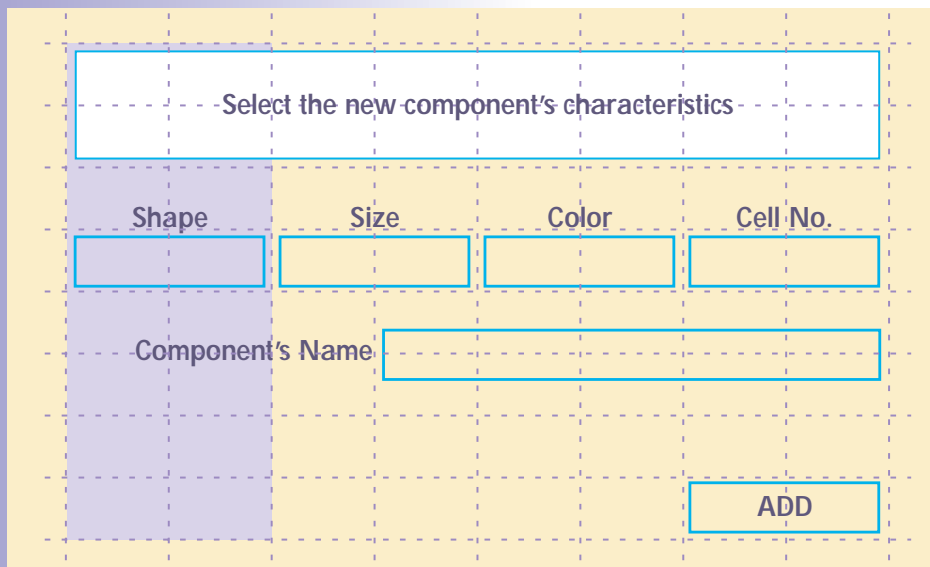


Figure 2: TestCustomLayout Add Component Screen grid

The algorithm designed to determine these values varies depending on what is required by the design of the layout manager. However, regardless of how the algorithm is designed, the following processes should happen:

1. Retrieve a list of components maintained by the container using the `Container.getComponents()` method.
2. Test if the component is visible.
3. Associate each visible component with its constraints, maintained by the layout manager.
4. Generate the correct x,y width and height values for the component.
5. Pass these values to the component's `setBounds()` method. Note that the container size may have changed since the last time it was laid out.

After the container calls the layout manager's `layoutContainer()` method, the container draws the components on the screen. The container does this by cycling through the container's array of components beginning at 0 and ending at $n-1$. When trying to facilitate overlapping, it is important to add the component to the beginning of the array, that is, index location 0. This can be accomplished by calling one of the container's `add` methods that includes an integer index value as one of its arguments, such as `add(Component comp, Object obj, int index)`. By specifying a different index value, the component can be inserted between two overlapping components.

Container Invalidation

Container invalidation occurs when the container's `setLayout()`, `add()`, `remove()`, or `removeAll()` method is called. The container is marked as invalid and the registered layout manager's `invalidateLayout()`

method is then called. This method can define what actions to take, if any, when a container is marked invalid:

```
public void invalidateLayout(Container target);
```

Defining Size

When the container's `getPreferredSize()`, `getMinimumSize()`, or `getMaximumSize()` is called, and if the container doesn't already have size information stored, the registered layout manager's `preferredLayoutSize()`, `minimumLayoutSize()`, or `maximumLayoutSize()` method is called.

These methods can optionally define what the container's size should be after laying out all the objects and taking into consideration the container's insets:

```
public Dimension minimumLayoutSize(Container parent);
public Dimension preferredLayoutSize(Container parent);
public Dimension maximumLayoutSize(Container parent);
```

There's no guarantee that any one of these methods will be called before the `layoutContainer()` method (see "Container Layout").

Defining Location

When the container's `getAlignmentX()`, or `getAlignmentY()` is called, the registered layout manager's `getLayoutAlignmentX()`, or `getLayoutAlignmentY()` is called. These methods can optionally define a percentage value representing the distance that a component should be from the edge of a container:

```
public float getLayoutAlignmentX(Container target);
```

```
public float getLayoutAlignmentY(Container target);
```

At minimum, a default return value of 0.5f for centered should be supplied.

Designing the RelationalGridLayout Manager

The resulting class from the example's design is the `RelationalGridLayout`. The `RelationalGridLayout` class, like the `GridBagLayout`, uses an associated constraints object (see `CellConstraints.java` Code) for each component added to the container. In addition, a `LocationManager` class was created to define nine locations within a cell area (see `LocationManager.java` Code) as public final static int types.

Unlike the `GridBagConstraints` class that gives direct exposure to its instance variables, the `CellConstraints` class contains four constructors that initialize a set combination of instance variables. This design allows for only four possible ways to define how the component should be laid out.

Each constructor takes an integer that indicates the cell number that contains the upper-left corner of the component. Three of the constructors take two integers, each representing the number of cells the component will span horizontally and vertically. These two values are used to represent the cell area in which the component will be placed. One of the constructors takes a `java.awt.Point` object that indicates the x and y coordinates of the component's upper-left corner within the cell area, and one of the constructors takes an integer that represents a predefined location within the cell area using one of several `LocationManager` values.

The following paragraphs discuss key points in the `RelationalGridManager`'s implementation (see code listing).

The Constructor

To understand how the grid is represented, the constructor needs information about the targeted container's size, the number of columns and the number of rows. This information is used to initialize the instance variables, calculate the cell size and create a hash table used to hold the component layout information. Note that component layout information is stored in a private class object called `ItemInfo` (lines 7-19).

The Layout Manager's Behaviors

This design requires the implementation of only three of the 10 behaviors (the remaining are implemented either with empty bodies or with default return values; lines 43-50), which include:

KL Group

Java Beans

<http://www.klg.com>

Adding Components

The `RelationalGridLayout` implements the `LayoutManager2` interface to support constraint-based modifiers. As such, it only implements the `addLayoutComponent` method that takes a reference to the component added by the container and `Object` (line 81). After testing for validity, the `Object` is cast to a `CellConstraints` type and assigned to a method variable (lines 87–89). In line 91 the size of the component is retrieved next and also saved in a method variable.

After the method variables have been assigned, the point values for the component's location are calculated (lines 94–111). If a point within the cell area is specified, the `calcCellInset()` method is called (lines 185–192). This method takes the point, the cell area size, and the component's size to calculate the distances of the component's edges from the cell's edges. If, instead, a `LocationManager` value is specified, the `calcInsets()` method is called (lines 203–278) to determine these same values based on the `LocationManager` value supplied. Once the component's insets are calculated for the cell area defined by the column and row values, the x,y coordinate or point value within the container's dimensions is calculated using the cell ID number and the component's insets (lines 98, 100 and 108).

Next, in lines 115–118, the method takes the calculated point and the component's size and converts this information into proportional ratios. These ratios are stored in an `ItemInfo` object and added to the instance's hash table keyed on a reference to the component (line 121).

Removing Components

No special processing is needed here other than to remove the component's display information from the hash table (lines 124–129).

Laying Out Components

The code to lay out the components in the container begins at line 131 by first determining if the container has changed in size since the last time the container was laid out. Line 136 compares the saved container's size (`PanelSize`) with the current size of the container retrieved in line 135. If the size has changed, the new size is saved and a new cell size is calculated.

Next, in line 142, a reference to the container's array of components is retrieved. Then the method loops through the array retrieving the component's modifier information from its associated `ItemInfo` object, converts it from ratios to actual pixels based on the current size of the container

and then sends the x,y width and height information to the component using its `setBounds()` method (lines 143–153).

The rest of the `RelationalGridLayout` implementation contains helper methods that assist in performing some of the calculations needed by the behaviors described. Two useful methods to note are:

1. **Calculating Cell Area Insets.** The `calcInsets()` (lines 221–296) method uses the supplied `rowspan` and `colspan` values to determine the size of the cell area in which the component will be located. Then, using the supplied `LocationManager` index value, the appropriate algorithm is selected to calculate and return the component's inset value for the cell area.

2. **Calculating Cell Area.** The `calcArea()` method (lines 54–59) is a public method used both internally and externally to assist in determining cell area size. By using this method as a public instance method, the size of the component can be preset relative to the size of the cell area in which it will be located.

Using the RelationalGridLayout

The `TestCustomLayout` applet demonstrates how the `RelationalGridLayout` is used. It also provides a way to interactively try out `RelationalGridLayout` functions like overlaying components, adding components to a specific place on the grid and maintaining the component's location when the container size changes. The `TestCustomLayout` applet's `Add Component Screen` is one example of how to use the `RelationalGridLayout`.

First, a sketch of the screen was created (see Figure 2). Then a grid pattern was derived by identifying the smallest element in the sketch; in this case the shape, size, color, cell number labels and their associated input fields were used to identify the size of the smallest cell. From this, an 8 x 8 grid pattern was created.

The `createPanel()` method in the `AddCompPanel` class provides the source defining the representation of the `Add Component Screen` (see `AddCompPanel.createPanel` Code). Much of the code is taken up by creating the individual components and specifying their characteristics. Line 6 creates the `RelationalGridLayout` object supplying the size of the panel and the number of columns and rows in the grid pattern. Notice that a method variable is created to hold the reference to the layout manager, as opposed to creating the reference directly in the `setLayout()` method call in line 7. This was done so that we can use the layout manager's `calcArea()` method to specify and preset component sizes.

Lines 10–104 create the instances of

each component and set their characteristic values. Each component's size is set directly because the layout manager doesn't calculate the size, but calculates the component's proportional size (proportional to the container). Lines 17, 25, 28 and 96 use the layout manager's `calcArea()` method to set these sizes relative to the sizes of their intended cell areas. For the shape, size, color and cell number labels; their fields; and the add button, a method variable (line 28) is assigned with the size calculated by the layout manager's `calcArea()` method as two cells wide, one cell high, with a 10-pixel inset, just as the sketch shows.

Once created, the components are added to the container using the panel's `add(Component comp, Object obj, int indx)` method (lines 105–126). With each add a `CellConstraints` object is created, passing the cell number, the number of columns the component will span horizontally, the number of rows the component will span vertically and a location within the cell area. Some components have a value of 1 for either the column or row spans. The `RelationalGridLayout` does not assign a valid default value for these arguments. Also, the index value passed to the add method is 0, thus placing each component at the beginning of the container's component array.

Conclusion

Creating a custom layout manager in Java is a simple process that consists of implementing the appropriate layout manager interface that supports the container add methods that will be used, deciding which behaviors the layout manager will support and designing the algorithms that will calculate and track each component's location within the container. When one of the Java AWT default layout managers does not solve all the layout requirements, creating a custom layout manager may be the solution. ☛

▼▼▼▼▼ CODE LISTING ▼▼▼▼▼
The complete code listing for this article can be located at www.JavaDevelopersJournal.com

About the Author

Joseph Cozad is the Java technologist for Motorola's Advanced Media Platforms in Austin, Texas. As a Sun Microsystems certified Java developer, he serves as Motorola's corporate knowledge champion on Java technology. Prior to being a Java technologist, he established and coordinated online interactive systems. You can contact him at rzxd50@email.sps.mot.com.



rzxd50@email.sps.mot.com

Sun Test

Java Testing Solutions

<http://www.suntest.com>

What AWT VERSION Do You Use?

*Force components to
use 1.1 and discover
those that use 1.0*

by Andrei Cioroianu

We all know what AWT is. We know that AWT 1.0 is simple and easy to use. It's compatible with the old versions of the Web browsers, but has an inefficient event-handling model. The events are received by the components, which handle or ignore them. Worse, if you want to add a small functionality to a component, you have to create a new class of components. AWT 1.1 corrects these defects. It's a little more complex, but has many advantages. The new delegation-based event model plays a key role in the JavaBeans architecture as it facilitates the communications between beans. Design patterns allow the development tools to use reflection to become smarter. With AWT 1.1 you can create lightweight components (like those of Swing) that are serializable so you can create persistent user interfaces. The delegation-based event model allows programmers to isolate the application logic from the user interface and improve UI performances of the applications.

AWT 1.1 has to struggle to offer compatibility with AWT 1.0. A component may use the old event model or the new one, but not both at the same time. The documents from Sun advise you not to mix event models in the same application. How can you believe that advice if all AWT 1.1 components are born as AWT 1.0s? (I'm talking about instances, not classes.) The event model of AWT 1.0 is the default event model of the AWT 1.1 components. Each component has a flag - `newEventsOnly` - that indicates the type of event model used by the respective component. The flag is initialized false when the component is created. When you register listeners to a component, the `newEventsOnly` flag is switched to true. Only after that occurs will the component use the AWT 1.1 event model. The transformation is irreversible (see Figure 1).

If you're using Java 1.1 or a later version, your components have the potential to become AWT 1.1 components. Most of them die as



AWT 1.0 components, however, because the programmers don't register listeners to them. Why register a listener to a TextField when you have a getText() method? Why register a listener to a Checkbox when you have a getState() method?

If you're using Java 1.1 or 1.2 and you don't call deprecated methods of AWT 1.0, then you can benefit from all the advantages of AWT 1.1. If you're not careful, however, many of your components will use the AWT 1.0 event model. Your app will work fine but won't run at maximum speed. This article shows you how to force all the components to use the AWT 1.1 event model.

How to Improve UI Performance

You should ask yourself first whether your apps need improved UI performance. Most of the simple applets run at the same speed no matter what event model they use. But suppose you're working on a complex application that places components in a slow container – to create compound documents, for example. You'll want all of the events addressed to the components not to be passed to the container.

Let's consider a simple example. I wrote a small applet whose area is covered by only one component (a java.awt.Canvas instance). The init() method of the applet creates the Canvas object and stores a reference to it in the c member variable.

```
public void init() {
    setBackground(Color.red);
    c = new Canvas();
    setLayout(new BorderLayout());
    add(c, "Center");
}
```

This applet is a slow container. Each time the mouse is moved within the applet's area, the status bar is modified a hundredfold. This operation proved slow enough for most of my browsers (HotJava 1.1, Navigator 4.05 and AppletViewer). SystemMonitor of Win95 showed that my processor, a Pentium MMX 166, was 100% busy. Still, Internet Explorer 4.0 was very fast at the redraw of the status bar, so I had to increase the limit of the i counter at 1,000. If you have a very fast machine you may grow to this limit, but be careful because the applet might freeze your browser.

```
public boolean mouseMove(Event evt, int x, int y) {
    for (int i = 0; i < 100; i++)
        showStatus(Integer.toString(x * y + i));
    return true;
}
```

The applet's entire area was covered by the Canvas object, and the mouse events received by this component were sent further – to the applet. This means that the Canvas instance uses the AWT 1.0 event model. However, if I click on the canvas, the status bar won't modify anymore (mouseMove() isn't called anymore) because the applet intercepts the mouse click and calls the addComponentListener() method of the Canvas instance from the mouseDown() method. Although undocumented, this is the only way to force the components to use the AWT 1.1 event model.

```
public boolean mouseDown(Event evt, int x, int y) {
    c.addComponentListener(null);
    return true;
}
```

No listeners are registered to the Canvas object because the parameter of the addComponentListener() method is null. However, this method will set the internal newEventsOnly flag of the component to true. From this moment the applet won't receive any mouse events because its entire area is covered with an AWT 1.1 component.

The Canvas object was born as an AWT 1.0 component and became an AWT 1.1 component after the "c.addComponentListener(null)" call. This is a good trick that allows you to force the AWT components to use the delegation event model even if no listeners are registered to them. There is no correct way to transform an AWT 1.1 component into an AWT 1.0 component. The applet not only uses the AWT 1.0 event model for its entire existence, but it also overrides deprecated methods.

Find Out What Event Model Your Components Use

The only way to determine what event model is used by a particular component is to examine the newEventsOnly flag, which isn't public. How can you do that in the absence of a getNewEventsOnly() method? Fortunately, the flag is not private. It's friendly, so you may write your own AWT class whose methods can read newEventsOnly. You can do this without altering your Java Virtual Machine (JVM), as long as you write stand-alone applications. The trick doesn't work with Navigator and Internet Explorer without breaking their security mechanisms, and I don't know how to do that. After all, I'm just a Java developer, not a professional hacker. You also must know that Java 1.2 has a better security mechanism than 1.1. If you want to run the DemoCompTree application with JDK 1.2 beta 4, you'll have to



use the `-Xverify:none` option of the Java interpreter. Otherwise, an `IllegalAccessException` will be thrown (source code will be explained later).

I named the new AWT class `CompTree` (see Listing 1). You don't need to place `CompTree.class` in the `classes.zip` file. Just insert the `"package java.awt;"` declaration at the beginning of the source file and, after you compile it, copy the classfile in your `_dir/java/awt`. Then add your `_dir` directory at `CLASSPATH` and you'll be able to use the `CompTree` class like any other AWT class (your `_dir` might be the current directory `."`). This is possible because the JVM is locking after system classes in all of the directories and `.zip` files from `CLASSPATH`. The JVM makes no distinction between `java/awt/Component.class` from `classes.zip` and `java/awt/CompTree.class` from your `_dir`. There's nothing to stop you from splitting the classes of a package into more than one directory.

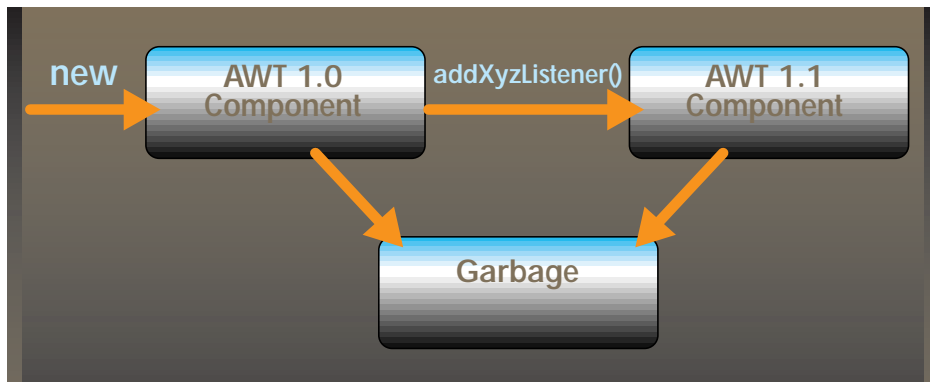


Figure 1: The life cycle of AWT components

The `CompTree` class has only one public method (`printTree()`) that takes a `java.awt.Container (c)` as argument and prints (to console) the entire component hierarchy, whose top is referenced by `c`. Note that this hierarchy has nothing to do with the class hierarchy of the `java.awt` package. The public `printTree()` method calls the private `printInfo()` and `printTree()` methods.

The `printInfo()` method (see Listing 1) shows information about the component whose reference is passed as parameter. It prints the name of the class of the component (which is returned by the `getName()` method) and the value of the `newEventsOnly` flag. `True` means that the component uses the AWT 1.1 event model. `False` indicates that the component uses the old event model (of AWT 1.0). If a component is a container (`java.awt.Container` is a subclass of `java.awt.Component`), then the name of the `LayoutManager` class is printed. More information about the state of the component could have been obtained with the `toString()` method.

