SYS-CON

Go Beyond AJAX with Adobe Flex 2.0 • See CD for free beta

Adobe

No. 1 *i*-Technology Magazine in the World

JDJ

JDJ.SYS-CON.COM VOL.11 ISSUE:3

— SEE PAGES 48-49 AND 60-61 FOR DETAILS —

REAL-WORLD AJAX ONE DAY SEMINAR

March 13, 2006
New York, NY

April 24, 2006
San Jose, CA

June 5-6, 2006
New York, NY

www.ajaxseminar.com

SOA WebServices Edge 10th International conference+expo Edge 06

2006 ENTERPRISE> OPENSOURCE CONFERENCE+EXPO

June 5-6, 2006
New York, NY

soaconference.sys-con.com

SUPER-CHARGE JSF AJAX DATA FETCH

RETAILERS PLEASE DISPLAY
UNTIL MAY 31, 2006

$5.99US $6.99CAN

04

0 09281 01751 6

PLUS...

Putting a Face on Web Services and SOA

Struts and JavaServer Faces

Don't Tell Me Cause It Hurts

THE WORLD'S LEADING *i*-TECHNOLOGY MAGAZINE JDJ.SYS-CON.COM

Java 911

Put out code fires with JProbe®

Tired of constantly fighting fires? Extinguish code issues and prevent future memory flare-ups with Quest Software's JProbe.

Discover the root cause of performance and memory issues in your Java applications. Easily test and fix your applications without any code changes. And even pinpoint performance problems down to the line of code. All with award-winning JProbe.

Stop the fire drills. Gain confidence in your code with JProbe.

Download a trial of JProbe at:

www.quest.com/911

Application Management | Database Management | Infrastructure Management

QUEST SOFTWARE®

Are 'Paternity' Suits the Latest Phenomenon in *i*-Technology?

Jeremy Geelan

Almost anyone who writes about Internet technologies, or *i*-Technology in shorthand, runs into a problem area from time to time concerning the issue of what in the *i*-Technology world was invented by whom?

Take Java, for example. No sooner does a commentator refer to James Gosling as "The Father of Java" than he gets 25 e-mails pointing out that the genesis of the programming language known originally as "Oak" was collective…and that, in any case, Gosling didn't name it, Sun's product management and marketing folks did. Or try calling Tim Bray "The Father of XML" and you'll get a similar e-mail, most likely from Tim himself, underlining the contributions of other eminent people in the XML community, such as Sun's Jon Bosak and Yuri Rubinski, a pre-eminent figure in the development and acceptance of SGML.

Recently I mentioned Adam Bosworth as the "Father" of Google's AJAX-based apps like Gmail and Google Maps, and Adam was quick to correct me. He would accept that he perhaps "godfathered" AJAX at Google, but denied outright paternity and hastened to remind me to give proper credit to progenitors like Paul Rademacher (Google Maps) and Paul Buchheit (Gmail).

Is this professional modesty a unique characteristic of software development? It's certainly refreshing: all the more kudos then to Bosworth, Bray, and top-flight professionals like them.

Last week, I was taken to task by Dave Carabetta in his blog for being one of the tens of thousands of people in the past 12 months to refer to Jesse James Garrett as the "Father" of AJAX. Dave didn't mention that in the full online announcement of SYS-CON's sell-out "Real-World AJAX" One-Day Seminar (www.ajaxseminar.com), this shorthand phrase is unpacked as "the man who first coined 'AJAX'" – instead he merely made the point, which is absolutely not in dispute, that Garrett is the one who coined the term AJAX "for an already-existing technology for asynchronously communicating with your server via JavaScript."

The issue of technology paternity is understandably a touchy one. Is David Heinemeier Hansson the "Father" of Ruby on Rails? Is Brian Behlendorf the "Father" of the Apache Web Server? Is Johann Gutenberg the "Father" of printing?

In the case of "AJAX," sometimes you might be forgiven for sensing a whiff of something almost akin to professional envy in the air, as if perhaps some developers resent that one little term, coined in one single essay (www.adaptivepath.com/publications/essays/archives/000385.php), should have had such a disproportionate effect in galvanizing public, worldwide interest in *an approach to technology* that predated the term itself.

But word-magic often has a life of its own, in technology as in other walks of life, and Jesse James Garrett can hardly be blamed for being given due credit for fathering the term that he fathered just because it took off so spectacularly. Anyone who believes in the power of asynchronously communicating with your server via JavaScript ought to be pleased, not perturbed.