The private `printTree()` method (see Listing 1) gets the list of the components of the container (`c` is a parameter variable) with the help of `getComponents()`. The `printInfo()` method is called for each component. The `printTree()` method is called recursively for those components that are also containers. This way, the entire component tree is printed. An integer parameter is used for indenting.

What Is the `CompTree` Class Good For?

You already know the answer. You can use it to identify the components of a container that doesn't use the AWT 1.1 event model because they don't have listeners, and then force them to use it. You can do that, for example, by calling the `addComponentListener()` method with a null parameter, as I described earlier.

Next I'll show you how to use the `CompTree` class as a tool. In my "Persistent User Interface for Multiuser Apps" article

(see Listing 1) gets the list of the components of the container (`c` is a parameter variable) with the help of `getComponents()`. The `printInfo()` method is called for each component. The `printTree()` method is called recursively for those components that are also containers. This way, the entire component tree is printed. An integer parameter is used for indenting.

The `SmartLogin` app is fast enough because it's simple. If I force the nine components that use the old event model to work with the new one, the increase in speed is imperceptible so the source code should remain unchanged. For a real-world, complex application, however, the techniques presented in this article may be useful.

Note that in last month's article the constructor of the `SmartLogin` class was private. This made sense because the app was creating only one `SmartLogin` instance in the `main()` method (`SmartLogin` was a singleton). For this article I had to delete the private keyword to be able to study the component tree of the `SmartLogin` container with the help of `CompTree`.

`CompTree` can also be used as a utility class to verify or study the component hierarchies that you coded manually or built visually with a development tool. You don't necessarily need `CompTree` to do such an operation. If you have JDK 1.1/1.2, you can press `Ctrl+Shift+F1` to dump the component tree of a `Frame` to the console. However, you still need `CompTree` to show the `newEventsOnly` flags.

When Wouldn't You Use `CompTree` in Your Apps?

The `CompTree` class can't be downloaded with applets (it must be placed in `CLASSPATH`), but that isn't a major problem. When I wrote the `CompTree` class, I saw it as a tool to use at design time. You could include it in your stand-alone apps to be used at runtime, of course, but it's not a good idea because the `CompTree` class is already included in the `java.awt` package. What would happen if you deployed more than one version of this class in different `.jar` archives and another person wanted to use them together? Another architectural change to AWT (beyond JDK 1.2) might cause the deletion of the `newEventsOnly` flag. Until something is modified, use `CompTree` as a tool to tune your apps, and don't forget the `-Xverify:none` option if you use JDK 1.2.

Request for Enchantment

Let's take a short look inside AWT and Swing. The constructor of the `java.awt.Component` class contains no instructions.

(*JDJ*, Vol. 3, Issue 8), I presented an app, `SmartLogin`, whose window was serialized into a file. I'm not interested right now in persistence. What I really want to find out is what event model was used by the components of the `SmartLogin` window (two `Panels`, two `Buttons`, three `Checkboxes`, two `Labels` and two `TextFields`). Therefore, I wrote a small app that creates an instance of the `SmartLogin` class (which inherits from `java.awt.Frame`) and calls the `printTree()` method of the `CompTree` class. The `main()` method has only three lines.

```
SmartLogin sl = new SmartLogin("");
CompTree.printTree(sl);
System.exit(0);
```

To use `CompTree` in this app, I had to include an `import` directive in the source file.

```
import java.awt.CompTree;
```

Listing 2 shows the results after the app is run. Most of the components (nine of 12) use the AWT 1.0 event model. Note that `SmartLogin` application doesn't use depre-

JTest

by Parasoft

<http://www.parasoft.com/jtest>

```
protected Component() {
}
```

The constructor of the Swing's `JComponent` class calls the `enableEvents()` method, which is inherited from `java.awt.Component` via `java.awt.Container` (`JComponent` extends `Container`). The `enableEvents()` method will set the `newEventsOnly` flag to `true`. The conclusion is that all of the Swing components use the delegation event model since they're created. They're also lightweight components; they don't have native peer classes.

```
public JComponent() {
    super();
    enableEvents(AWTEvent.FOCUS_EVENT_MASK);
}
```

Let's go back to AWT. Wouldn't it be nice to have a static flag – `defaultNewEventsOnly` – that could be used by the `Component()` constructor?

```
// This class variable doesn't exist
static boolean defaultNewEventsOnly = false;
```

```
// This constructor doesn't exist
protected Component() {
```

```
newEventsOnly = defaultNewEventsOnly;
}
```

If you were sure you used only AWT 1.1 in an application, then you could call, right from the start, a method like this:

```
// This method doesn't exist
public static newAWTOnly() {
    defaultNewEventsOnly = true;
}
```

This way you wouldn't need to call `addComponentListener(null)` to force the components to use the AWT 1.1 event model.

Summary

For compatibility reasons the AWT 1.0 event model is the default event model of the AWT 1.1 components. You usually have to register listeners to components to convince them to use the delegation event model.

This article teaches you two things: how to force the components that don't need listeners to use the AWT 1.1 event model, and how to discover the components that use the AWT 1.0 event model. You also learned a hacking trick: how to access a friendly vari-

able of a `java.xxx.Yyy` class without altering the JVM. 📧

References

1. Sun Microsystems, "The AWT Home Page," <http://java.sun.com/products/jdk/awt/index.html>.
2. Sun Microsystems, "Delegation Event Model," <http://java.sun.com/products/jdk/1.1/docs/guide/awt/designspec/events.html>.
3. Sun Microsystems, "Deprecated Methods in the 1.1 AWT," <http://java.sun.com/products/jdk/1.1/docs/guide/awt/Deprecated-Methods.html>.
4. Andrei Cioroianu, "Persistent User Interface for Multiuser Applications" <http://www.javadevelopersjournal.com/>.
5. Andrei Cioroianu, "Inside AWT," <http://www.geocities.com/SiliconValley/Horizon/6481/InsideAWT.html>.

About the Author

Andrei Cioroianu is an independent Java developer. He has a BS in mathematics-computer science and an MS in artificial intelligence. His focus is on 3D graphics (Java 3D), software components (JavaBeans) and user interface (AWT, JFC). You can reach Andrei for questions or comments at andcio@hotmail.com.



andcio@hotmail.com

◆ LISTING 1: The `CompTree` class.

```
// CompTree.java

package java.awt;

public class CompTree {

    public static void printTree(Container c) {
        // Prints the component tree, whose top is c
        printInfo(c, 0);
        printTree(c, 4);
    }

    private static void printTree(Container c, int l) {
        // Gets the component list of the c container
        Component a[] = c.getComponents();
        if (a == null) return;
        // Prints the information about each component
        for (int i = 0; i < a.length; i++) {
            printInfo(a[i], l);
            if (a[i] instanceof Container)
                printTree((Container) a[i], l+4);
        }
    }

    private static void printInfo(Component c, int l) {
        // This string computing is not optimized
        String s = new String();
        for (int i = 0; i < l; i++)
            s += ' ';
        // Prints the information about the c component
        System.out.print(s);
        System.out.print(c.getClass().getName());
        System.out.print(" -- ");
        System.out.print(c.getName());
    }
}
```

```
if (c instanceof Container) {
    LayoutManager m = ((Container) c).getLayout();
    System.out.print(" (");
    if (m != null)
        System.out.print(m.getClass().getName());
    System.out.print(")");
}
System.out.print(" -- ");
System.out.print(c.newEventsOnly);
System.out.println();
// Uncomment next line for more information
// System.out.println(s + " + " + c);
}
```

◆ LISTING 2: The component hierarchy of `SmartLogin`.

```
SmartLogin -- frame0 (java.awt.GridLayout) -- true
java.awt.Label -- label0 -- false
java.awt.TextField -- textField0 -- false
java.awt.Panel -- panel0 (java.awt.GridLayout) -- false
    java.awt.Checkbox -- checkbox0 -- false
    java.awt.Checkbox -- checkbox1 -- false
    java.awt.Checkbox -- checkbox2 -- false
java.awt.Label -- label1 -- false
java.awt.TextField -- textField1 -- false
java.awt.Panel -- panel1 (java.awt.FlowLayout) -- false
    java.awt.Button -- button0 -- true
    java.awt.Button -- button1 -- true
```

▶▶▶▶▶ CODE LISTING ▶▶▶▶▶

The complete code listing for this article can be located at www.JavaDevelopersJournal.com

JHL Computer Consultants

<http://www.jhlcomp.com>



JComponentTree

How to render a tree structure in a scrollable window with a visual component

by Claude Duguay

This month we put together a visual component, designed to render a tree structure in a scrollable window. `JComponentTree` can display a set of arbitrary components in a tree hierarchy. It can orient the tree in any of the cardinal directions (north, south, east or west); display the connections between nodes using straight or angled lines; and align each of the tree nodes to be left-, center- or right-justified.

The `JComponentTree` stores its data in a `DefaultTreeModel` compatible with `JTree` and uses `TreeNode` objects that extend the `DefaultMutableTreeNode` class.

Figure 1 shows an example `JComponentTree` hierarchy, fairly typical of the kind of structures you might want to display using the `JComponentTree` widget.

Using the JFC TreeModel

The `JComponentTree` widget is designed to take advantage of concepts developed by Sun Microsystems in the JFC `JTree` component. We work with the `DefaultTreeModel`, for example, so that migrating between views is possible. An application could implement a structure based on the `DefaultTreeModel` and easily switch between display components, permitting the `JTree` or the `JComponentTree` – or both – components to display the model.

The `TreeModel` provides methods that manage (`TreeModelListener`) listeners, get the root node and permit access to, or statistics about, the nodes at each level in the hierarchy. However, the `TreeModel` interface doesn't provide methods that actually change values in the model, so we use the `DefaultTreeModel` as the basis for our own `JComponentTree` model, accessing available nodes through the `TreeModel` interface.

My first attempt to keep the compatibility between `JTree` and `JComponentTree` optimized led to a few deep forays into the `JTree` source

code. Starting with a pure implementation of the `TreeNode` interface wasn't enough since it doesn't allow you to change any values. For that you need the `MutableTreeNode`, which provides a method for setting a user object – in our case a `Component` to be displayed. It doesn't provide a way to get the object back, however. This is a bit of a mystery to me, and clearly a shortcoming of the interface design. Our attempt to make the tree model as generic and compatible to `JTree` as possible leads us finally to the `DefaultMutableTreeNode` class.

Since we have to use the `DefaultMutableTreeNode` as the basis for our own tree structure, we'll add another layer, the `ComponentTreeNode`, to provide type safety, thus ensuring that the user object is always a component. By extending `DefaultMutableTreeNode`, we can also make sure that a `JTree` control could display the same basic structure with minimal effort.

Listing 1 shows the source code for the `ComponentTreeNode` class. We extend `DefaultMutableTreeNode` and pass the component to be stored as the user object

in the constructor, saving it by calling the superclass constructor. The `getComponent` method casts the user object into a `Component` when we ask for it.

The ComponentTreeLayout Manager

To keep the coupling to a minimum, we use a layout manager. Unfortunately, this implementation is slightly more coupled than you might normally expect since we need to draw the lines connecting nodes by explicitly using the `paintComponent` method in the parent container.

The `ComponentTreeLayout` class performs a lot of work. Here's a quick list of its major responsibilities.

- Set/get values for alignment, linetype, tree direction, etc.
- Calculate `minimumLayoutSize` and `preferredLayoutSize`.
- Calculate component positions and lay out the tree.
- Draw the lines that connect each of the components.

The more notable member variables are alignment, linetype and direction. The alignment value determines whether child nodes are aligned to the LEFT, CENTER or RIGHT of the parent. The linetype value determines whether lines are drawn directly between nodes in a STRAIGHT line or as SQUARE lines, forming right angles to the nodes. The direction value determines whether the tree is drawn with its leaves to the NORTH, SOUTH, EAST or WEST.

Each of these values has an associated set and get method that follows the JavaBean standard (`setValue/getValue`, where Value is the actual name of the variable), and constant values are declared in the `ComponentTreeConstants` interface, which you can see in Listing 2. In addition to these, we have access to the `TreeModel` model value, which is exposed through the `getModel` and `setModel` methods, and the root node, through the `setRoot` and `getRoot` methods. Several constructor variants are available. Each requires a `TreeModelListener` so that we can



Figure 1: JFC

DATA

Representations

<http://www.datarepresentations.com>

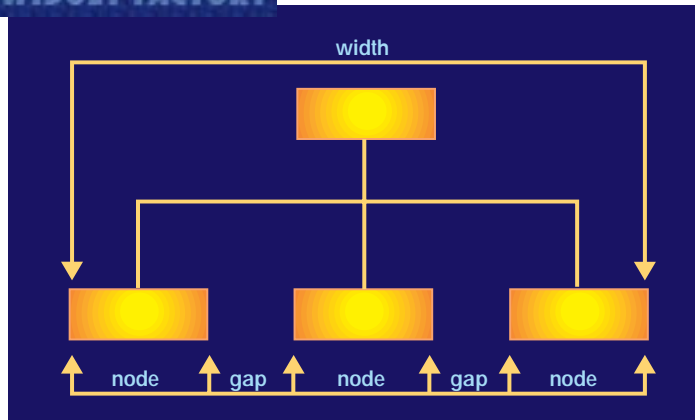


Figure 2: Recursive width calculation

notify the parent container of any changes to the model.

Size Calculations

The `ComponentTreeLayout` subclasses the `AbstractLayout` class, which you might remember from an earlier article I wrote, “Practical Layout Managers” (*JDJ*, Vol. 3, Issue 8). It provides default behavior for most layout manager calls and a good foundation to start from when you’re working with them. As with any layout manager, the `ComponentTreeLayout` must calculate the minimum and preferred size for the container in which it’s used.

Listing 3 shows code for the preferredLayoutSize method. We take the inset values into account and consider the horizontal gap between each node. Figure 2 shows how we can determine the width of a given node and its immediate children. We simply add the child node widths and the `hgap` values together, and test to make sure the parent node is not wider. If it is, we always take the wider value.

The main preferredLayoutSize method calls `getPreferredSize` with the root node, retrieved directly from the model, and `getPreferredSize` calls itself for each of the child nodes it finds along the way. This accumulates the height and width until we’ve traversed the entire tree and then returns the correct value at each level. The `minimumLayoutSize` calculation is almost identical, though we ask each node for its `minimumSize` instead.

Positioning Components

When the container calls `doLayout`, the layout manager calls the `layoutContainer` method. This method determines which orientation we’re dealing with and provides a starting `x` and `y` value along with the root node to the layout method, which recursively repositions and resizes each component using the `setBounds` method.

The layout method needs to determine the size of each node and its immediate

children, but it can’t call the `getPreferredSize` method without running into proportion problems. Instead, we have a near duplicate of `getPreferredSize` called `getLayoutSize`. Listing 4 shows the `layoutContainer` and layout methods, but leaves out `getLayoutSize` because it’s almost identical to `getPreferredSize` in Listing 3.

The layout method takes into account the orientation and node alignment before deciding where each node should be placed. We walk through the child nodes, calculate the `x` and `y` positions and recursively call the layout method to traverse the tree structure.

To paint the lines between components, we have to call the `drawLines` method explicitly. Listing 5 shows the `drawLines` method, which recursively draws lines by calling `getBounds` on each component to determine where they’re positioned. Painting always happens after the layout call, so this is perfectly safe.

The JComponentTree Component

The `JComponentTree` code, shown in Listing 6, provides an interface to a number of `ComponentTreeLayout` methods, allowing the model, orientation, linetype and alignment to be set and retrieved. Whenever one of these attributes is reset, the `doLayout` method is called, along with `repaint`, to refresh the `JComponentTree` view. Listing 6 shows only one of the available constructor variations, and skips some of the accessor methods and most of the methods required by the `TreeModelListener` interface. Each of the constructors calls the `ComponentTreeLayout` equivalent. The one in Listing 6 is the most extensive.

The `JComponentTree` control also implements the `TreeModelListener` interface to monitor changes in the tree structure. If one of these events is fired, the layout is recalculated and redrawn. When field values are changed, we also fire the layout method and `repaint` the tree. The exception is `setDirection`, which also calls `setSize` to make sure



Figure 3: An application component hierarchy

the scrollable panel size is correct.

The `addNode` method creates a `ComponentTreeNode` object and adds the component to the container. To make it possible to create children that refer to an added node, we return the `ComponentTreeNode` object and handle null parents as a request to set the root node. You’ll want to pay attention to this since accidental null parent nodes won’t throw an exception and you’ll end up with unexpected results.

When you download the code from our Web site, you’ll find a test class called `JComponentTreeTest`. This class generates a random tree with variations in depth, maximum width and randomly selected components. In addition, it provides a set of buttons that lets you dynamically change the direction, linetype and node alignment so you can get a feel for what can be done.

Summary

You now have yet another control to add to your programming toolbox. `JComponentTree` offers an occasional alternative to `JTree` and provides some overlapping functionality, but it’s typically used in situations that require displaying a tree structure in different ways. This widget lets you organize arbitrary components rather than using a renderer to display data, so it has a different overall purpose. Still, there’s just enough commonality with `JTree` to allow for easy migration. ☺

▼▼▼▼▼ CODE LISTING ▼▼▼▼▼

The complete code listing for this article can be located at www.JavaDevelopersJournal.com

About the Author

Claude Duguay has been programming since 1980. In 1988 he founded LogiCraft Corporation, and he currently leads the development team at Atrivia Corp. You can contact him at claudio@atrieva.com.



claudio@atrieva.com

DISTINCT SOFTWARE

<http://www.distinct.com>

ADVERTISER INDEX

Advertiser		Page	Advertiser	Page	Advertiser	Page		
Borland www.borland.com	408 431-1000	19	KL Group Inc. www.klg.com	800 663-4723	B/C	Silverstream www.silverstream.com	888 823-9700	83
Bristol Technology www.bristol.com	203 438-6969	75	Live Software info@livesoftware.com	619 643-1919	41	Sockem Software www.sockem.com	814 696-3715	65
Coriolis www.coriolis.com	800 410-0192	77	Net Dynamics www.netdynamics.com	650 462-7600	79	Stingray Software Inc. www.stingsoft.com	800 924-4223	2
Greenbrier & Russel www.gr.com/java	800 453-0347	25	ObjectShare www.objectshare.com	800 973-4777	43	SunTest www.suntest.com	415 336-2005	11
Halcyon www.halcyonsoft.com	888 333-8820	35	Object Matter www.objectmatter.com	305 718-9101	50	Sybex Books www.sybex.com	510 523-8233	63
IBM www.ibm.com	800 426-5900	58&59	ObjectSpace www.objectspace.com	972 726-4100	4	The Object People www.objectpeople.com	919 852-2200	23
IEC-EXPO www.iec-expo.com	888 222-8734	73	Object Management Group www.omg.org	508 820-4300	53	SYS-CON Publications www.sys-con.com	800 513-7111	71
ILOG www.ilog.com	415 688-0200	17	Progress/Cohn & Godly www.apptivity.com	800 477-6473	21	Thought, Inc. www.thought.com	415 836-9199	48
Installshield www.installshield.com	800 374-4353	13	ProtoView www.protoview.com	609 655-5000	3	Visionary Solutions, Inc. www.visolu.com	215 342-7185	50
Inno Val www.innoval.com	914 835-3838	38	Roguewave www.roguewave.com	800-487-3217	15	WebMethod www.wbmethods.com	888 831-0808	33
Keo Group www.keo.com	978 463-5900	22&37	Sales Vision www.salesvision.com	704 567-9111	47	Zero G. Software www.zerog.com	415 512-7771	6

Beautiful Things Come in Small Bundles

Against greed: An argument for more startup companies in the computer industry and more alliances for effective marketing

by Alan Williamson

'Morning...or is it afternoon? It could even be evening. Whatever it is, welcome. Another month has rolled in and we're now sailing dangerously close to that Christmas mark again. Goodness, where has the year gone? I've no idea. This is my wee corner of the journalistic minefield of the computing industry. Here I don't teach about Java. I don't take you through the joys of class design or the hell of class threading. Instead I ask you to sit back, push away the keyboard, tether the mouse and prepare yourself for some gentle mental stimulation as I give you some food for thought about the overall picture, the greater goal.

This month let's plump for greed, one of the seven deadly sins. When talking about money, it's really the only one that can apply. I want to take a look at the financial issues facing a small company in the Java world.

As most of you know, I run N-ARY Ltd., a British-based Java consulting company I gave birth to more than three years ago with nothing. I donated my old 486 and that was the only thing of note on the balance sheet. Three years later we have a team of 12-plus people and projects all over the world. And what do we owe our investors? Nothing. We don't have investors.

This is the first point I'd like to bounce around. When starting out on this long road to success, there seems to be the British way of doing things and the American way of doing things. It appears that when a group of skilled individuals get together to start something, what happens next depends largely on the side of the Atlantic you wake up on.

In America the first thing on the agenda, or at least one of the top things, is to find an investor. Whether in the guise of a friend, bank or, more than likely, venture capitalist doesn't matter. It seems my American cousins prefer working with a large sum and working down, hoping they start making money before it reaches zero. Us Brits on the other hand start with zero and work up, hoping we don't expand beyond the month's cash flow. From day one the American company is worth, say, \$1 million, whereas the British company is worth practically nothing. After three years the tale can be somewhat different.

Granted this is a gross generalization, but it serves to illustrate my point. The question I put to you is, Which way is the right way? Talking to you who are CEOs, if someone approached you with advice on starting their own company, would you recommend the route you took? Perhaps another route? And if a different approach, why not the one you started off on? Answers on a postcard, please!

As one of the CEOs of a Union Jack company as opposed to a Stars and Stripes company, I'd have to say that our way may have worked 20 years ago or for companies that have no ambitions for growing multimillions. But in this day and age hard work alone is not enough. To successfully compete in today's high-paced industry I think we have to look at the American way of doing things. If companies are to succeed, there's not enough time in the day, alas, to grow the company organically. Ironically, the root of the problem may be the multimillion dollar companies, the very thing we CEOs strive to take our companies to. But why?

Many of the large companies have become so large that they can afford to lose money on projects in order to stifle competition. If a small company pops up with an innovative and exciting product and begins marketing it, chances are the marketing strategy employed will be very targeted and low-cost. Now if one of the bigger companies happens to see the idea, one of three things will happen: the first is for them to ignore it. The second is for the large company to buy out the smaller company. This is probably the best-case scenario, and I'll come back to it later. The third - and most damaging - response is for the large company to produce their own version of the product and market off the back of one of their more successful products.

For the smaller company this generally means a death sentence. Remember back in the old days of the World Wide Web, where one of the benefits that was touted around was the ability to give everyone a level playing field? Give the smaller company the ability to compete with the corporates. All you had to do was produce a Web site and you

could start the duel. However, anyone who's tried this knows that it's not so much a duel as a complete mismatch. Sure, the Web site may be far superior, but what's the use of a Web site that no one visits? This is where the corporate entity can win...in its ability to attract users.

The corporation has the money to market the Web site elsewhere, the money to advertise the site both online and in more traditional media such as magazines and on billboards. The smaller company does not. Chances are the smaller company may have a far superior product, but what's the use of that if no one knows about it? It's a catch-22.

On my travels I heard of a small company based in Brisbane, Australia. Now these guys are typical of what I've just described. They have an exciting product that's doing famously well in the local market but they're scared to take it to America for fear someone might steal it. Granted, their very presence on the Web opens up their treasure chest for possible perusal by some Californian. The assumption is that they're unknown in the global arena and want to stay that way...for the time being at least. So this column is really going to help them, eh? Oops....

The point is that in any other industry they have a strong chance of survival. They could grow the company slowly and surely. In our industry, however, the plot isn't as straightforward. All it takes is for Microsoft or Sun or another big company to take the idea and bundle it free with their next release. What chance does the small company have at this point? They now have to justify to their potential clients why they have to pay for the product when it's free elsewhere. A tough sale to make.

Before the Internet, an idea had the time to evolve and grow. Only a handful of people would know about it, and this would be through word of mouth. A good idea comes along now, and before you know it 16 other companies around the world are suddenly engulfed in the same project. A scary prospect at the best of times.

I think the best example of that has to be Web-based e-mail. When Microsoft bought out Hotmail earlier this year, did anyone notice the sudden flood of Web-based e-mail sites coming online? Even Yahoo! offers some sort of Web-based e-mail system and these guys are supposedly a search engine. There are a



number of smaller Web-based systems around that are far more feature-rich than the “big boys” but have no chance of survival on a commercial level against the wrath of freebies.

I’m sure you can think of many examples of where this has happened. Another, more famous one is the handheld computer from the GO Corporation. Their CEO, Jerry Kaplan, tells a very entertaining – and in parts scary – story in the book *Startup*. I’d recommend it for any budding CEO, ignoring the last part; it’s a very inspirational book. In this situation, however, Kaplan had venture funds but they proved ineffective against the might of Apple and Microsoft. The moral? Just because your project may be heavily funded doesn’t mean it’s guaranteed to succeed.

A company structured from a venture fund generally has an exit plan, which is usually to be bought out in three to four years. That’s the American dream, to be bought out and walk away with a million or more dollars in the back pocket. But let’s take a quick look at the overall picture. Someone once commented to me that they had read in a journal that in around 10 years there’d be only a handful of computer companies. All the smaller companies would have been absorbed into the bigger corporates.

Now on the face of it the comment seems rather silly. Think of the thousands of companies all around the world, all developing vari-

ous tools and services. So let’s see if this could be possible. We already know that the exit plan for many startups is to be bought out by a bigger company. Admirable. If this continues, however, with bigger companies buying up the smaller ones, the above statement suddenly starts to ring some truth bells. Maybe there’s something in it after all. But there has to be a cutoff at some point, and maybe this is where the statement falls apart. Looking at the current trend now, it’s possible to see a future with tougher competition and fewer companies.

So should we forget about running our own companies? I don’t think so. Conversely, I think that in order to crack the monopoly the larger corporates hold, more small organic companies should be started, and alliances should be formed to effectively compete in the marketplace. The beautiful thing about the Internet is that it brings together this rich tapestry of talent and diversity and allows it to be explored and realized. We need more variety, not a handful of companies controlling everything. Do we want the computing industry to go down the same route as the publishing industry? Where two or three individuals control and own the whole industry worldwide? Names like Ted Turner and Robert Mudoch are familiar to everyone. We’re heading down there already, but it’s not too late.

Java has been responsible for a new influx of startup companies. We are one of these companies. Java has given us the ability to compete with companies irrespective of platform. In many respects we’ve been given the perfect tool to compete effectively with the big companies. We don’t need to make that agonizing decision about which platform to support. We can let the big companies play their own politics; by our not directly supporting them, we’re slowly weakening them. I love meeting other CEOs and learning about their growth and their excitement in using Java. It’s good to see this wonderful array of companies explode into the marketplace.

It all comes back to greed, the greed of the bigger corporates who want to own everything and control every corner. We’re in an exciting time, and if everything does go pear-shaped at the end of the day, well, at least we’ll have good stories to tell our grandchildren of how we tried to take on the world...and, in a small way, won. ☺

About the Author

Alan Williamson is CEO of N-ARY Limited, a UK-based Java software company specializing solely in JDBC and servlets. He can be reached by e-mail at alan@n-ary.com or online at <http://www.n-ary.com>.



alan@n-ary.com

INSTALLSHIELD Java Edition

<http://www.installshield.com/java>



Emblaze Audio/Video

by GEO Publishing, Inc.

You'll be amazed by how easy it is to add multimedia to your Web site

by David Jung



The Internet has come a long way from Gopher and WAIS sites for distributing information. The World Wide Web has opened up a whole new avenue of products and a more usable method of deploying information. Multimedia has become a popular way to display your ideas over the Internet. There are a number of products on the market to help you deploy multimedia files through the Internet, but most require plugins and if you can't get them configured properly you might be left on the side of the road on the Information Superhighway. Other Internet multimedia components require your clients to install a browser plug-in or special server-side software on your Web server.

GEO Publishing has a pair of products that raise the ante on streaming multimedia over the Internet. These products, Emblaze Audio and Emblaze Video, offer small Java applets to allow their respective media to stream data over the Internet without the need of a plug-in or server-side software.

Gently Down the Stream-ing

In case you've heard these terms before but never quite knew what they meant, here's the abridged version of their definitions. The term *streaming* refers to a transmission of information in one direction, in which both the client and the server cooperate for uninterrupted data. The client side will buffer a few seconds of data before it starts sending it to the screen and/or speakers. This compensates for any momentary delays in packet delivery. Therefore, *streaming audio* refers to the uninterrupted one-way transmission of sound bytes to the client, whereas *streaming video* refers to the uninterrupted one-way transmission of

video bytes to the client.

Hear and See the Difference

Emblaze Audio allows you to put streaming audio on your Web site. It handles all of the popular audio sound files - AIFF and SND for Mac users, and WAV for PC users. The Emblaze Audio compression program, as shown in Figure 1, takes your audio file and compresses it for optimal transfer. The compressed file becomes an .EA file and the program generates the necessary HTML document to run the audio file. All you have to do is put the custom Web look and feel you want, and publish it and the Emblaze Audio applet to your Web site. When a person goes to your Web site, as soon as the Java applet loads it starts

Emblaze Audio/Video

GEO Publishing, Inc.
21110 Oxnard Street
Woodland Hills, CA 91367
Phone: 818 703-8436
Fax: 818 703-8654
Web: www.emblaze.com
E-mail: sales@geopub.com
Price: Emblaze Audio \$99 (PC or Mac)
Emblaze VideoPro \$295 (PC only)

playing the audio track.

The compression ratio for Emblaze Audio is quite impressive. It was able to compress a 2.7 MB file down to a 201 Kb file. Listening to the file over the Internet with a 28.8 connection, I found the quality to be excellent. The only thing that your clients might not like is that there's no way to stop the audio once it starts unless they move to a different HTML document.



Figure 1: Audio compression program

JDesignerPro by BulletProof

<http://www.bulletproof.com>

Emblaze Video allows you to add streaming video to your Web site. VideoPro isn't a video editor; it's designed to prepare Microsoft Video (.AVI) files for the Web. Just as with Emblaze Audio, you process your files through a compression application, illustrated in Figure 2. GEO really took a look at their competition for the type of streaming format to consider for this product. There are 12 predefined settings to choose from, ranging from "28.8 with smooth playback, small window with audio" to "T1 smoothest playback, original size image with audio." If you don't like any of these settings, you can select the Advanced option to mix and match your own compression preferences. Unlike the Emblaze Audio interface, you can bring multiple files into a list and compress them one by one, or find the settings you like and compress them all at once. This results in an .EV2 file, and an HTML page is generated with the necessary Java applet arguments. Just customize the HTML page, and publish it and the Emblaze Video applet to the Web. Just like Emblaze Audio, once the Java applet is loaded into memo-

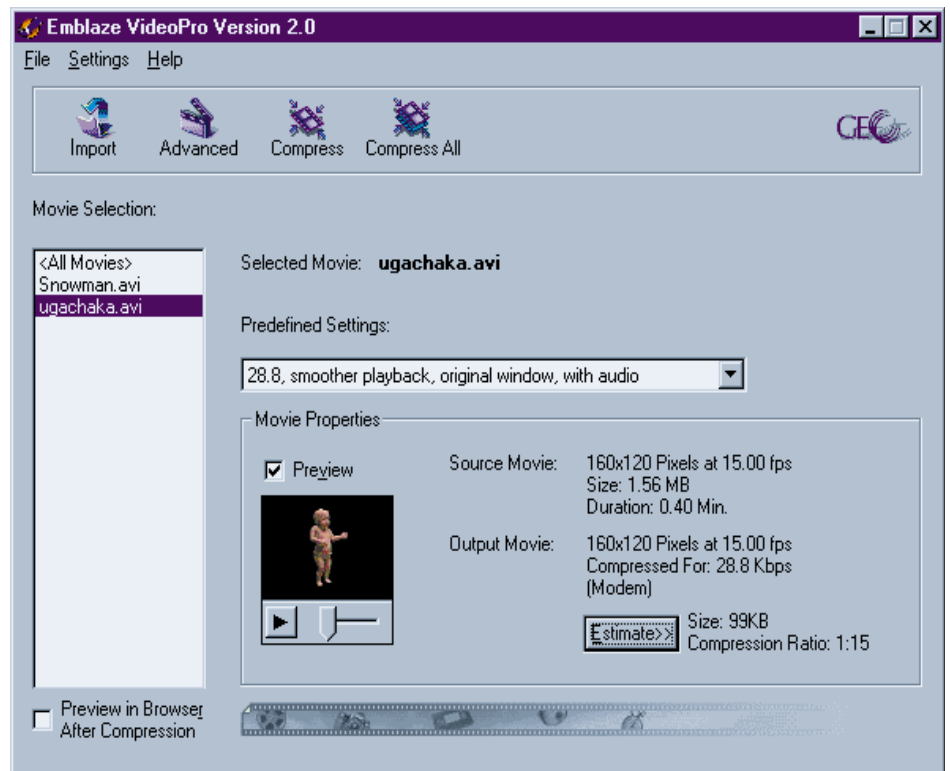


Figure 2: Video compression program

SYS-CON RADIO

Tune in for
detailed discussion of
products from JDJ advertisers!
Java readers voted #1 with their browsers!

2 million banners delivered each month
(more than all other Java media added together!)

ry, not only will you be able to hear the difference you'll also be able to see the difference.

This tool is more difficult to get used to than Emblaze Audio, but that's expected because it has to deal with a lot more settings. The compression ratio for VideoPro is just as impressive as Emblaze Audio. A 17 MB AVI file (320x240 pixels at 15 frames per second running 4.5 minutes) compressed down to 955 KB keeping the same pixel size and frames per second, optimized for a 28.8 modem connection.

Batteries Not Included

Not to discourage you from trying this out yourself, but you should be aware of what you'll need to make it work. Assuming you want to put your multimedia on your site, you'll need software to create your media files. Both Windows and the Macintosh versions come with software that allows you to record audio files, but they're limited as far as the quality and recording time. There's a large amount of shareware and commercial audio recording software out there, so you shouldn't have any problems creating your sound bytes.

Video creation is another story. You can scan the Net for your favorite AVI files, but due to copyright reasons you aren't allowed to put them on your Web site without written permission from the owner. In order to create your own video clips, you'll need to have a video capturing system, like IOMega's Buz or Diamond Multimedia's Supra Video Kit as

well as a video input source like a camcorder or videotape player.

Conclusion

With both of these products, you'll be amazed by how easy it is to add multimedia to your Web site. You don't need a super-fast computer with MMX to take advantage of either applet. You don't have to rely on any plug-ins, which will make your target audience happy. And not having to purchase any expensive server-side software will make you, your Webmaster and your wallet very happy. For organizations that have firewall issues, the limited testing that I performed behind one gave me no problems getting to the streaming media, unlike some other streaming media plugins I've used. The manuals for both programs are spartan, but cover everything necessary. There isn't any online help, not that you're going to need it.

These are great products that work well with both Netscape and IE (you might need the service pack for IE 4.9 though). If you've ever wanted to put streaming audio and video on your Web site, these products are definitely worth looking into. ☺

About the Author

David Jung is a senior programmer analyst for a national medical center in Southern California. He is a key architect for all client/server development for the organization. He can be reached at davidj@vb2java.com.



davidj@vb2java.com

Emule SDK by SlangSoft

<http://www.slangsoft.com>



Vision JADE 4.0 Business Logic Server Studio

by Vision Software Tools, Inc.

A business rules development platform for creating and deploying components across multitier environments

by Jim Milbery



Vision JADE 4.0 is the newest release of Vision Software's application development environment for Java. While there are a bevy of Java development environments on the market, Vision JADE offers a different approach to developing Java applications. The basic design of JADE builds upon a model-driven approach to application development that Vision Software pioneered long before the world had heard of the Java language. Vision Software is in the process of releasing an updated version of JADE, version 4.0, which features a number of enhancements including an application server component.

Product Installation

Vision JADE 4.0 is scheduled to be released into production by the early fall, and I was given a version of the beta code, complete with features to work with. Despite the fact that this was an early release, the installation CD-ROM came complete with a standard Windows install program that simplifies the process. There are four components to install as figure 1 shows.

You'll want to be sure and read the options before you proceed with installation so you'll be prepared for any last minute changes. In my case I needed to select the "install third party products" choice in order to install the required versions of Intersolv's ODBC drivers. Although you can install the complete system on a stand-alone Windows NT machine, I chose to install the development environment on one machine and the Business Logic Server alongside my Oracle database on a second machine. In most cases you'll only need to install the Vision Development Platform and Vision Business Application Server (the proxy service is provided for

sites that will be accessing the BAS from behind a firewall). The overall installation went smoothly, despite the fact that it was a pre-release of the software, and the entire installation process took only an hour of my time and about 100 megabytes of disk space.

Vision JADE Product Components

The Vision JADE suite of products is not just another Java development environment. Rather, it's designed more for the business application developer than the pure Java technologist. The JADE product family is composed of three key components: the Vision JADE Business Studio, the Business Logic Server and the JADE extensible data access framework (XDA).

The JADE Business Studio is the hub of the development environment and serves as the starting point for building applications. At the heart of the development environment is the JADE repository, which stores your data models and all application components. While the use of a central repository is a powerful concept, the JADE repository can be stored only in a Microsoft Access database in this release of JADE. Vision has a strategy for supporting the forthcoming repository standards being discussed at the Microsoft and Oracle camps. For the moment, JADE developers will have to make use of a PC-based file system in which to store the repository (since MS-Access runs only on PC file systems).