Besides, unlike "Java," for example, in the case of AJAX there is no paternity suit. Jesse James Garrett most certainly is the Father of "AJAX" – making him one of the few people alive to have watched the global success of an acronym in his own lifetime.

Jeremy Geelan is group publisher of SYS-CON Media and is responsible for the development of new titles and technology portals for the firm. He regularly represents SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas.

jeremy@sys-con.com

JAVA DEVELOPER'S JOURNAL

Editorial Board
Java EE Editor: **Yakov Fain**
Desktop Java Editor: **Joe Winchester**
Eclipse Editor: **Bill Dudney**
Enterprise Editor: **Ajit Sagar**
Java ME Editor: **Michael Yuan**
Back Page Editor: **Jason Bell**
Contributing Editor: **Calvin Austin**
Contributing Editor: **Rick Hightower**
Contributing Editor: **Tilak Mitra**
Founding Editor: **Sean Rhody**

Production
Production Consultant: **Jim Morgan**
Associate Art Director: **Tami Lima**
Executive Editor: **Nancy Valentine**
Associate Editor: **Seta Papazian**
Research Editor: **Bahadir Karuv, PhD**

Writers in This Issue
Marius Constantin, Richard Conway, Yakov Fain, John Fallows, Jeremy Geelan, Jason Halla, Jonas Jacobi, Onno Kluyt, Michael O'Neill, Heman Robinson, Chris Schalk, Alex Smolen, Joe Winchester

To submit a proposal for an article, go to http://jdj.sys-con.com/main/proposal.htm

Subscriptions
For subscriptions and requests for bulk orders, please send your letters to Subscription Department:
888 303-5282
201 802-3012
subscribe@sys-con.com

Cover Price: $5.99/issue. Domestic: $69.99/yr. (12 Issues) Canada/Mexico: $99.99/yr. Overseas: $99.99/yr. (U.S. Banks or Money Orders) Back Issues: $10/ea. International $15/ea.

Editorial Offices
SYS-CON Media, 135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9638

Java Developer's Journal (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. Postmaster: Send address changes to: Java Developer's Journal, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
For List Rental Information:
Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

Newsstand Distribution Consultant
Brian J. Gregory/Gregory Associates/W.R.D.S.
732 607-9941, BJGAssociates@cs.com

Altova® MapForce® 2006 – The premier data integration and Web services implementation tool.

ALTOVA® mapforce® 2006

Streamline Web services

Hook up with MapForce® 2006, and build Web services without writing any code.

New in MapForce 2006:

- Drag-and-drop Web services implementation
- Advanced flat file parsing and integration
- Project-wide code generation
- Embedding in your applications via OLE / ActiveX

Altova® MapForce, the tool awarded for easy integration of XML, database, text, and EDI file formats, now also lets you implement Web services in a visual way. Simply drag connecting lines from information sources to targets and drop in data processing functions. MapForce converts data on-the-fly and auto-generates data mapping code in XSLT 1.0/2.0, XQuery, Java, C++, or C# for use in your data integration and Web services applications. Give your data direction!

Download MapForce® 2006 today: www.altova.com

Also available in the Altova XML Suite.

ALTOVA®

MARCH 2006 VOLUME:11 ISSUE:3

JDJ contents

JDJ (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. Postmaster: Send address changes to: JDJ, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

Yakov Fain
Enterprise Editor

Don't Tell Me Cause It Hurts

After one of my recent Java talks, a woman from the audience came to me and said, "I'm being displaced. But that's okay; the company gave me enough time for retraining. I've been working with Java , but would you recommend that I learn .NET?"

This lady deserves a lot of respect for at least two reasons:

1. She has maintained her positive attitude toward her current employer.
2. She is ready to start learning again.

So last week you used to be a programmer, but not anymore. What's next?

Life After a Pink Slip

Do not panic. Start collecting unemployment (in the U.S. it's about $400 a week for most programmers). Incorporated contractors are also entitled to unemployment benefits – if you can't sign a new contract within three months, just close your corporation and get some cash flow from the unemployment office.

If you haven't found a new job and your unemployment benefits are ending in a month, find any school that is approved by the Department of Labor and register for their longest course. Even if you won't learn anything useful there, your benefits may be extended for the time that you're in school.