After installing JADE and starting the Business Logic Server, I launched the JADE Business Studio to begin developing my first JADE application. The starting point for any JADE application is a business data/object model. JADE does provide its own tool for designing a data/object model, but it can reverse-engineer an existing data/object model from any of the databases that it supports. Vision JADE has the ability to create tables on its own; I would recommend using a more robust data-

Vision Software Tools

2101 Webster Street, 8th floor
Oakland, CA 94612
Phone: 800 984-7638
Fax: 510 238-4101
Web: www.vision-soft.com
E-mail: sales@geopub.com
Price: \$3000/Developer Studio, Business Automation Server \$6000/CPU NT,
\$20,000/CPU UNIX

modeling tool such as ERWin, PowerDesigner or Oracle Designer to create your database if it doesn't already exist. I used the reverse-engineering tool to load a data model from an existing Oracle8 database into JADE. It was a simple four-table database that models a student records application. JADE was not only able to quickly import this data model into the JADE repository, but it was also able to detect the primary keys and foreign key relationships between each of the tables. The next step in the process would be to model the various business rules in the JADE Business Rules Editor. However, I chose to jump ahead and generate a quick application based on the default tables I had pulled from my Oracle8 database. Within minutes I was created a simple application that would allow users to interact with the base tables of the application.

The left-hand project browser can be used to navigate through the various data entry forms that I created in the application. The right-hand panel offers a visual view of the forms hierarchy and the basic transition between form elements. JADE automates many of the routine tasks that would be associated with developing a data-centric Java application. I was able to use the business rules designer to modify the CLASSES table of my application to serve as a drop-down lookup list for the UGRADS table. Once I began to generate data forms based on the data model, JADE automatically made use of the lookup list for all forms that referenced the CLASS table. Overall, the development environment is well organized and easy to use, but you can't "dock" many of the tool

JSales by SlangSoft

<http://www.slangsoft.com>

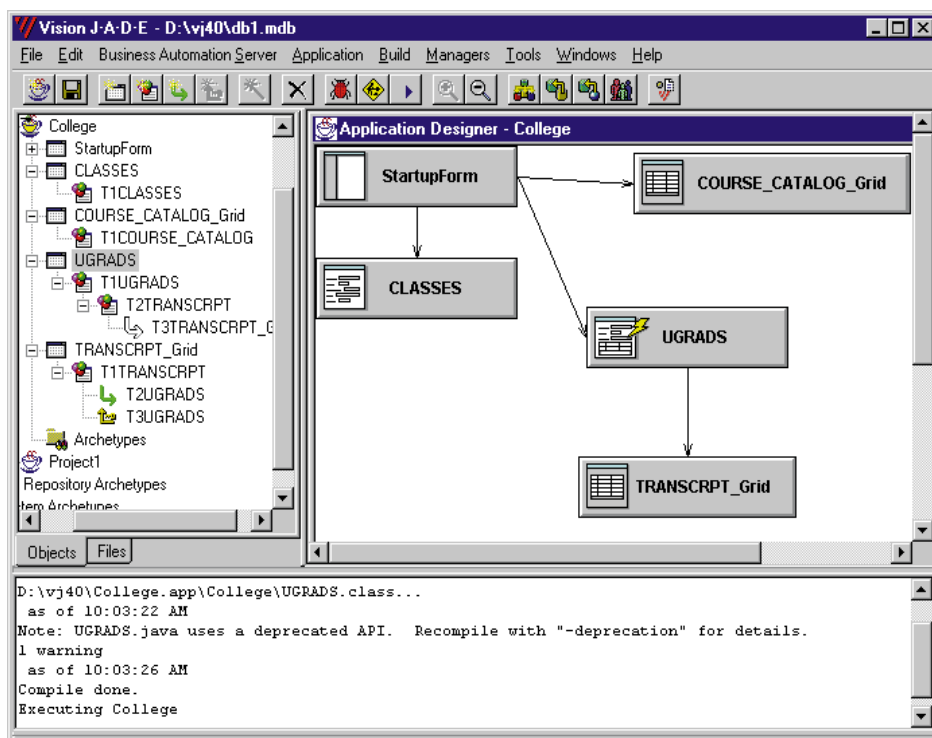


Figure 1

bars to the main IDE window, which can make the desktop a bit cluttered.

Maximizing Your Productivity

The real power of JADE can be found in two key areas: the business rules designer and the archetype builder. The business rules designer is used to add rules to the various data objects you create. You can use this facility to create data integrity rules, processing logic and action events for any of your base data objects. When you generate (or deploy) your application, JADE will generate the necessary database definition language (DDL) commands to implement your database and integrity rules. In addition, JADE generates the necessary Java code for implementing the remainder of your business rules and event code. JADE will automatically partition the code to run on the Business Logic Server. While this can save you considerable time, JADE does not allow you to fine-tune the partitioning process. Forms can be built directly on top of data model objects and relationships, or they can be built on top of canned SQL queries. Unfortunately, JADE does not provide its own query-builder, and requires developers to use the query-building facility built into MS-Access.

Archetypes

While JADE generates a large amount of the code for you, developers can affect this process directly through JADE's archetypes builder. For those of you who are unfamiliar with this term, the *American Heritage Dictionary* defines archetype as "original model or type after which other similar things are pat-

terned." All of the key application generation capabilities of JADE are based on a series of archetypes. JADE's archetype designer is used to move a system archetype to an application where it can be edited to reflect your company's standards for application design. Archetypes are a powerful facility for implementing standards throughout your organization, but they should be approached carefully. While application specialists will find JADE easy to use, the process of physically editing and testing archetypes is best left to the technical specialists.

The JADE installation comes complete with over 20 demonstration applications that can be used to give you a head start on building JADE applications. These samples cover the gamut from building servlet applications to adding JavaBeans to an application. There is even a sample XDA application that will help you connect JADE to a custom data source. While these demo applications are well designed, I would still recommend training before you embark on a major application with JADE.

Deploying the Application

JADE allows you to test your application directly from within the development environment, either as an application or as an applet within a browser. In addition, the development environment allows you to call out to other Java development tools (such as Microsoft J++ or Symantec Visual Café), as well as makes use of your favorite visual HTML editing tool. I was able to integrate Symantec's Visual Page with JADE, which allowed me to edit my HTML with a graphical

editor, a feature that JADE doesn't provide. This release of JADE offers a Java-based application server for the middle tier (the Business Logic Server). To make the process of deploying and managing these applications easier, JADE comes equipped with a Business Logic Console tool.

The Business Logic Console offers an impressive suite of capabilities, including the ability to manage database connections, users and security. While the development tool itself can deploy the application to the server for you, the Business Logic Console is used to configure the application once it has been deployed. The left-hand pane of the Business Logic Console works like an outline control, while the right-hand pane shifts as you select components with your mouse. I was able to restrict access to the data tables in my application to certain users directly through the Business Logic Console. In the longer term, Vision intends to support third-party application servers and provide the ability to deploy business objects as Enterprise Java Beans (EJBs). Vision has already implemented failover capabilities and load balancing, which will be enhanced in future releases of the BLS. Vision has certified the BLS to run under Windows NT initially, and will certify the server on a number of key UNIX platforms.

Test Environment

Client: Dell Pentium II 200 MHz, 64 MB RAM, 4 gigabyte disk drive, Windows NT 4.0 (Service Pack 3), ViewSonic 15-inch SVGA monitor, 3COM Etherlink XL Adapter and 8X CD-ROM.

Server: Dell Pentium II 266 MHz, 128 MB RAM, 8 GB disk drive, Windows NT 4.0 (Service Pack 3), Sony 15-inch Trinitron, 3COM Etherlink XL Adapter and 32X CD-ROM.

Summary

Vision JADE 4.0 offers a high-productivity environment for building Java-based business applications, and I recommend that you put this product on your shortlist of development tools to evaluate. JADE is attractive for organizations looking to leverage the power of their application designers in the Java world. You may wish to consider alternatives if you're looking for a low-level Java development environment. ☛

About the Author

Jim Milbery is an independent software consultant based in Easton, Pennsylvania. He has over 15 years of experience in application development and relational databases. Jim can be reached at jmilbery@milbery.com, or via his Web site at www.milbery.com.



jmilbery@milbery.com

ObjectPeople TopLink to store Java Objects

<http://www.objectpeople.com>

CASE STUDY

by Sriram Sankar

Unique Java issues, solved with specialized tools

"...building tools designed to solve specific issues related to software development, rather than to simply retrofit C and C++ technology..."

About the Author

Dr. Sriram Sankar holds a bachelor of technology degree in computer science from Indian Institute of Technology as well as MS and Ph.D. degrees in computer science from Stanford University. Currently the president and CEO of Metamata, which he founded in 1997, he can be reached for questions or comments at sriram.sankar@metamata.com.



sriram.sankar@metamata.com

Large Scale Software Development in Java: *Issues and Solutions*

As developers are increasingly using Java for advanced applications, they've become dependent on the availability of scalable technologies and tools to support their development, including quality assurance (QA), testing, maintenance, release and customer support requirements. The technologies available today have been inherited largely from those available for languages such as C and C++, including visual IDEs and a host of other tools that offer a solution to a particular problem. A few tools have been tailored specifically for Java and enhance the strengths of the language (like the incremental IDE VisualAge and InstallShield's installer for Java that allows Java applications to be installed onto any Java-compliant platform).

This article discusses specific issues related to large-scale software development in Java, suggests ways to address them and concludes with an overview of the Metamata (derived from meta-automata) toolsuite's answer to some of these problems.

While large-scale software development in Java faces some of the same issues as those of other languages, some are unique to Java. Factors related to C and C++, such as memory leaks, don't exist in Java because of garbage collection. On the other hand, Java introduces different issues such as thread analysis and memory debugging.

Standard IDEs don't address most of the problems that arise during large-scale software development. In fact, they aren't designed to be a complete solution for large-scale development. Hence their functionality must be augmented with specialized tools.

Organization and Maintenance of Software Components

This is one of the big tasks of large-scale software development. The system must be arranged into a set of small, manageable components that interact with each other. The interaction should take place through well-defined, organized interfaces, simplify-

ing the task of managing and maintaining the components.

Typically, time constraints and insufficient experience combine to introduce defects in the way software systems are architected, leading to decreased quality and larger overheads in managing and maintaining the system.

As an answer to this problem, a number of studies have measured software systems for complexity, which has led to a standardized set of software quality metrics. While there's no substitute for experienced project managers, the metrics do offer insight into assessing software complexity and quality.

It's also important to be able to detect inconsistencies in the program as it changes. Typically, a program may be changed in one place, but the effect of these changes in other places is overlooked. For example, by changing a type it's possible to make an existing type cast located in a different module no longer necessary, and also overlook this type cast.

Time Constraints

A problem faced by large software systems development (in any language) is waiting for the system to be rebuilt after every small change. The time required to rebuild after each change increases with the size of the system, adding up to expensive overhead costs. After a certain point, the necessary, endless rebuilds significantly reduce productivity.

Organizing a system into well-architected components and reusable libraries goes a long way toward solving this problem. Yet developer tools still need to be smart about how much rebuilding they have to do for each small change.

The best solution to this problem is incremental development environments such as VisualAge. They recompile only the minimum amount necessary when a system is changed. This concept of incrementality can also be extended to other activities beyond the standard development steps to include QA, testing and so forth.

Memory Management

Large systems tend to use a lot of memory, and unless it's managed carefully the capacity of the underlying hardware can quickly be exhausted. In systems written in C and C++, the developer has complete responsibility for making sure that unused memory is recycled for future use rather than retained indefinitely. Java addresses this problem with automatic garbage collection, i.e., the Java Virtual Machine periodically searches for memory that is no longer in use and recycles it for future use.

Unfortunately, garbage collection can take a significant amount of time when systems use a lot of memory, severely contributing to performance degradation. Most Java programmers today assume that they have to live with this in large Java programs.

The solution is to actively manage memory, and simply let the garbage collector kick in for the smaller chunks of memory as well as what slips through the cracks of the explicit memory management routines. Hopefully, better garbage collection algorithms will become available shortly in Java Virtual Machines and the problems related to garbage collection will soon be a memory; the next six months will reveal this possibility.

Another memory-related problem specific to Java is leaks due to the unrelinquishment of memory that's no longer necessary. In Java the garbage collector can only recycle memory that isn't being retained by the user program. Memory retained erroneously by the user program will never be collected, even if it's not used anymore.

To solve these problems, debuggers and profilers need to provide specialized features for Java. Debuggers should provide capabilities to determine whether a memory leak is occurring, and profilers should provide insight into the details of memory allocation.

Performance

Clearly, performance is the biggest issue for Java programmers. Performance of Java programs, in relation to C and C++ programs, necessarily suffers because of the following reasons:

- Java is an interpreted language, and by nature interpreted languages run slowly. A lot of work is being done to improve performance while remaining in an interpreted environment (e.g., Just In Time [JIT] compilers). However, performance will never catch up with compiled languages.
- Garbage collection contributes to performance degradation, especially in large programs.
- Java is a richer and more secure language

than C and C++. It offers features such as serialization and reflection that are inherently inefficient although they significantly enrich the language. Also, Java performs checks at runtime, such as bounds checks for every array reference that causes a degradation in performance.

Good profilers are important with Java. Information provided by profilers can help developers modify their programs to run faster. Furthermore, a certain amount of

code optimization during the compilation process can also improve performance. For example, field access can be inlined, and certain classes and methods can be made final during final packaging of a system.

Threads

Given that threads are an integral part of the Java language, there has been a significant increase in thread use to solve problems more elegantly than within a sequen-

The Metamata Toolsuite

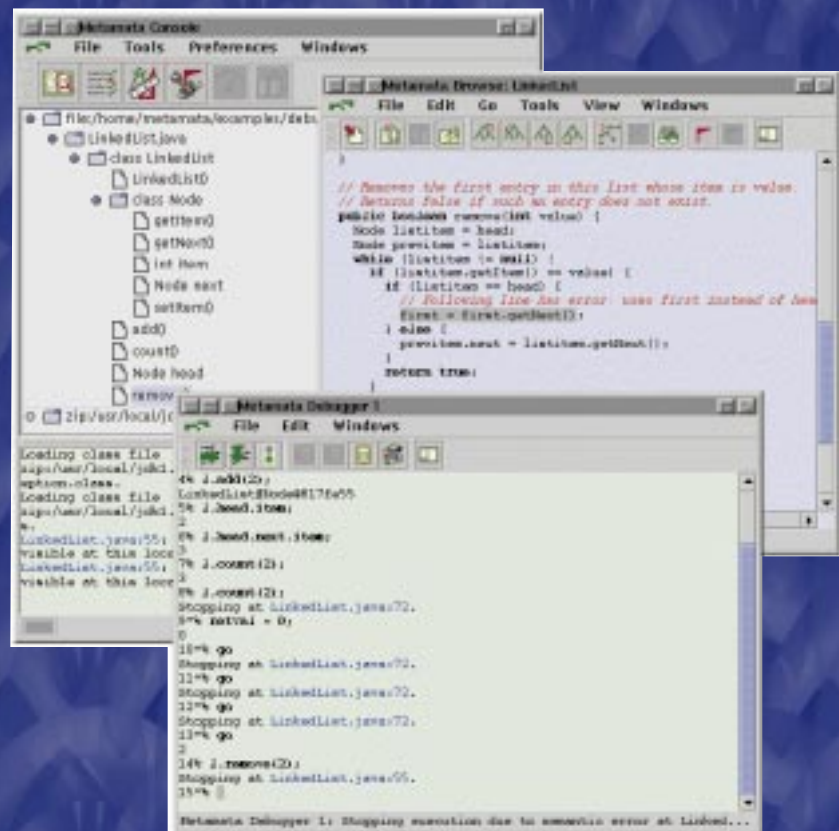
The Metamata toolsuite is a set of tools aimed at enhancing the productivity of Java developers and improving the quality of their Java software. The toolsuite complements and enhances the functionality of standard IDEs and has been designed to be used alongside them.

The toolsuite provides the following fundamental features:

1. Java source file processing: The Metamata toolsuite processes Java source files in all their detail, in addition to compiled Java object files (class files/bytecode files). This is in contrast to other tools that process only compiled Java files or depend on the JDK to perform this processing. By processing Java source files, Metamata's toolsuite offers developers two advantages. First, the toolsuite can be applied on Java

files that may fail to compile – many times due to problems that are identified by the toolsuite itself. Consequently, the toolsuite can be used for maximum gain much earlier in the software development life cycle. And second, all possible information about the Java source file is available to the toolsuite. Object files lose information (such as overloading details) and therefore tools that process only these are inherently limited in their capabilities.

2. Incrementality: The Metamata toolsuite is able to incrementally update its knowledge of a Java system when it's changed (for example, by editing files through an IDE).
3. Written in Java: The Metamata toolsuite is written completely in Java. It therefore addresses all the issues related to



CASE STUDY

tial framework. However, programming with threads is inherently more complex than sequential programming because there are more ways in which multiple threads of control can interact with each other and as many ways for things to go wrong. Furthermore, it's usually difficult to

reproduce a problem caused by the interaction between threads since multithreaded programs are nondeterministic in their execution.

A lot of research has been devoted to understanding the issues of multithreaded systems over the past 20 years. The Java language design offers state-of-the-art fea-

tures based on this research, which does contribute to simplifying thread-based systems. However, a good language alone is not enough – there's also the need for good debugging and analysis tools to facilitate a better understanding of how a multithreaded system works.

I believe the best way to deal with threads is to have a diagnostic capability in which probes are permanently inserted within the Java program. These probes save information pertaining to program execution, which can then be used later to analyze the program's behavior. This analysis can be performed to ensure that certain properties always hold (e.g., no two threads simultaneously execute a certain portion of code).

Safety

When building large systems a lot of assumptions are made regarding how the system works. If the system is correct, these assumptions are met by the system execution. However, since bugs in software are always expected, these assumptions may not always hold. The debugging process essentially means running the system in a controlled manner to determine if these assumptions are met, and looking for ways

The Metamata Toolsuite (continued)

portability. The toolsuite has already been ported easily to Windows and UNIX platforms.

The Metamata toolsuite currently includes four tools – a source code browser (Metamata Browse), a debugger and command line interpreter (Metamata Debug), a code quality analyzer (Metamata Audit) and a quality/complexity metrics evaluator (Metamata Metrics). Additional tools planned later this year include a memory and time profiler, and a packager.

Metamata Browse

Metamata Browse is a source code browser that understands the semantic structure of Java. It allows navigation through source code in an intelligent manner (such as locating declarations and uses of variables).

Metamata Debug

Metamata Debug is a Java debugger and command-line interpreter especially geared toward complex and mission-critical systems. It enables rapid prototyping and debugging of partially written programs. It offers all the standard debugging features such as breakpoints, watchpoints, single-stepping and evaluating expressions. The command line interpreter allows one to type free-form Java code and get it executed, thus greatly simplifying the debugging process – there is no need to write a specific main program to perform execution. It provides a good user interface for thread debugging, with separate command windows for each thread.

Metamata Audit

Metamata Audit evaluates code for programming errors, weaknesses and style against a set of standard principles that define good coding practices. It helps improve the performance and quality of the code and makes it more uniform, easy to understand, robust, extensible and clean. It provides hyperlinks into Metamata Browse to obtain more details of the errors. Metamata Audit implements approximately 50 rules from a set of 150. The remaining 100 will be implemented over the next few months. Examples of what it checks for are unnecessary type

casts, catching too general an exception, excessive visibility and dangerous over-loadings.