Read this article I wrote in the July 2005 issue of *JDJ* about looking for a job: http://java.sys-con.com/read/111193.htm. Let me stress it again, your résumé should be customized for each job that you are applying to. The same facts about your work experience can be presented in multiple ways. As Nelson DeMille wrote in one of his novels, "He never lied. He could give 10 correct answers to the same question."

If, Else If, Else If

Even though this article is written for *JDJ*, I'd like to make it useful for non-Java programmers as well. Let's write some *if-statements* for a victim of a recent layoff.

```
if (urMainframeProgrammer){
  /* Consider learning WebSphere appli-
  cation server, messaging, and integra-
  tion tools.
    Get yourself IBM certified in these disci-
    plines. For example, WebSphereMQ
    administration is one of many career
    choices. Try selling your industry
    experience. */
} else if (urVBorPBProgrammer){
  /* Consider a database-related career.
     Improve your database skills, obtain
     all possible certificates from one
     of the major database vendors and
     present yourself in the job market as
     a database developer who also knows
     a front-end tool. Visual Basic people
     should master C# and ASP. Everyone
     should know XML. */
}else if (urJavaOrDotNetProgrammer){
  /* Stay where you are, but make your
     résumé stand out from the crowd
     by analyzing what the hottest skills
     are in your area using job-related
     Web sites. Perform multiple searches
     using different keywords that are
     close to your area of expertise, and
     meticulously write down the number
     of ads for each keyword. After finding
     the hottest skills, get yourself certi-
     fied in this skill by a recognizable
     software vendor like Sun Microsys-
     tems, IBM, Microsoft, Oracle, etc. Do
     not send out your résumé until
     you've received the certificates. */
}
```

–continued on page 55

"IT is a moving target and, if you can't keep up with it, you should find something more suitable"

Yakov Fain is a senior technical architect at BusinessEdge Solutions, a large consulting and integration firm. He authored the best selling book *The Java Tutorial for the Real World*, an e-book *Java Programming for Kids, Parents and Grandparents*,(smartdata-processing.com) and several chapters for *Java 2 Enterprise Edition 1.4 Bible*. He leads the Princeton Java Users Group.

yakovfain@sys-con.com

SYS-CON MEDIA

President and CEO:
Fuat Kircaali *fuat@sys-con.com*
Group Publisher:
Jeremy Geelan *jeremy@sys-con.com*

Advertising
Senior Vice President, Sales and Marketing:
Carmen Gonzalez *carmen@sys-con.com*
Vice President, Sales and Marketing:
Miles Silverman *miles@sys-con.com*
Advertising Sales Director:
Robyn Forma *robyn@sys-con.com*
Advertising Sales Manager:
Megan Mussa *megan@sys-con.com*
Associate Sales Manager:
Kerry Mealia *kerry@sys-con.com*

Editorial
Executive Editor:
Nancy Valentine *nancy@sys-con.com*
Associate Editor:
Seta Papazian *seta@sys-con.com*

Production
Lead Designer:
Tami Lima *tami@sys-con.com*
Art Director:
Alex Botero *alex@sys-con.com*
Associate Art Directors:
Abraham Addo *abraham@sys-con.com*
Louis F. Cuffari *louis@sys-con.com*
Assistant Art Director:
Andrea Boden *andrea@sys-con.com*

Web Services
Information Systems Consultant:
Robert Diamond *robert@sys-con.com*
Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Wayne Uffleman *wayne@sys-con.com*

Accounting
Financial Analyst:
Joan LaRose *joan@sys-con.com*
Accounts Payable:
Betty White *betty@sys-con.com*
Accounts Receivable:
Gail Naples *gailn@sys-con.com*

SYS-CON Events
President, SYS-CON Events:
Grisha Davida *grisha@sys-con.com*
National Sales Manager:
Jim Hanchrow *jimh@sys-con.com*

Customer Relations
Circulation Service Coordinator:
Edna Earle Russell *edna@sys-con.com*
JDJ Store Manager:
Brunilda Staropoli *bruni@sys-con.com*

Innovations by InterSystems

Messaging, plus…

Composite Application Development, plus…

Business Process Orchestration, plus…

Business Activity Monitoring

Integrate Your Enterprise. Ensemble Enables EAI, BAM, BPM, SOA, And All That Jazz.

Ensemble is the first product that enables fast integration and rapid development. This unique platform combines the capabilities of an integration server, data server, application server, and portal development software.