Metamata Metrics

Metamata Metrics calculates global complexity and quality metrics on portions of code. It provides meaningful object-oriented metrics tuned for Java to enhance software processes and quality of team-based projects, optimizes testing and maintenance resources and improves project planning activities. Information is displayed graphically, as well as saved as reports.

Metamata Metrics implements all the standard complexity metrics that have been well studied in the QA community as indicators of bad organization, bad



structure, maintenance problems, etc. Metrics such as cyclomatic complexity and lines of code offer a more syntactic insight into complexity, while metrics such as lack of cohesion of methods and coupling between objects go further to understand the semantic structure of the software. ☺

to make corrections when they aren't.

In mission-critical systems some assumptions are important and require enforcement. Similarly, in multithreaded systems where it's often impossible to reproduce a problem, the violation of any assumption must be reported.

Providing diagnostic APIs solves this problem, allowing assumptions to be built

into the program as constraints that must hold during the program's execution. Tools to help manage them are needed to encourage users to write such constraints. One important capability necessary to encourage using diagnostic constructs is an easy way to strip out these constructs when it's time to package the system for final shipping.

Portability

Compiled Java files can be moved to different platforms and executed using different Java Virtual Machines with no recompilation required. To facilitate portability, the Java language definition has gone to great lengths to specify exactly how a Java program must run. Very little ambiguity remains.

Only a few problems exist in writing portable Java applications. The most important:

- Thread scheduling can vary from platform to platform. For example, one platform may provide small-time slicing for thread swapping while others may not. This can cause system liveness to differ on various platforms.
- Use of platform-specific notation – the most obvious example is to refer to a file as a raw string (such as "C:\METAMATA\Test.java"). Clearly the presence of such a string in a program will cause it to perform poorly on a UNIX platform.
- Bugs in Java compilers and Virtual Machines can cause an otherwise correct program to behave differently in different environments.
- User of nonstandard APIs: certain APIs are available on only a few platforms. Making your program depend on such APIs will (obviously) cause porting problems.

While these problems are really quite trivial when compared to the issues involved in porting programs written in other languages, the promise of "write once, run everywhere" exacerbates these problems when developers expect their Java program to run smoothly everywhere.

The only real way to solve this problem is to test Java systems on as many platforms as possible. In addition, several heuristics to writing portable Java programs have been developed over the past couple of years. Facilitating portable testing and checking Java programs for certain portability heuristics violations can help developers in writing portable Java programs.

Developing Multiplatform Applications

There's no better approach to facilitate this development than to perform development on multiple platforms. Ideally, individ-

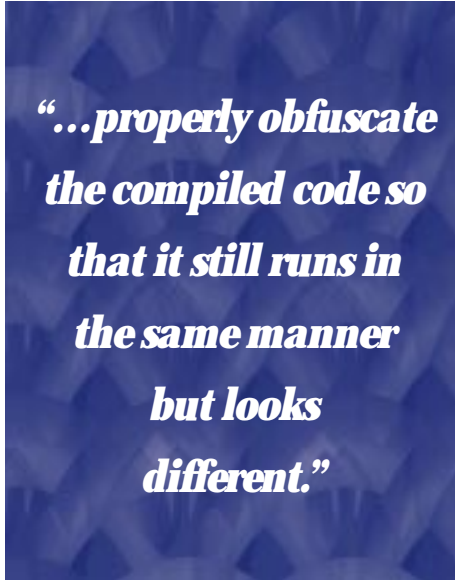
ual developers should already work on different platforms. It must also be possible for the same developer to move between platforms.

The message here is that development must be performed using tools that port to multiple platforms. An IDE that runs on only one platform can be severely constraining on a development team building multiplatform software.

Platform Accessibility

If a Java application is developed, taking care to ensure that it's portable, and then shipped for use by customers on a wide variety of platforms, you can be sure that customers will run the application on platforms you don't have access to. Furthermore, there are bound to be problems reported by these customers. Special care needs to be taken to ensure that it's possible to support them.

One thing to do is to ask the customer to



*“...properly obfuscate
the compiled code so
that it still runs in
the same manner
but looks
different.”*

run a general probing tool that provides full information on the customer's Java environment. It's also useful to have a version of the software that's heavily instrumented and then ask the customer to attempt to reproduce the problem using this instrumented version. Usually, it should simply be the shipped application running with a special environment setting. Then it's possible to study why the application runs differently on the customer's machine.

Build Reusable Libraries

Java encourages better organization and maintenance by making it much easier (compared to other languages) to build reusable libraries of software components. Widespread reuse of third-party components is common and developers tend to build ones that are as general and reusable as possible. This leads to a software bloat,

which occurs when the system contains a large amount of useful functionality that's never used by the system itself but exists for possible future use. In many cases it's difficult to identify the "system" from a set of reusable libraries.

As a result, it's necessary to trim these libraries down to only the essential pieces of code to enable systems to be packaged for release.

Obfuscation

Compiled Java code is rather high-level. Hence it's possible for someone to (illegally) reverse-engineer compiled applications. Therefore, care needs to be taken to properly obfuscate the compiled code so that it still runs in the same manner but looks different. There are a variety of ways to obfuscate Java code, from schemes as simple as changing the names of variables to sophisticated schemes where the compiled Java code is encrypted. However, regardless of the scheme used, it must be possible to interpret error messages, stack traces, etc., for customer support purposes.

Conclusion

Java has been adopted rapidly by both industry and academia, and software developed in Java is growing in complexity. The challenge to Java-tool developers so far has been simply to keep up with the pace of growth. Now they face a greater challenge of building tools designed to solve specific issues related to software development in Java, rather than simply to retrofit C and C++ technology for Java. Over the next few months we should see many new and exciting tools to solve problems related to, for example, garbage collection, performance and portability.

One year ago a 100,000-line Java program was large and there were only a handful of them. If a system could handle tens of thousands of lines of code, it was good enough. Today many Java programs exceed 100,000 lines, and soon we should start seeing a few programs reach a million lines of code. Tool builders will therefore be required to scale up their tools to handle such large systems efficiently.

We should also see new Java Virtual Machines capable of running much faster than current ones and significantly closing the gap between native code and interpreted execution. There'll also be native Java compilers that can compile code to execute natively on a platform-by-platform basis. The performance issues will essentially disappear once this happens.

These are very exciting times indeed for the Java community and I look forward to a more mature set of Java developer tools at next JavaOne! ☺

**Cold
Fusion
Journal
.com**



Active Menus Without Graphics

Active menus for Web pages using cascading style sheets, layers and JavaScript

by Ken Jenks

There's a problem faced by all Web designers: making a menu look interesting without taking forever to load.

Many modern applications address this problem using active menus that indicate where the mouse is by changing the color of the menu items as the mouse cursor passes over them. (Pull down any standard menu on a Windows machine to see this in action.) On Web pages this is often done by a JavaScript application that swaps images, but this technique is slow, and it doesn't work in all browsers.

In this article I'll show how to build this kind of active menu with text, not graph-

ics. This "change style on hover" feature is built into Dynamic HTML in Microsoft Internet Explorer, but in Netscape Communicator 4 this trick requires cascading style sheets, layers and JavaScript.

This technique saves download time by using text menus instead of images. There's only one connection from the client to the server, and the text is much smaller than an image of text, even with a few hundred bytes of JavaScript. In addition, since the entire menu is in one HTML file, configuration control is easier when you need to update your menu.

The Easy Way

Doing this in Microsoft Internet Explorer 4 is easy, thanks to style sheets. There's a quick example shown in Listing 1.

There are three things to notice here. First, we're using cascading style sheets, as described at www.microsoft.com/msdn/sdk/inetsdk/help/dhtml/dhtml.htm.

Second, the STYLE tags use HTML comments, `!--` and `-`, to hide the style descriptions from older browsers.

Third, the hover pseudoclass is defined with a gray background and italic font. See www.eu.microsoft.com/msdn/sdk/inetsdk/help/dhtml/references/css/hover.htm for details.

But this doesn't work in the most popular browser, Netscape 4. To do this, we need to mix several techniques: a little JavaScript, Netscape's inline layers and style sheets.

◆ LISTING 1.

Change Link Color on Hover, MSIE 4 Only

```
<!--
A { text-decoration: none; font-weight: bold; color: blue; font-family: Helvetica, Arial, sans-serif; }
A:hover { background: #AAAA80; font-style: italic; } /* MSIE Only
*/
-->
```

```
<BODY BGCOLOR="#8080FF">
```

Change Link Color on Hover, MSIE 4 Only

```
<A HREF="http://raven.ubalt.edu/features/poe/poecanat.htm">E. A. Poe Society of Baltimore</A>
<LI><A HREF="http://www.poemuseum.org/">Poe Museum</A>
</UL>
```

◆ LISTING 2.

Change Link Color on Hover

```
<STYLE TYPE="text/css">
```

```
<!--
A { text-decoration: none; font-weight: bold; color: blue; font-family: Helvetica, Arial, sans-serif; }
A:hover { background: #AAAA80; font-style: italic; } /* MSIE Only
*/
.whitelink { color: white }
-->
```

```
<BODY BGCOLOR="#8080FF">
```

Change Link Color on Hover

```
<SCRIPT LANGUAGE="JavaScript">
```

```
<!--
do_the_layers = 0;
if (document.layers) { // Netscape 4.0 and friends
  if (navigator.userAgent != 'Mozilla/4.03 [en] (X11; U; SunOS 4.1.3_U1 sun4c)') { // Exclude buggy version
    do_the_layers = 1;
  }
}
layer_number = 0;
```

```
// Calling this function like this:
// hover_link("http://tale.com/", "Stories");
// ...will create a new layer with a link like this:
// <A HREF="http://tale.com/">Stories</A>
// ...but the link will turn gray when the user holds his mouse over it.
// The var layer_number ensures a unique layer name (layer1, layer2...)
// for each link.
```

```
function hover_link(whi ch_url, what_content) {
  hover_link_style(whi ch_url, what_content, "");
}
```

```
// Just like hover_link(), but this one applies <SPAN CLASS=whitelink> to link
```

```
function hover_link_white(whi ch_url, what_content) {
  hover_link_style(whi ch_url, what_content, "whitelink");
}
```

```
// Just like hover_link(), but this one applies the given STYLE to link
```

```
function hover_link_style(whi ch_url, what_content, whi ch_style) {
  if (do_the_layers) { // If Netscape 4, create an in-line
```

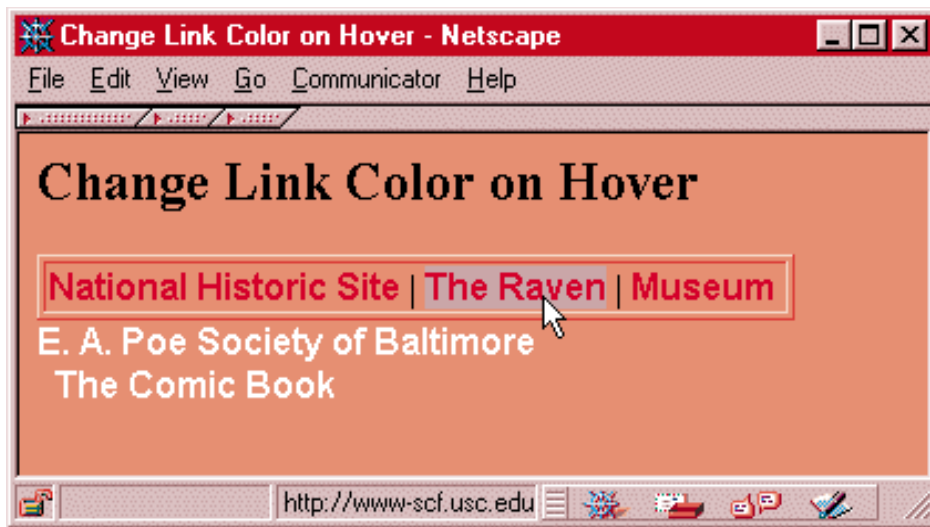


Figure 1: The active menu in Netscape 4

The Hard Way

Unfortunately, Netscape 4 doesn't allow you to change the style of an element on a page after it's been rendered on the screen, so this hover thing doesn't work. But it does let you change the contents of layers at any time. See <http://developer.netscape.com/docs/manuals/communicator/dynhtml/contents.htm>.

Each element of the following JavaScript application works in both MSIE 4 and Netscape 4, and is downward-compatible with other browsers, as seen in Listing 2, instead of having a link as in Listing 3.

In MSIE and older versions of Netscape, the `onMouseOver` event triggers the JavaScript `make_bg_gray()` function, but because the global var `do_the_layers` is zero, this will do nothing. (Same with `make_bg_null()`.) MSIE 4 will activate its

```
//layer...
whi ch_layer = 'layer' + layer_number;
layer_number++;
document.write('<ILAYER ID="' + whi ch_layer + '">');

q_l ayer = '\\' + whi ch_layer + '\\';
document.write(
  '<A HREF="' + whi ch_url + '" ' +
  'onMouseOver="make_bg_gray(' + q_l ayer + ')" ' +
  'onMouseOut="make_bg_null(' + q_l ayer + ')" ' +
  '>');
if (whi ch_style == "") {
  document.write(what_content);
} else {
  document.write('<SPAN CLASS="' + whi ch_style + '>' +
    what_content + '</SPAN>');
}

document.write('</A></ILAYER>');

} else { // If not Netscape 4, just make a normal link
  document.write('<A HREF="' + whi ch_url + '">');

  if (whi ch_style == "") {
    document.write(what_content + '</A>');
  } else {
    document.write('<SPAN CLASS="' + whi ch_style + '>' + what_content
+
    '</SPAN></A>');
  }
}

function make_bg_null (whi ch_layer){
  if (do_the_layers) {
    document.layers[whi ch_layer].bgColor = null;
  }
}

function make_bg_gray (whi ch_layer){
  if (do_the_layers) {
    document.layers[whi ch_layer].bgColor = '#AAAAAA';
    // setTimeout because onMouseOut event doesn't always trigger
    setTimeout(make_bg_null, 3000, whi ch_layer);
  }
}
//-->
</SCRIPT>

<TABLE BORDER=1><TR><TD>

<SCRIPT LANGUAGE="JavaScript">
<!--
  hover_link("http://www.nps.gov/edal/", "National Historic Site");
  </SCRIPT>
</NOSCRIPT>
<A HREF="http://www.nps.gov/edal/">National Historic Site</A>
</NOSCRIPT>

|

<SCRIPT LANGUAGE="JavaScript">
<!--
  hover_link("http://www.scf.usc.edu/~khachato/poeraven.html", "The
  Raven");
  </SCRIPT>
</NOSCRIPT>
<A HREF="http://www.scf.usc.edu/~khachato/poeraven.html">The
  Raven</A>
</NOSCRIPT>

|

<SCRIPT LANGUAGE="JavaScript">
<!--
  hover_link("http://www.poemuseum.org/", "Museum");
  </SCRIPT>
</NOSCRIPT>
<A HREF="http://www.poemuseum.org/">Museum</A>
</NOSCRIPT>

</TD></TR></TABLE>

<SCRIPT LANGUAGE="JavaScript">
<!--
  hover_link_white("http://raven.ubalt.edu/features/poe/poecanat.htm"
  , "E. A. Poe Society of Baltimore");
  </SCRIPT>
</NOSCRIPT>
<A HREF="http://raven.ubalt.edu/features/poe/poecanat.htm">E. A.
  Poe Society of Baltimore</A>
</NOSCRIPT>

<BR>
&nbsp;
<SCRIPT LANGUAGE="JavaScript">
<!--
  hover_link_white("http://pariah.simplenet.com/Poe/poe2.html", "The
  Comic Book");
  </SCRIPT>
  </NOSCRIPT>
  </TD>
  </TR>
  </TABLE>
  </SCRIPT LANGUAGE="JavaScript">
  <!-->
```

```
</SCRIPT>
<NOSCRIPT>
<A HREF="http://pariah.simplenet.com/Poe/poe2.html">The Comic
Book</A>
</NOSCRIPT>

</BODY>
</HTML>
```

```
hover_Link("http://pariah.simplenet.com/Poe/poe2.html", "The Comic
Book");
//-->
</SCRIPT>
<NOSCRIPT>
<A HREF="http://pariah.simplenet.com/Poe/poe2.html">The Comic
Book</A>
</NOSCRIPT>
```

◆ LISTING 3.

```
<A HREF="http://pariah.simplenet.com/Poe/poe2.html">The Comic
Book</A>
```

We'll need a link like this:

```
<SCRIPT LANGUAGE="JavaScript">
<!--
```

▼▼▼▼▼ CODE LISTING ▼▼▼▼▼
The complete code listing for
this article can be located at
www.JavaDevelopersJournal.com

hover pseudoclass normally, giving us the same effect as in the simple previous example.

But Netscape 4.0 (and any other JavaScript-compatible browser supporting layers) will use the onmouseover and onmouseout events to trigger make_bg_gray() and make_bg_null() to change the background color of the inline layer (ILAYER). Slow computers like my old 486 don't always trigger the onmouseout event, so we always turn the background back again three seconds after entry. Setting the background color to null instead of document.bgColor makes the area transparent, in

case you're using a background image on the page. (Because of the way we styled the hover pseudoclass, MSIE 4 will show the text as italic on hover; Netscape 4 will not.)

Unfortunately, a bug in the X Window System/SunOS version of Netscape 4 makes it crash when this code is invoked, so we've put in a little guard code, showing why it's always good practice to test your JavaScript code under all available browsers.

Conclusion

This little program demonstrates some of the advanced features of MSIE 4 and Netscape 4, while retaining backward com-

patibility with older browsers. If you find a use for these active menus or the techniques you've learned here, drop me a line. ☛

About the Author

Ken Jenks has been programming for more than 23 years. He holds a BS in computer science, an MS in aerospace engineering and is working on a Ph.D. in mechanical engineering. In his day job he works for the federal government. Evenings and weekends he runs a Web-based publishing company, Mind's Eye Fiction (visit <http://tale.com>). He can be reached via e-mail at MindsEye@tale.com.



MindsEye@tale.com

1/2 Ad

JDJ o JavaDeveloper

online!
rsJournal.com

Handling the Load: Peak Application Performance with JavaLoad Load Testing

by Matt Evans

Henry Ford revolutionized manufacturing with the first automated assembly line in 1913, incorporating the latest time-study theories to make Model T's flow through the system "like the flow of water through a pipeline." His engineers and foremen repeatedly analyzed every task in the assembly process to find ways to save valuable inches and seconds.

A lot has changed over the last century. The assembly lines of businesses today are high-performance client/server computer networks – including the Internet – that move information quickly and efficiently, speeding communications and shortening the business cycle. Implementations of Java applications and applets have emerged as the leading technology to implement client/server and Internet-based applications.

Just as Ford's engineers and foremen kept assembly lines moving smoothly, a process called load testing helps system developers isolate potential bottlenecks or problems in Java-based applications and applets. JavaLoad software from Sun Microsystems helps developers understand how these applications will behave and react under a variety of load conditions before being deployed.

JavaLoad Software

From a single point of control on any tested client, anywhere in the network, JavaLoad software stresses and monitors both the application under test and the network to isolate problem areas at every system level.

JavaLoad software creates virtual users to perform a specific sequence of actions that test the main functions of an application. It's multiple "users" generate workload from a series of client workstations or PCs to help answer critical questions about network and application capacity and scalability such as:

- How many concurrent users can the server or application handle?
- How does system performance respond to configuration changes or system upgrades?

- What happens during peaks in usage?

JavaLoad software load-tests both Java and non-Java enterprise applications across large, heterogeneous environments to ensure that any Java-based distributed application is rock solid – end-to-end.

JavaLoad Software's Commander Network

The commander network is the heart of JavaLoad software. Commanders generate repeatable, targeted, client-side activity and constitute a self-maintaining, distributed load-testing network. They're deployed on each host in the test network to manage and conduct the activities of the load test.

A Central Commander starts the load test on all the hosts and maintains the network time stamps, synchronizing virtual users

across the network to begin at the same time. It also collects the results of the load test from the individual commanders for deposit in a central repository, where they're stored for future analysis alongside reusable test component files that can be used to construct future tests, saving time and money.

The effects of the generated load are measured through JavaLoad software's Telemetry Service and executed by the commanders (which can be used on every platform – client, middle-tier or server). They can record:

- The response time of any type of client transaction
- The number of active simulated users
- Memory and thread use of any Java program involved in the application under test, server or client
- System-level statistics including CPU use, disk use and network performance.

The service can be customized to collect specific application-level statistics and can graphically display the telemetry data in real time, or archive the data for postanalysis and hard-copy report generation.

System Requirements

Supported Platforms:

Solaris, Windows NT and any platform containing a certified Java-compliant Virtual Machine supporting JDK versions 1.1.4, 1.1.5 and 1.1.6.

System Requirements:

- 3 MB RAM per JVM
- 3 MB RAM per GUI client user
- 60 KB per network user
- Any additional requirements of the system under test

The SunTest Suite Family

JavaLoad software is just part of the testing solution. Sun's SunTest Suite of Java testing tools provides an integrated test environment that lets developers leverage the full power of Java technology in each phase of an application's development.

Additional Java Test Tools from SunTest

- **JavaStar:** SunTest's automated software testing tool for testing Java applications and applets through their GUIs. JavaStar is integrated to run under JavaLoad software's control to capture load and performance data for the GUI users.
- **JavaScope:** SunTest's powerful, but easy-to-use, test coverage analyzer. JavaScope functions in the background during testing to monitor and ensure thorough test coverage.
- **JavaSpec:** SunTest's powerful tool for API or class testing of Java applications helping you perform formal specifications testing.

JavaLoad Console and Session Profiles

The JavaLoad Console is the powerful graphical user interface that controls and continuously monitors the JavaLoad software load-test network across the enterprise. It's the portable point of control from which developers can design, launch, monitor and analyze the results of load-testing sessions.

The JavaLoad Console is the central place where the separate components of a load test – tasks, telemetry channels, virtual users and commanders – are pulled together into a Session Profile. The Session Profile instruction file defines what each particular load-test run should do, providing a blueprint for each aspect of the test such as:

- Which tasks are to be performed
- Which telemetry channels to use for system measurement
- The session's number of virtual users
- Which commanders are included in the test

To ensure that the right changes are made to right problems without creating new ones, JavaLoad assigns a new session ID each time the load test is run so that each set of results can be compared with earlier tests.

JavaLoad software leverages the animation capabilities of Java technology to display real-time graphical reporting. The JavaLoad Console also generates standard and customizable load-test reports for comparing different test sessions.

WebLoader and DataProxy plug-ins are available with the software, allowing it to record and replay HTTP events and JDBC and Socket messages as tasks within a normal Session Profile.

WebLoader: No Browser Required

WebLoader captures and replays HTTP events (the kind that occur on any Web browser) to test a Web or proxy server's performance and load-test Web-based applications.

WebLoader captures and records a browser's interaction with any number of Web servers in the form of HTTP events, independent of the particular type of browser used. It then creates a Java program that faithfully replays these recorded events without the need for a browser within a regular Session Profile.

The freedom of replaying HTTP events without a browser means the WebLoader playback program can be easily replicated and distributed over the entire test network to simulate a few dozen to thousands of users concurrently accessing a Web-based application.

DataProxy: Java Database and Socket Messages Captured

DataProxy enables JavaLoad software to capture Java Database (JDBC) and Socket messages in a form that can be replayed as normal tasks within the load-test Session Profile. DataProxy controls the launch of a Java application in order to capture JDBC or Socket network traffic traveling between client and server, regardless of protocol. Once it intercepts the messages, DataProxy records them to a data file and generates Java source code for a JavaLoad program that's able to multiply and replay the JDBC or Socket messages within a Session Profile. The server receiving the messages captured by DataProxy can't detect any difference between the replicated messages and the originals.

JavaLoad software's ability to simulate a wide variety of protocols frees developers to generate significant system loads without excessive amounts of hardware testing.

Load Testing in Heterogeneous Environments

Today, applications in many sectors, including e-commerce, the Web, banking, telephony, order-entry, ERP, call center and manufacturing, need load testing. Because Java technology links together heterogeneous environments, load testing these applications becomes more important than ever. With virtue of JavaLoad software being written in Java programming language, developers don't have to pay extra for each platform. This means all Java-compatible platforms can be used for load-testing, which yields a more thorough and realistic load-test of both client and server. Sun Microsystem's JavaLoad software can be used throughout the product development life cycle, and is unique among load-testing tools because it can validate architectures, benchmark hardware and stress the integrated system.

For more information about the JavaLoad member of the SunTest suite contact your SunTest product specialist, or call 1 888-TEST-JAVA. ☎

About the Author

Matt Evans is the principal developer of JavaLoad as well as the manager of the SunTest tool suite engineering team at Sun Microsystems in Menlo Park, California. With Sun for more than eight years, Matt focuses primarily on development and management of projects in the area of software verification technologies. Matt can be reached at matt.evans@sun.com.



matt.evans@sun.com

ObjectMatter.com



Persistence in Enterprise JavaBeans

Bean- and container-managed EntityBean persistence and their merit

by Patrick Ravenel

Persistence in Enterprise JavaBeans is encapsulated in the notion of EntityBeans. This article describes bean- and container-managed EntityBean persistence and the relative merit of these techniques with respect to portability, productivity and performance.

Sun Microsystems produced the Enterprise JavaBeans specification (version 1.0) in March 1998. According to its cover page, the goal of the specification was to provide a "component architecture for the development and deployment of object-oriented distributed, enterprise-level applications." Two important conceptual components were described by this architecture: "EnterpriseBeans" and their "container." EnterpriseBeans are of two varieties: SessionBeans and EntityBeans. SessionBeans don't have persistent state; EntityBeans do. In other words, you can't expect the state of SessionBeans to outlive their process. EntityBeans on the other hand have some or all of their state persist between incarnations.

EJB Architecture Overview

To discuss EntityBeans and their persistence, we first have to outline the components of the EJB architecture that will be important to our discussion. This section won't attempt to describe all the components, only the relevant ones.

From a bean implementer's point of view, the first decision is whether the bean is going to be a SessionBean or an EntityBean. Both can have business logic but, generally speaking, SessionBeans are unshared servants to a client and have no persistent state. EntityBeans are shared among clients and do have persistent state.

For every bean there is a set of other interfaces that must be in place for the bean to be accessible to clients. Each bean must have a mandatory remote object interface as well as a Home interface (think factory). The EJBObject-derived interface is

a remote interface that typically delegates calls to the bean; the EJBHome-derived interface contains bean lifecycle and management functionality. The home interface is also remote.

You're probably asking yourself, Where does this so-called "container" fit in? Well, from a bean-provider perspective, everything that facilitates the life cycle of the bean and the dispatch of a remote client method invocation to your bean is "the container." For the purposes of this article, this includes the EJBObject- and EJBHome-derived remote interfaces (which are application-dependent) as well as services like the implementation of JTS and JNDI (which are application-independent) that your bean can depend on. Some containers may also support container-managed persistence. This is an API that an EntityBean developer can use to delegate the persistence of the bean to instead of writing all of the persistence himself (by using JDBC or JSQL).

Figure 1 shows the relationship of an EntityBean (with container-managed persistence), its container and the client. In this figure the EmployeeBean is the implementation of all the logic for the bean. Employee is just a remote object interface declaration and the same is true for EmployeeHome. These three elements along with a DeploymentDescriptor (described later) are supplied by the bean provider and are application-dependent. Tools supplied by the container provider will typically generate the implementations of the remote interfaces.

These application-dependent components wrap the bean for the container. We say they "wrap the bean for the container" because they're mandatory and you can access the bean only through them.

Application-independent components are also part of the container, of course. Figure 1 shows that developers can take advantage of the application-independent

APIs, that is, JTS, JNDI and the container-managed persistence (CMP) API. Notice that in this figure, which depicts container-managed persistence, the container can add connection pooling and transactional caching as well as dynamic schema management. These capabilities provide a lot of transparent functionality to the bean with container-managed persistence.

Figure 2 is very similar except that the bean provides its own persistence. Most of the rest of the container is intact, but the ability to provide caching, dynamic schema management and connection pooling transparently is lost. So in this figure the bean is fatter because to write these capabilities on top of JDBC would be a tremendous burden on productivity. Plus your bean's code is just plain fatter.

Since this article is mainly about EntityBeans and how their persistence is managed, we're going to skip the discussion of SessionBeans except to reiterate that they're unshared and typically don't have persistent state. They may, however, have transactional, conversational state. SessionBeans will typically contain most of your session-based (unshared between clients) business logic. The good news is that they'll typically make use of other EnterpriseBeans through remote-object interfaces. This leads to a highly scalable federated architecture.

The Development/Deployment Process

As you can see, there is a substantial amount of code needed to make an EJB application server of any complexity. You may wonder whether you need to write all that code. Well, the development process that you'll follow will be highly dependent on the tools supplied by your EJB application server and development tool vendor (which may be the same). A large part of the code required is the glue that plugs your bean into the container of your EJB app server vendor. The fine folks who wrote the EJB specification provided a nifty item called the DeploymentDescriptor. This serialized class is provided for each bean and contains enough meta data for tools to gen-

erate the framework to plug the bean into an EJB container. This would include the EJBHome, derived class, the remote object interface and other hooks to manage the bean.

Contained in a bean's DeploymentDescriptor is a ControlDescriptor for each method. The descriptors contain meta information about the bean and its methods with respect to their behavior in several dimensions (e.g., transactional, security).

With the state of EJB implementations today, there are basically three approaches to EJB development: (1) hand-code everything, (2) hand-code the bean and the deployment descriptor and (3) generate most of the bean and the deployment descriptor from a graphical environment. Let's look at these in a little more detail.

Hand-Code Everything

This means that the EJB app server doesn't support any auto-generation tools but does provide the API for its container. In this case you'll be expected to write implementations for the beans, their remote object interfaces, their home interfaces and any other hooks required by the vendor's API.

Hand-Code the Bean and Descriptor

EJB was written with this in mind. By producing a EJB-jar file containing compiled code for beans, their remote object and home interface declarations, and their associated serialized DeploymentDescriptors, a tool could be used to generate the rest of the infrastructure. In this manner the container's API doesn't necessarily need to be exposed to the programmer. Rather, the code generator takes care of this mapping. Figure 3 illustrates this approach.

Graphical Bean and Descriptor Generation

For ease of use and EJB-jar file generation from a graphical environment can be employed. In this development environment you simply create an object model of the beans and their relationships. The graphical environment takes care of generating the .jar file containing the beans and their descriptors. Tools like PowerTier for EJB from Persistence Software, Inc., take this graphical approach and not only generate the container framework but also an RDBMS-independent object-relational map-

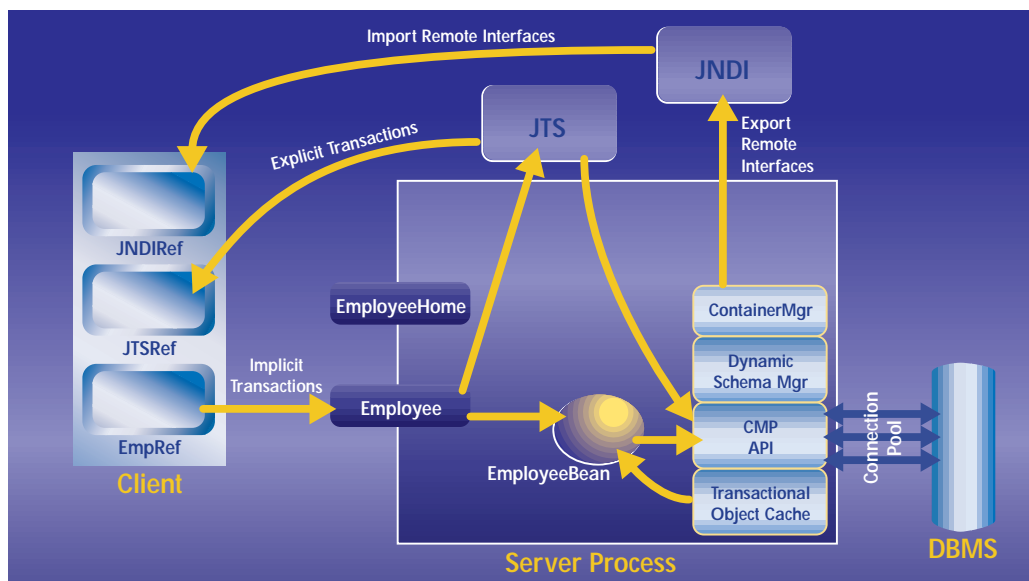


Figure 1: Container managed persistence architecture "thin bean-fat container"

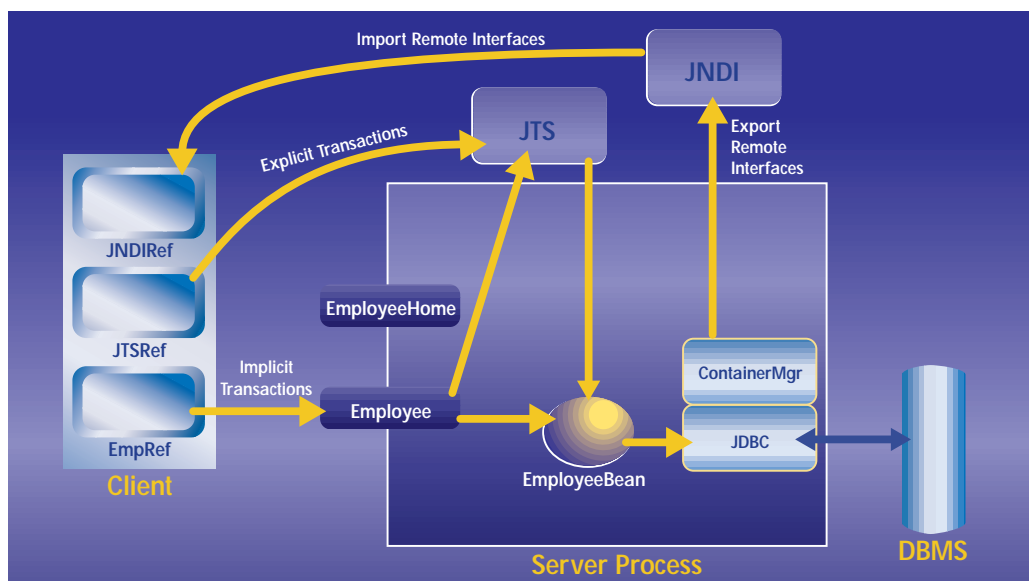


Figure 2: Bean Managed Persistence Architecture "fat bean-thin container"

ping for EntityBeans. Figure 4 illustrates this approach.

The bottom line with respect to generation is that once you have the EJB-jar file container-specific tools can then generate the framework for adapting beans to their container.

EntityBeans: The Persistence in EJB

Given an understanding of how EntityBeans fit into the EJB architecture, the choice that a bean developer has to make is between bean-managed and container-managed persistence. How do you choose between them? Well, let's look at the relative merits of the approaches with respect to productivity, performance and portability. In a sense, the choice for developers is a choice between fat beans, where the bean is smart about data management, and fat servers, where the EJB container is smart about data management.

Container-Managed Persistence

For container-managed persistence the EJB container automatically implements the object-relational mapping services for the bean. The EJB container uses additional meta information in the deployment descriptor to determine how to implement the object-relational mapping for the bean. This makes the bean itself "thin" because the bean only needs to contain the custom business logic added by the developer. Figure 2 shows this notion of "thin" beans and a "fat" container. Figure 1 shows this type of architecture.

The main advantages of container-managed persistence are performance, runtime/development-time flexibility and productivity.

EJB containers with container-managed persistence can provide data management services that are unavailable with bean-managed persistence, including:

- *Object-relational mapping*: automates the task of mapping complex beans – including inheritance, aggregation and association – to relational tables.
- *Shared transactional caching*: many clients can share access to beans in the EJB container with the same transactional integrity provided by a database.
- *Object state management*: container-managed persistence can keep track of which objects have been changed within a transaction. In bean-managed persistence it's up to the developer to manage this, usually by setting a dirty flag in each object.
- *Object management optimizations*: the container can support joint database queries in the database that return networks of heterogeneous beans and enable navigation between bean classes in the container cache.
- *Data management optimizations*: the container can defer write operations in a transaction until commit, then batch multiple database operations into one call.
- *Connection management*: the container can transparently manage database connections and implement native database operations for optimal performance.
- *Container-based agents*: the container can monitor events within the server and notify clients when specific events occur.

“...one severe limitation is that it hardwires the mapping to a particular database schema into the guts of the bean.”

To develop a class with container-managed persistence, the developer performs the following steps (this example describes the Persistence Software, Inc., PowerTier for EJB container):

1. Define the object model and object-relational mapping using a CASE tool.
2. Use the EJB container tools to load the modeling information from the CASE tool and generate beans whose deployment descriptors and implementations allow them to use container-managed persistence.
3. Add custom code to the bean class (code insertion points protect the custom code when the bean is regenerated).

The generated beans implement all required EJB methods but add convenient query methods and methods for complex relationship management. The container provides transparent connection management, exception handling, integrity and transactions.

In addition to enabling rapid application development, container-managed persistence also makes enterprise bean classes independent from the database schema. Thus beans that use container-managed persistence are more flexible and are portable across data schemas. Changes to the object model or database schema are handled by the container's object-relational mapping, providing a high degree of flexibility and support for iterative development. Finally, containers whose implementation of Container Managed Persistence includes shared and transactional caching can provide extreme performance and scalability advantages because they can manage concurrent transactional access to the bean from multiple clients.

The main disadvantage is that they aren't as portable as bean-managed persistence beans with respect to the current EJB specification. This is mainly because EJB is underspecified in the area of a portable

container API for container-managed persistence.

Bean-Managed Persistence

For bean-managed persistence the developer writes database access calls using JDBC and SQL directly in the methods of the bean. This makes the bean itself “fat,” requiring hundreds of lines of developer code to implement each bean. On the other hand, this makes the EJB container relatively “thin,” giving it a small footprint and high portability. Figure 2 showed this notion of the “fat” bean with a “thin” container.