What makes this possible is an innovation by InterSystems: We developed Ensemble as a *single* technology stack, not a stitched-together suite of separate components. With a unified environment for integration, development, and management, Ensemble dramatically reduces time-to-solution.

We back these claims with this money-back guarantee: *For up to one year after you purchase Ensemble, if you are unhappy for any reason, we'll refund 100% of your license fee.** We are InterSystems, a global software company with a track record of innovation for more than 25 years.

InterSystems ENSEMBLE™

Request a FREE proof-of-concept project at **www.InterSystems.com/Ensemble12P**

* Read about our money-back guarantee at the web page shown above.

Which Way?

How to get men to ask for directions

by Marius Constantin

The University of Southern California has a student body of about 40,000 people and a fairly large main campus. Around registration and graduation times, it often happens that parents or future students stop me asking for directions to buildings where they want to go. There are several large maps installed at key points on the campus and map leaflets are available at the Information Booths and are distributed during the orientation sessions. Although these maps are quite detailed, a visitor can be put off by the large number of buildings and, if the distance to the destination is long, people can get lost.

After talking with my colleagues and several students, we came to the conclusion that it would be a good thing to offer our visitors an application they could use to get step-by-step directions within the campus.

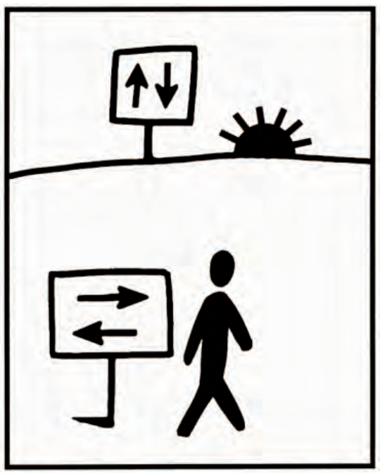
Thus the "Campus Directions" idea was born and within two months a pilot installation (with limited functionality and based on a somewhat older map) was available. We extended the idea and came up with an implementation that can cover any university campus or any relatively small area. The setup can be a fun two or three hour work and you get a program that shows step-by-step directions, with lengths and turns ("turn right," "turn left," "go straight") and a map image of the path trace similar to "yahoo maps" or "mapquest."

A sample installation can be seen at http://romania.usc.edu/directions. (To test drive, please send me an e-mail.) Code samples can be downloaded from http://romania.usc.edu/directions/code.

Additional help and a comprehensive list of setup steps can be found at http://romania.usc.edu/directions/help.

Marius Constantin is a programmer analyst III with the University of Southern California. His technical interests include portals, mobile devices and algorithms.

marius@usc.edu



Functionality

The application consists of two different modules:

- *PixelMapper*, a Swing tool used by administrators to set up (initialize) the maps
- The *Web Application* implemented using JSP, Struts, and custom JSP tag libraries and responsible for showing the directions. It also has "search" functionality that people can use to look up points

To initialize a map, an administrator uses the PixelMapper tool to load the map image (gif, jpeg, or png), and add points and links to it. After all the points have been added and linked, the tool calculates the shortest path and saves that data in a database. The reason why a database is being used is because finding the shortest path in a large set of points tends to take time and because the shortest path between two fixed points is a constant and recalculating a constant value over and over again would be a waste of processing cycles.

There are two types of points and three types of links. A **point** can be "named," which means it can be the starting point or end point in searches, or it can be a "link point" when it only serves as a connection between named points. In Figure 1, UCC (University Computing Center) and DEN (Norris Dentistry School) are named points whereas the ones along the streets are link points.

A **link** is the path element between two points (1 and 2). Links can be "bi-directional," "uni-directional 1-2," and "uni-directional 2-1." That means the path between points 1 and 2 can be traversed in either direction (1-2 and 2-1) or in only one direction (1-2 or 2-1.) That way the paths between two points A (e.g., UCC) and B (e.g., DEN) can be different if traversed from A to B rather than if traversed from B to A (as in the case when at least one path element along the way isn't bi-directional.)

After the map has been initialized and the shortest paths calculated, a request for the shortest path between UCC and DEN returns the results shown in Figures 2 and 3.

The application automatically crops the map to show only what's meaningful (in this case the rectangle containing the path). The administrator doesn't enter the lengths. Instead they are computed from the distance in pixels multiplied by a factor that depends on the scale of the map. So the more accurate the map, the more exact the directions.

Theoretical Considerations

To calculate the shortest paths I wrote (yet another) Java implementation of the simple algorithm that the Dutch mathematician Edsger Dijkstra created in 1959. The Internet abounds in samples and applets that demonstrate this algorithm. A piece of well-explained pseudo-code can be found on Wikipedia at http://en.wikipedia.org/wiki/Dijkstra's_algorithm.

I'll attempt to give a simplistic and visually driven description of the algorithm. Suppose we have the graph shown below; let's call the vertices (1,2,…5) **points** and the edges (1-2, 2-4, 4-5, etc.) **links**. Since there's no link between points 2 and 3, we'll say the distance is infinite (this is important in our implementation).

If we want to calculate the shortest paths from point 1 (the **source**) to all the others, let's start with a two-row table like the one shown in Figure 5. Let's fill the cells with the distances we know already (see Figure 5).

The distance from point 1 to itself is 0. We have the distances from 1 to 2 and 3. For the points that aren't linked directly to point 1 we write ∞ (for infinity.) Now let's pick the nearest point from 1; that's 2. Compare the distances from 1 to 3, 4, and 5 with the distances from 1 to 3, 4, and 5, but through the point 2. (e.g., d(1,4) > d(1,2) + d(2,4)) If the distance on the left is larger, replace it with the value on the right. In the example above the inequality is true since d(1,4) is infinite (no link). So d(1,4) will take the value d(1,2) + d(2,4) (see Figure 6).

Note that from now on, there's a path between 1 and 4 and that's 1-2-4 (in the general case, it may not be the shortest, though). Repeat for points 3 and 5. (To digress a moment, the shortest distance to point 3 is obvious, since there's a direct link but, in general, when the graph edges represent cost, it may be cheaper to travel 1-2-4-3 than 1-3 directly. However, that's not the case here.) At this point our table should look like Figure 7.

It already contains the shortest paths. But we have an overly zealous algorithm (greedy) so we'll continue by discarding point 2 and again find the closest point to 1 without considering 2. We repeat the steps above until all the points have been discarded. We have just found the shortest paths from point 1 to all the others. (To read the table, just move under the needed point. For example, d(p1,p4)=19.)

In the example above we've found the shortest paths on the graph when the starting point was point 1. To find the shortest paths from each point to all the others, the procedure would have to be enclosed in another loop that iterates through the set of points and fixes a different one as the source each time. However, in our case, not every point in the set has to be considered since the (unnamed) "link points" can't act as the start or end of a path.

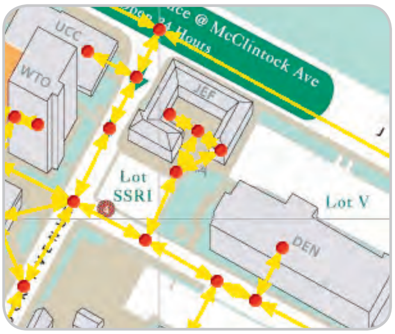


Figure 1 A crop of the PixelMapper screenshot with points and links defined

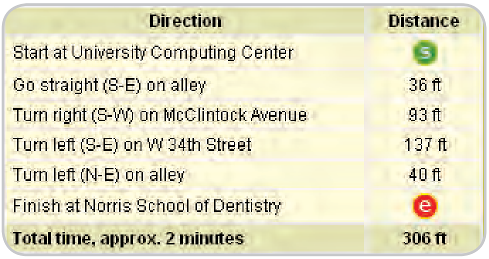
Technology and Implementation

The application was written entirely in Java. AWT 2D (the two-dimensional functions of the Abstract Windowing Toolkit) and JAI (Java Advanced Imaging extensions) were used for image processing (like tiling, painting on the map, and rendering.) The administration tool (PixelMapper) was written using Swing components and the Web interface uses JSP/Struts and custom JSP tag libraries. The application has been tested with MySQL 4.1, HyperSonicDB 1.7.2, and Oracle 9i and 10g release 1.

Now, before the application can be used, the map has to be "initialized" using the PixelMapper tool (the significant points added, linked, and the shortest distances calculated and saved into the database). Our graph will actually consist of the points on the map (see Figure 1) and the links between them. The activity diagram is shown in Figure 8.