The main advantage of bean-managed persistence is its portability. A bean managing its own persistence using JDBC or JSQL is very portable, especially with respect to the current state of the EJB specification.

The main disadvantages of the bean-managed persistence approach is that writing the object-relational mapping by hand can be time consuming and requires extensive knowledge of SQL. For example, the developer must hand-code database access calls that implement the object-relational mapping in the enterprise bean callback methods (ejbCreate(), ejbLoad(), ejbStore(), etc.).

Another severe limitation of this approach is that it hardwires the mapping to a particular database schema into the guts of the bean. This means that any change to the object model or the underlying database schema can require extensive changes to the code for a particular bean, limiting the flexibility and reusability of beans built using bean-managed persistence.

To develop a class with bean-managed persistence, the developer must perform the following steps:

1. Define the object-relational mapping for the bean.
2. Implement required EJB class methods. Using the JDBC API and SQL, write methods to perform ejbCreate(), ejbRemove(),ejbLoad(),ejbFind<>() and ejbStore(). This requires a detailed knowledge of SQL and the underlying database schema.
3. Implement accessor and mutator methods. For each attribute or relationship write methods to manipulate the bean value. Relationship access code to manage many-to-one relations and delete constraints can be very complex.
4. Manage database connections. Each method must manage its own database connections by interacting with the database connection pool.
5. Handle exceptions. Each method must handle both Java and database exceptions appropriately.

SYS-CON RADIO

Tune in for **LIVE** coverage of...

Java Business Expo & Java Developer's Journal Award Ceremony

Only from... **SYS-CON PUBLICATIONS**

www.sys-con.com

READER'S CHOICE AWARD

Spirit by eVisNet

<http://www.evis.net>

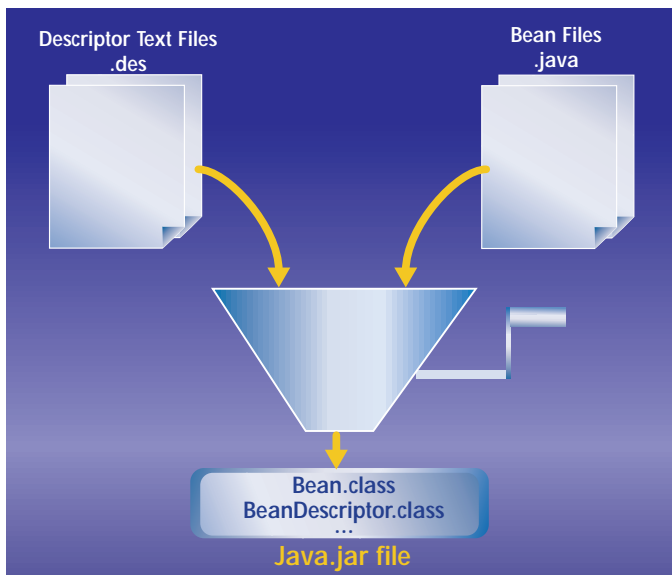


Figure 3: Hand-coding development process

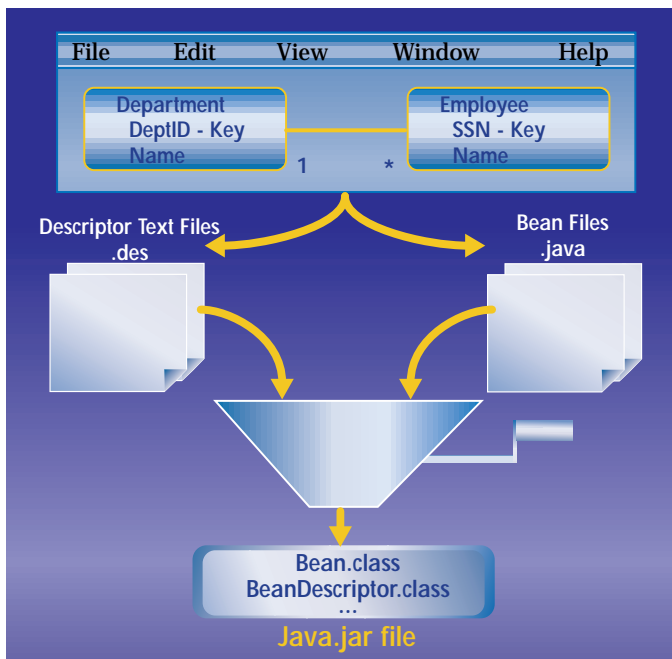


Figure 4: Graphical development process

6. Manage bean integrity. Any method that changes the state of the class must set a flag to mark the bean as modified so the changes will be sent to the database.
7. Ensure transaction integrity. Collectively, all the objects in any transaction must be able to "roll back" their state if a transaction fails.
8. Add custom code to the bean class.

Even if the SQL mapping were generated for you, the gains in performance and flexibility would not be as great as container-managed persistence unless you wrote all of the outlined features yourself...and then you've thrown out productivity.

Finally, because you typically won't find shared and transactional cache management in Bean Managed Persistence, it will have extremely poor performance and scal-

ability since multiple clients will block each other for the entire scope of their transactions. This can be costly even in the case of implicit transactions.

Conclusions

While bean-managed persistence provides the most portability, it will generally fall short with respect to productivity.

It should be clear at this point that container-managed persistence in EntityBeans provides the most productivity. This method will also typically provide the highest degree of performance since, together, the code generation tools and the bean programmer can take advantage of container-specific mapping and caching technology. However, container-managed persistence currently falls short in the area of portability.

Below are some guidelines for making decisions with respect to which kind of persistence management to use.

When to Use Container-Managed Persistence

Developers who'll benefit most from container-managed persistence are those requiring rapid application development, high reliability and scalable performance in their deployed application.

- **Rapid application development:** productivity is the number one reason for choosing container-managed persistence. For simple beans, bean-managed persistence can require 25 lines of hand-written code for every line of hand-written code required by container-managed persistence. For more complex beans the ratio is even worse.
- **Reliability:** the markedly smaller amount of code the developer has to write using container-managed persistence guarantees fewer defects and higher reliability.
- **Performance:** container-managed persistence allows use of shared transactional caching, deferred database operations

and native database implementation to optimize performance.

The kinds of applications that benefit most from container-managed persistence include:

- **Data intensive applications:** these applications tend to have larger and more complex object-relational mappings.
- **OLTP applications:** these often require good database performance, robust scalability and rock-solid data integrity.
- **Application integration:** which must map information from several different data sources.

When to Use Bean-Managed Persistence

The primary reason for using bean-managed persistence today would be to improve portability for beans across servers from multiple vendors. One known omission in the Enterprise JavaBean 1.0 specification is that it doesn't define a standard API for EJB containers to ensure portability among containers from different vendors.

This omission will have to be corrected in the Enterprise JavaBean 2.0 specification, as JavaSoft has announced that full support for entity beans will be mandatory in EJB 2.0. Even with bean-managed persistence, however, the number of differences between EJB 1.0 servers in areas such as security and transaction management are likely to make full portability difficult in the next 12 to 18 months.

Future Directions in Portability and Scalability

Container-managed persistence will be made portable through standard deployment descriptor formats that can help define the schema as well as standard metadata-driven internal container APIs for transparent, portable, container-managed persistence. This can be done in a fashion similar to the OMG's Persistent State Service specification.

Speaking of the OMG, CORBA integration will be essential for any EJB app server to be scalable to the enterprise. IIOP is the perfect unifying protocol to support EJB applications and is necessary to overcome some of the limitations of JRMP, the most serious of which are bugs in distributed garbage collection (IIOP would do away with DGC) and service context propagation (without this, it is difficult to implement JTS and other services). ☺

About the Authors

Patrick Ravenel is director of distributed computing for Persistence Software, Inc. He can be reached by e-mail at patrick@persistence.com.



patrick@persistence.com

InternetWorld '98

MecklerMedia

<http://www.mecklermedia.com>



Software Engineering in Startup Companies

The life-cycle stages and tracking of requirements

by Juergen Brendel

The discussion about software engineering in the special environment of startup companies continues with a focus on the software life cycle model and the tracking of requirements.

Software Life Cycles

According to classical software engineering (SE), the development of software takes place in stages. Each stage has distinct outputs, which can be tested before you proceed to the next stage. They are:

- **Analysis:** The problem and requirements for a solution are identified. *Main output:* Software requirements document.
- **Design:** A software system is designed to fulfill the previously identified requirements. *Main output:* High- and low-level design documents.
- **Implementation/coding:** The software system is implemented according to the previously defined design. *Main output:* Source code.
- **Testing:** Individual components as well as the entire system are tested for fulfillment of the requirements identified during the analysis stage. *Main output:* Test results.

Numerous models that describe the arrangement of the individual stages and the feedback among them have been suggested. These are called the *software life cycle* models. Some examples are the waterfall model, spiral model and incremental model, which are thoroughly discussed in SE literature (for example, *Software Engineering: A Practitioner's Approach*, 4th ed., by R. Pressman, McGraw-Hill).

Consciously following a life cycle model lends structure to an otherwise amorphous effort. When you can identify the end of a stage, you know the time has come to perform specific tests, tests that enable you to find errors at an early stage in the development process. A major design flaw that can

be fixed with just a stroke of a pen during the design stage may require major recoding if discovered when the software's almost finished. It's therefore important to perform these tests not just at the end of the development effort, but rather from the beginning and throughout the process. A lifecycle model facilitates this.

By testing the output of a stage, you provide a well-understood and firm foundation for the team to build on. Once such a foundation is set, it's not supposed to change. In the ideal case all team members know what to achieve next, since this was set forth in the previous stage in a nonambiguous manner.

In a startup company, however, the software life cycle is usually not well ordered. Markets develop swiftly, and requirements change even long after the analysis stage

has supposedly been completed. Time and time again the engineering department finds itself under pressure to do whatever it takes to provide new features originally not planned.

Is there a lifecycle model that not only works under these conditions but also helps to improve them? Of the many models developed, the incremental model seems to lend itself most closely to the way a startup company operates, but it requires a few modifications.

As you can see in Figure 1, individual releases of the software are developed in a "pipelined" fashion. In theory this allows the rapid release of new features for your software. The incremental model works well for conventional companies operating in established markets, which use it to reduce the complexity of an individual release. Many of the features for the next releases are already known through market observation, feedback from customers of other products, established marketing channels and so forth. The more established companies also have the resources to maintain multiple parallel development streams.

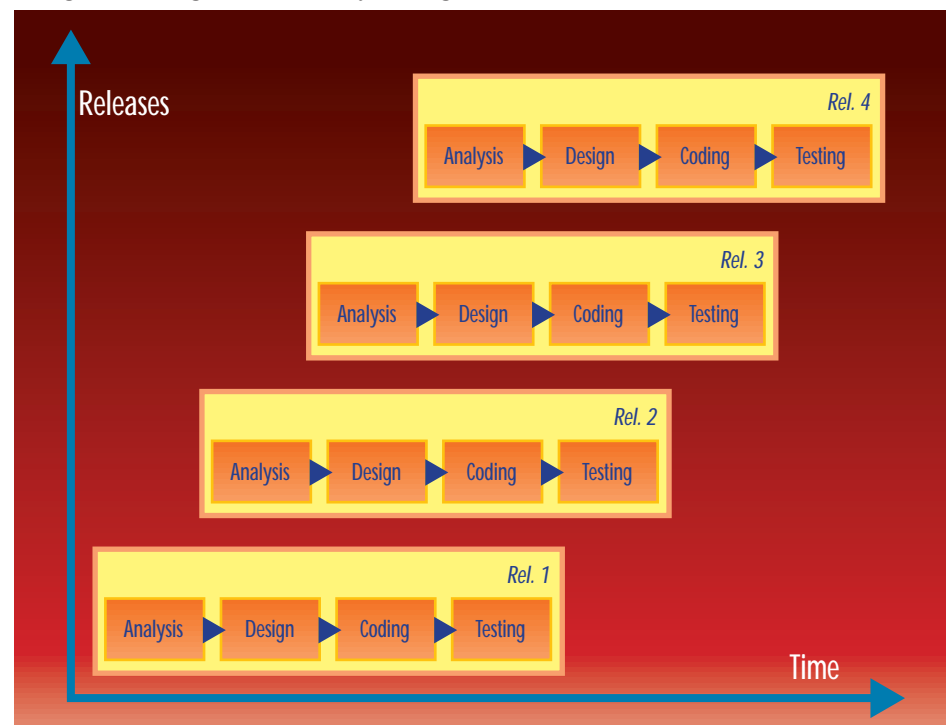


Figure 1: Classical incremental life-cycle model

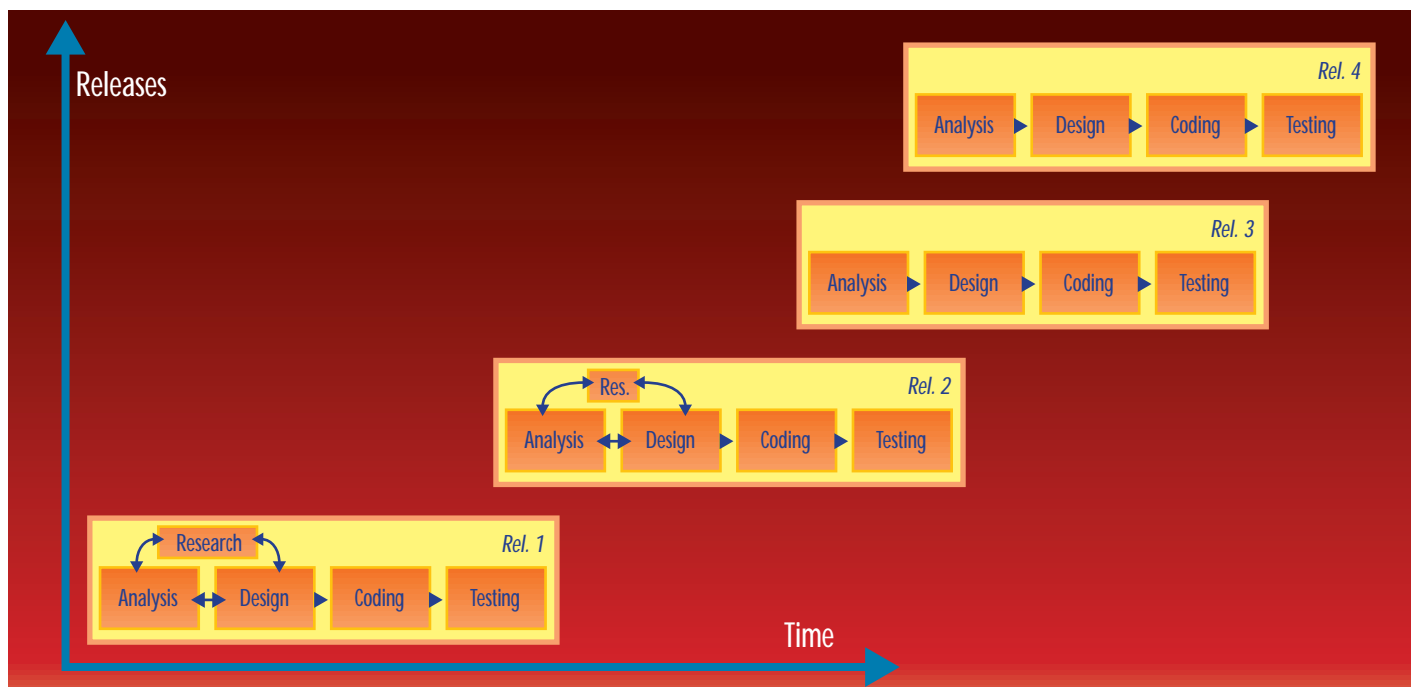


Figure 2: Modified incremental life cycle model

The startup reality renders this model impractical. Hiring qualified personnel is particularly difficult for a startup. It's unrealistic to assume that you'll hire a team of experienced analysts at the very beginning, followed first by designers and then by developers. In theory, software engineers should be able to handle all phases of product development. Unfortunately, the proliferation of this title throughout the industry has greatly reduced its value. Many people who call themselves software engineers really don't have a thorough software engineering education and often their experience is only in coding and maybe some design. I am in the same situation and am still learning. So while you can find many software engineers, those with the necessary skills, training and experience for all product-development stages are few and far between.

The overall head count in your company is likely to be very low for an initial period, only to increase quite rapidly later on. Thus, in the beginning, each developer is also in the position of analyst as well as designer. Obviously, given the lack of personnel, you may not be able to do the analysis for the next release during the design phase of the first release. You have neither enough resources nor sufficient market feedback to begin the development cycle of the second release right away. After all, you haven't even released the first version of your product. Occasional feedback is passed on to you by marketing and sales, gathered from discussions with potential customers. But you won't get true customer feedback until you've shipped the first version to beta customers. Compared

to established markets that provide a brightly illuminated playing field, a startup operates in the dark.

During analysis – and design – you may have to perform research to prove technical concepts or ideas on which you plan to base your product. This may be done in the form of a prototype, which provides feedback for the analysis and design stage, adding complexity to the initial development stages.

Quickly developing markets, initially missing customer feedback and lack of resources as well as analysis and design stages influenced by research lead me to suggest a modified incremental life-cycle model for startup companies.

As Figure 2 indicates, analysis, research and design are intertwined for the first release. Analysis for the second release begins at a later stage when two conditions have been met:

1. Requirements for the next release are available.
2. Enough new developers have been hired to free the most senior developers to work with marketing on the analysis stage of the next release.

The analysis for the second release starts after enough customer feedback has been collected to get a good feel for what the market wants. Without that feedback there's really no point in attempting to release yet another version of a product that may have had a lukewarm reception the first time around. The feedback is important and therefore needs to be properly analyzed and prioritized. You have to resist the temptation to stuff all requested

features into the next release.

Once you have a product on the market, you'll get a constant stream of requests. Thus, after the initial lag, you can start working on new releases earlier and earlier as staffing permits. The modified incremental life cycle model reflects this reality.

Also note that the research component becomes less with each subsequent release. The reason is simply that you've established a core technology with the first release from which you'll continue to leverage. Yours is a commercial company, not a research lab. It's important for you not to have too much research in the critical path of your project as you progress, since research can't be scheduled properly.

By keeping the first release small and simple, you'll receive market feedback sooner. Such early feedback is important to align the company and its product with the market. The longer it takes you to get feedback, the more time you spend developing "blind," in possibly the wrong direction. Provide the core functionality in the first release. The market will let you know in which direction to go. Potential customers are often willing to negotiate now if the fancy feature they want can be promised to them in an upcoming release. This lifecycle model sets you up for quick releases to satisfy customers without having to drastically change the requirements for an ongoing development cycle.

Tracking Requirements

As we discussed in the first part of this series (*JDJ*, Vol. 3, Issue 7), the analysis stage will not be as rigorous as classical SE would suggest. Many requirements are not

Requirements	Analysis	Design	Coding	Testing	Docs
Comm packet size <500 bytes	Spec, chapter 3.2	Design spec, chapter 3.3.4	SendPkt.send_it()	case #304, #305	chapter 1.9, "Sys.Requirements"
Pop-Ups for critical errors	Spec, chap. 5.5-5.8	Design spec, chapter 6.2.1, 6.3.9	Errors.display() Errors.pop_up()	case #238	chapter 3.7, "Exceptions"

Figure 3: Sample traceability matrix

known at all, or at least are not understood enough to formulate them in a quantifiable manner.

Yet it's important not to drop a requirement accidentally through simple oversight. Consequently you have to track as many requirements as possible and as completely as possible. The traditional tool used for this task is the "traceability matrix."

The matrix is essentially a table. The individual requirements are written from top to bottom and hence label the rows. The individual development stages (analysis, design, implementation, testing) are written from left to right and label the columns. Each cell of the table contains a record of where and how the requirement was addressed in that stage. For the development of product documentation, either a similar table should be created or documentation should become an additional column in the matrix.

At the end of each development stage each requirement should be checked to see whether it has been addressed during that stage. A look at the table will reveal any omissions, which would be very costly to

fix in later stages of product development. Such a matrix can save time and money.

A traceability matrix will provide a company with an important benefit. Since the matrix records the trace of each requirement throughout the development stages, it shows the team which aspects of the product are affected by requirement changes. If the actual design and implementation takes place in a modular fashion, exhibiting low coupling and high cohesion (see the previously mentioned article), chances are that only those aspects of the product mentioned in that feature's matrix row need to be modified. As discussed earlier, flexibility and quick turnaround is a key point, especially for startup companies. The traceability matrix will facilitate such fast reaction times.

Startups have two particular problems with maintaining a traceability matrix. First, as already mentioned, not all requirements are known and not all are quantified. The requirement itself may thus have to be formulated in a very unspecific manner, making it difficult to fill the matrix cells with precise information. In that case the matrix

should still be maintained. Unquantified requirements should be marked and revisited as soon as more information becomes available. When that occurs the matrix will aid in identifying those parts of the product that need to be tested to see whether the modified requirement is still fulfilled.

The second startup difficulty with traceability matrices is the work required to maintain them. On complex products a very detailed matrix can fill hundreds if not thousands of pages. Clearly, a compromise needs to be made here. For starters, in many cases it doesn't have to be one monolithic matrix covering the whole product. Even though a complete matrix is always recommended to achieve product completeness, a company might choose to have each team maintain its own matrices. The requirements within one subcomponent or project are identified and listed in a matrix. Maintaining such a smaller matrix is naturally a much less resource-intensive task. On the downside, overall product requirements may suddenly be listed in the matrices of several product teams. In that case some communication overhead is required to keep these matrices in sync.

One might also choose to leave some of the requirements generally undefined and untraced. This is not at all ideal, but may be necessary due to a lack of resources. In that case the matrix should be limited to requirements that somehow have been deemed more critical than others. This method is risky since it again allows some requirements to be forgotten or not to be traceable if a change is required.

A traceability matrix is a powerful tool to ensure product completeness and a quick trace of a feature's "footprint" within the product. Even though startup companies are likely to compromise on some aspects of the matrix, it's highly recommended to keep it as complete as possible. The payoffs are significant.

Design, implementation and change control will be the topic of the next installment in this series. ☪

About the Author

Juergen Brendel is a software architect at Resonate Inc. He welcomes your feedback via e-mail at jbrendel@resonate.com.

jbrendel@resonate.com



SYS-CON Radio Host Robert Diamond with Bill Carson of ServerLogic

JAVA ON WEB RADIO

**Hear Live Interviews
with the Major
Technology Vendors
from the Java
Business Expo at
www.SYS-CON.com**

OMG Object Management Group

<http://www.omg.org>



Java Development Tools in Transition

Developers benefit from the symbiosis of static and dynamic modes of operation

by Paul Petersen

The tools available to the Java developer exhibit several unifying concepts, which provide a framework to explore the next transition in Java-development tools.

The first development tool that many Java developers use is the "javac" compiler that is bundled with the Java Development Kit (JDK). This compiler translates Java source code into the bytecodes that the Java Virtual Machine (JVM) executes. The "javac" compiler is not sophisticated, but it works and that's enough to get started.

Another tool that may be used is "java" or "jre", the runtime environments for the JVM. The JVM interprets the bytecodes generated by the compile to portably execute the compiled program.

At this point the developer has encountered two concepts. First, "javac" uses the notion of static analysis and static compilation to translate the Java language source code into the bytecodes stored in the class files. Second, the tools "java" and "jre" both use the concept of dynamic interpretation to drive the execution of the program.

To improve these tools we can add the concept of dynamic compilation. The latest releases of the JDK include a feature known as a Just-In-Time (JIT) compiler. A JIT allows the software developer to deliver the program in a portable format and to defer optimization until the bytecodes are dynamically compiled into machine instructions on the host computer.

The class files in the application are not all compiled - that would increase the delay between downloading the application and starting its execution. A JVM enhanced with a JIT can leave most of the program as bytecodes that are interpreted by the JVM; but when the JIT wants to optimize the execution of a section of the bytecodes, that section is dynamically compiled into machine instructions to increase the execution speed with a small compilation penalty.

The JIT model of dynamic compilation is great for clients that are executing different Java programs. But for servers which are always running the same Java program, it's occasionally better to compile the entire program into optimized

machine code for the target machine. This introduces the next concept, native code compilation. Native code compilers are like the "javac" tool we started with, except that they usually employ sophisticated semantic analysis, sometimes with the feedback of dynamically calculated information, to generate optimized machine instructions. Since these tools are designed to be occasionally executed, and the resulting program executed many times, these compilers can afford to do the extensive analysis necessary to optimize the Java program.

We have seen that as Java development and execution environments improve they usually incorporate increased usage of dynamic analysis techniques. This union of the compilation and execution models enables breakthrough tools which can take advantage of dynamic techniques in exciting new ways. Tools like profilers, feedback driven byte-code optimizers, memory leak detectors, or tools for source code coverage analysis are made possible when dynamic analysis is considered.

As an example of a novel application of dynamic analysis, consider the problems inherent when writing multithreaded Java programs. The Java runtime is inherently multithreaded, and Java supports explicitly multithreaded programming through the definition of a standard Thread class. But with this power comes a price. It's extremely difficult to prove that a multithreaded program is thread-safe, without making it inefficient by over-synchronizing all of the classes.

By using the concept of dynamic analysis, tools for multithreading defect analysis can be built to determine which objects are not correctly synchronized in the program and if the usage of the synchronized objects

could cause the program to deadlock. Safe, correct, multithreaded programming in Java is practical and easy with tools based on dynamic analysis that provide a safety net to find overlooked problems.

The compilation, analysis and execution tools for Java have evolved to encompass both the static and the dynamic modes of operation. The Java developer gets the advantage and benefit of this symbiosis in Java's advanced development environments. ♦

"It's extremely difficult to prove that a multithreaded program is thread-safe, without making it inefficient by over-synchronizing all of the classes."

About the Author

Paul Petersen is a lead developer for KAI's Assure. You can reach him at petersen@kai.com.



petersen@kai.com

Java JDK 1.2 Certification Insider

<http://www.certificationinsider.com>

KL Group Launches New JavaBeans for the Enterprise

(Washington, DC) – KL Group, Inc., has announced the release of JClass 3.5, a new version of its popular family of JClass JavaBeans providing powerful new databound components.

Automatic databinding is now built into many of the JClass components, and this release introduces an Enterprise Suite and two new JavaBeans, JClass HiGrid and JClass DataSource, that let

Java developers build complete database applications without writing a single line of code.

For more information, visit KL Group's Web site at www.klg.com, or call Lee Garrison at 416 594-1026, ext. 545, or e-mail him at lee@klg.com. ☛

Sun and IBM Introduce JavaOS

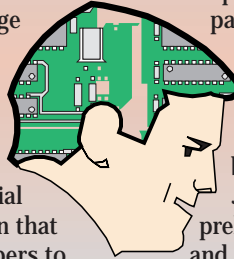
(Palo Alto, CA) – JavaOS for Business operating system software is available from Sun and IBM. This product provides an



economical way for companies to centrally manage business applications using Java technologies in network computing

MindQ Offers Educational Tutorial

(Herndon, VA) – MindQ Publishing, Inc., a Knowledge Universe Company, has announced that IBM will bundle its tutorial package, "Introduction to VisualAge for Java and Programming JavaBeans," with IBM's new release of VisualAge for Java 2.0. The tutorial features instruction that will enable developers to get the most out of VisualAge for Java's visual, team-centered, enterprise application development environment. It also includes a full tutorial on creating and using JavaBeans. MindQ's "Developer Train-



ing for Java" curriculum uses an interactive, multimedia training system designed to address diverse learning styles. It was developed by Java experts to meet the specific needs of beginners as well as the advanced users. "Essential Java Training" provides what a developer needs to learn the fundamentals of Java programming and prepare for Sun's Java certification. "Advanced Java Topics," designed for developers who want to become experts in Java, contains comprehensive code samples and reference material on topics like JavaBeans and Java APIs.

For more information, call MindQ at 800 646-3008 or visit their Web site at www.mindq.com. IBM's Web site is at www.ibm.com. ☛

environments

The two companies also announced related JavaOS for Business support programs for industry partners, including a full spectrum of software tools, testing facilities and educational assistance, to enable them to build network computing business solutions.

JavaOS software is specifically designed so that companies can centrally store and manage applications used to run their business (such as inventory management or insurance claims

processing) from servers on a network. The servers can be connected to network computers and other thin clients such as kiosks, ticket machines and remote terminals.

JavaOS for Business provides advantages over personal computer operating systems on networks because it enables businesses to manage the entire operating system, services and applications from a centralized server.

For more information, contact IBM at www.ibm.com/java/javaos or Sun at www.sun.com, or contact Datek at www.batavia.com. ☛

NetObjects Releases NetObjects BeanBuilder 1.0

(Redwood City, CA) – NetObjects, Inc., has announced that it will brand, market and distribute the next version of IBM's subsidiary Lotus Development Corporation's BeanMachine as NetObjects BeanBuilder 1.0. The product enables site builders and Web developers to rapidly assemble and deliver JavaBeans-based Web applications by working in a visual, point-and-click environment, without any programming.

The highlights of BeanBuilder's features are:

Schlumberger Announces Smart Card

(Austin, TX) – Schlumberger Smart Cards & Terminals has introduced its most powerful member in the Cyberflex family of smart cards. Cyberflex Open 16K which doubles the amount of memory available for application software.

Its new features also include a PC/SC interface and fully integrates an application processor; a smart card and a smart card manager.

Schlumberger has the only Web-based smart card support program, with a user discussion forum at www.cyberflex.slb.com.

For more information visit Schlumberger's Cyberflex Web site at www.cyberflex.slb.com or www.slb.com, or call Michele Bernhardt at 408 501-7145. ☛



Datek Delivers on Sun's JINI Promise

(Kuala Lumpur) – Datek has announced the first Enterprise Software Solution written to Sun's JINI standard for distributed computing. Datek is the

first Java software developer to partner with Sun for its newly launched JINI program. JINI allows computers and devices to have more intelligence in communicating and sharing with other computers across a network, including those that use different operating systems.

Madura, Datek's flagship

ERP and supply chain management (SCM) program, has been released with complete general ledger and systems administration modules. The program emphasizes project accounting, inventory tracking and job costing that allows corporations to know precisely their profit and cost of doing business for any customer or project.

Datek and Sun are ready to deliver on the notion of the "Internet appliance," making computers and networks as ubiquitous and easy to use as consumer electronics devices.

For more information contact Kamaralzaman Tambu at 603 456-2617 or by e-mail at tambu@pc.jaring.my. Or visit Sun's site at www.sun.com. ☛

JDJ Editors' Choice Awards

<http://www.javadevelopersjournal.com>

ObjectSpace Announces Voyager Pro 2.0

(Dallas, TX) – ObjectSpace has introduced Voyager Professional Edition, the next generation of widely adopted standards neutral, 100% Pure Java software development platform for distributed object computing.

VoyagerPro combines the power of mobile autonomous agents and remote method invocation with dynamic CORBA and RMI compatibility – providing in a single code base the first-ever seamless support for the most widely used distributed object models.

Developer's can remote-enable Java classes without modification and can write a single line of code to dynamically CORBA-enable Java objects at runtime without modification. The product provides fast and efficient delivery; rich messaging; a multilayered, scalable architecture; and mobile autonomous agents and

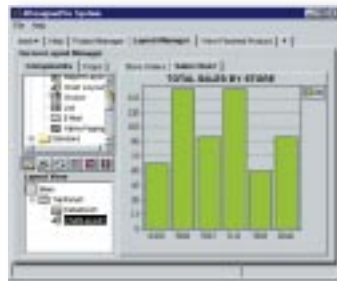
Dynamic Aggregation.

For more information, visit ObjectSpace's Web site at www.objectspace.com, or call 972 726-4100. ☛

BulletProof Announces JDesignerPro 3.0

(Los Gatos, CA) - BulletProof Corporation introduces a new visual server-side application builder, which includes new features such as the Method Explorer, that facilitate the development of Java modules for server deployment.

The Explorer graphically



shows the breakdown of Java classes. With existing tools developers waste time trying to determine the structure of a Java class file. BulletProof's

Inprise Joins Sun

(Denver, CO) – Inprise Corporation has entered an alliance with Sun Microsystems, Inc., to team Inprise's development technologies with the Sun Solaris™ operating environment. Corporations will be able to take advantage of Inprise's familiar and graphically appealing tools for building and running enterprise applications on the robust and scalable Solaris operating environment.

Inprise also released JBuilder 2 earlier this year. It allows corporations to use the latest 100% Pure Java technologies (including JDK 1.1.6 and JFC/Swing) to rapidly

create platform-independent business applications. The high-end version of the product, JBuilder 2 Client/Server Suite, includes support and integration for multiple JDKs, application deployment, Enterprise JavaBeans, Java Servlets, JFC/Swing components, CORBA and high-productivity code Wizards.

For more information, contact Inprise at www.inprise.com or visit Sun's Web site at www.sun.com. ☛



Simplicity for Java™ Introduced

(Summit, NJ) – Data Representations, Inc., Has announced version 1.1 of Simplicity for Java. The product is written completely in Java and it lets developers build Java applications and applets interactively. Simplicity presents the user with a working model of the actual application that they're creating. All changes to the code are immediately integrated into this working



model without the user needing to save and compile the changes. This dynamic execution reduces the traditional three-step code-compile-test software development process to a single step: design.

For more information, call Carl Sayres at 908 918-0396, fax 908 918-0397, email carl@datarepresentations.com, or visit their Web site at www.datarepresentations.com. ☛

new tool allows a simple graphical view, eliminating the need to manually browse through source code to understand which methods called which global variables, accessors and other methods.

JDesignerPro also includes the new SQL Wizard, which allows developers to add SQL statements to code with a few mouse clicks – resulting in flawless syntax.

For more information, visit BulletProof's Web site at www.bulletproof.com. ☛

CocoBase Enterprise 2.0 Available from Thought

(San Francisco, CA) – Thought Inc. has announced expanded features of database access framework with CocoBase Enterprise 2.0. To make it even easier to create an application that ties to your database and is high-performance and scalable, the CocoBase Enterprise Framework now includes:

- State-of-the-art object modeling tool Together/J, which can generate all of the database connection code automatically
- Polymorphism support for custom instantiation factories on select() and call()

methods for relational databases

- Automatic configurable shared server-side object caching to optimize and increase performance
- Revised documentation

For more information visit Thought's Web site at www.thoughtinc.com. ☛

Industry Leaders Bundle Zero G's InstallAnywhere Now!

(Irvine, CA) – ObjectShare will bundle Zero G's InstallAnywhere technology with upcoming releases of PARTS for Java products. In addition, Apple, IBM and Inprise will also bundle the product.

ObjectShare will also extend its current "Delivery Assistant" capabilities to send files to InstallAnywhere providing a total development-to-delivery solution.

InstallAnywhere Now! lists at \$149, and is available free to all registered PARTS for Java users. For more information, visit www.objectshare.com. ☛



Pervasive Software

<http://www.pervasive.com/sdk-jd>



Getting SQL Results from an Application Server:

It's not as easy as you think

THE GRIND

by Java George

"Getting SQL results sets isn't as simple as firing up any old JDBC driver that comes with the application server"

Java George is George Kassabgi, director of developer relations for Progress Software's Aptivity Product Unit. You can e-mail him at george@aptivity.com.



George@sys-con.com

The way some people hobble their application servers, you'd think they considered the server as pointless middleware. Most Java developers, however, want to unleash their application servers to do right by their applications.

"Doing right" in application-server terms means getting the stuff an application needs from various sources and delivering stuff from the application. When talking about data sources, getting and delivering corresponds to reading and writing data.

In general terms, application servers get and deliver stuff to three broad classes of systems: relational databases, from which it gets SQL results sets; legacy and packaged applications, such as SAP, Baan and PeopleSoft; and data emerging from distributed objects on the network, such as CORBA objects.

Over the next several columns we'll look at how application servers handle each of these sources. In this column we'll begin by looking at the first source, relational databases. This is probably the most common source of data accessed by application servers.

At first, getting SQL result sets from an application server seems pretty straightforward. Pass your request through the JDBC driver that comes with the application server and *presto!* You have your SQL results. In practice, however, it's not so simple.

For starters, many application servers, whether purchased or built by hand, are hardwired to proprietary database drivers for popular databases, which can hobble the application server. While a proprietary driver may be acceptable in the conventional client/server environment, it presents significant problems in the rapidly evolving Java world. Unlike the ODBC driver, which is middleware, the JDBC driver passes Java calls directly to the database. Database vendors and other parties continually bring out improved JDBC drivers for each database.

As a result, developers want to retain the flexibility to substitute the best JDBC driver for a given database. If the appli-

cation server is hardwired to a particular database driver, no matter how good it was when first built (and it's quite possible that it wasn't very good even then) you can be sure it won't be the best now. And it will really be an impediment compared to whatever is out there six months or a year from now.

What's the best JDBC driver? It depends. Some will deliver better performance; others will consume fewer system resources. Some will come from big-name vendors offering tons of support; others will come from hot little startups that can't even provide useful documentation. Java developers like having these choices.

But there's more to the SQL result-sets challenge than the JDBC driver. The application server must also do an effective job of providing a cache of result sets coming back from the SQL call. This isn't trivial, as anyone who has tried building such a cache can attest. The cache must be controllable by the development team and should provide for threads that fill the cache, if necessary. It's also important to provide governors that limit the total number of rows captured to prevent swamping the application server with an unexpectedly large result set.

Finally, the application server must allow for data binding of SQL data to client controls. While this sounds obvious, it's a sensitive issue that can adversely affect quality, performance and maintainability. The Java client must have easy access to the data. And, once again, an elegant and effective cache on the client side is key.

The bottom line: getting SQL result sets isn't as simple as firing up any old JDBC driver that comes with the application server. In fact, you must think beyond JDBC and consider cache and client control.

Now that we've taken care of relational data sources and SQL result sets, we can turn our attention, in a subsequent column, to legacy applications. Here we'll explore the problems that arise when the legacy data doesn't necessarily map well into JDBC type structures. Stay tuned. ☛

Voyager

<http://www.objectspace.com>

JProbe
by
KL Group

<http://www.klg.com/jprobe>