There are few basic entities that the application uses, some of which are shared by both the PixelMapper tool and the Web application. These types are:

- *Point*, which represents a point on the map. It has attributes like the coordinates (x, y) on the map, name ("Norris Dental School"), etc. A link point has null value for name. The *PointList* is a vector of point instances
- *OrientedLink*, which represents a path element; it's the link between two points. It holds references to the start and end *point* instances and attributes like *name* ("McClintock Ave.") and *length*. *OrientedLinkList* is a vector of *OrientedLink* instances.
- *PathElement*, which contains all the details of a segment of the current path. It has references to the two points and contains the distance and the turn. It's initialized with data from the database. (A path is a list of *PathElements*.)
- *Dijkstra*, which is the implementation of the shortest path algorithm.
- *Processor*, which uses data from the database to create the path image.
- *ImageCache*, which is an in-memory placeholder for *ImageCacheToken* instances. The *ImageCacheToken* holds the bytes of the generated path image plus the step-by-step directions.
- *CachePersistence*, which is the interface to be implemented by the custom persistences. (*FileSystemPersistence*, which



Direction	Distance
Start at University Computing Center	s
Go straight (S-E) on alley	36 ft
Turn right (S-W) on McClintock Avenue	93 ft
Turn left (S-E) on W 34th Street	137 ft
Turn left (N-E) on alley	40 ft
Finish at Norris School of Dentistry	e
Total time, approx. 2 minutes	**306 ft**

Figure 2 Step-by-step directions

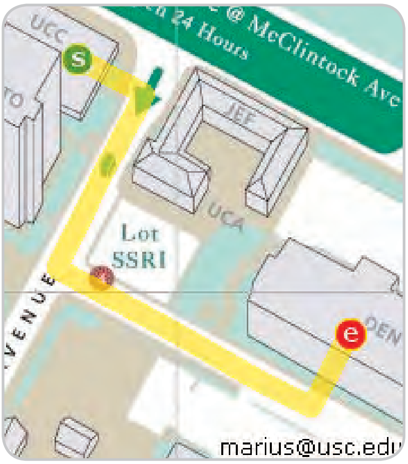


Figure 3 The path trace plotted over the map image and cropped

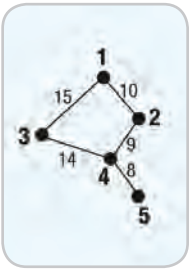
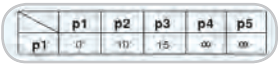
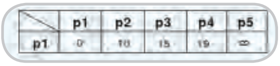


Figure 4 The graph used in the example



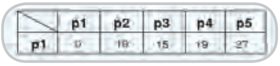
	p1	p2	p3	p4	p5
p1	0	10	15	∞	∞

Figure 5



	p1	p2	p3	p4	p5
p1	0	10	15	19	∞

Figure 6



	p1	p2	p3	p4	p5
p1	0	10	15	19	27

Figure 7

serializes the content of the cache to the file system, is provided.)

Figure 9 shows the relationships between the main types (class diagram).

The upper half of the image depicts the main classes used by the PixelMapper tool at map initialization, whereas the lower part shows the "model" layer of the Web application. The classes in the gray rectangle are shared by the two parts.

The implementation of the algorithm *Dijkstra::calculateShortestPaths* produces two double-dimensional arrays (two square matrices) – a *pointMatrix* and a *distanceMatrix*. The first array is used to find the shortest path between any two given points in the initial PointList, and the second gives the corresponding distances.

Figure 10 shows the *pointMatrix* partially populated, but containing enough data to find all the shortest paths when points 1 or 4 are used as the start points (refer to Figure 4.)

For example, if we wanted to find the shortest path between points 1 and 5, we'd fix the first row in the matrix (row 0), which corresponds to point 1 when used as starting point, and start from the fifth column (0,4), going left (the matrix is always populated with the points in reverse order so we start with point 5 and go back until we reach point 1).

At this point our path contains only one point (point 5). In the (0,4) cell we find a reference to point 4 (follow the arrows). We add point 4 to the path (point 5 -> point 4). Now we check the cell corresponding to point 4 (0,3) and find a reference to point 2. We add point 2 to the path, which now looks like point 5 -> point 4 -> point 2. Again, we read the content of the cell that corresponds to point 2 (0,1) and find that it references point 1. We add point 1 to the list, which now looks like (point 5 -> point 4 -> point 2 -> point 1) and stop, since point 1 was the end of the desired path.

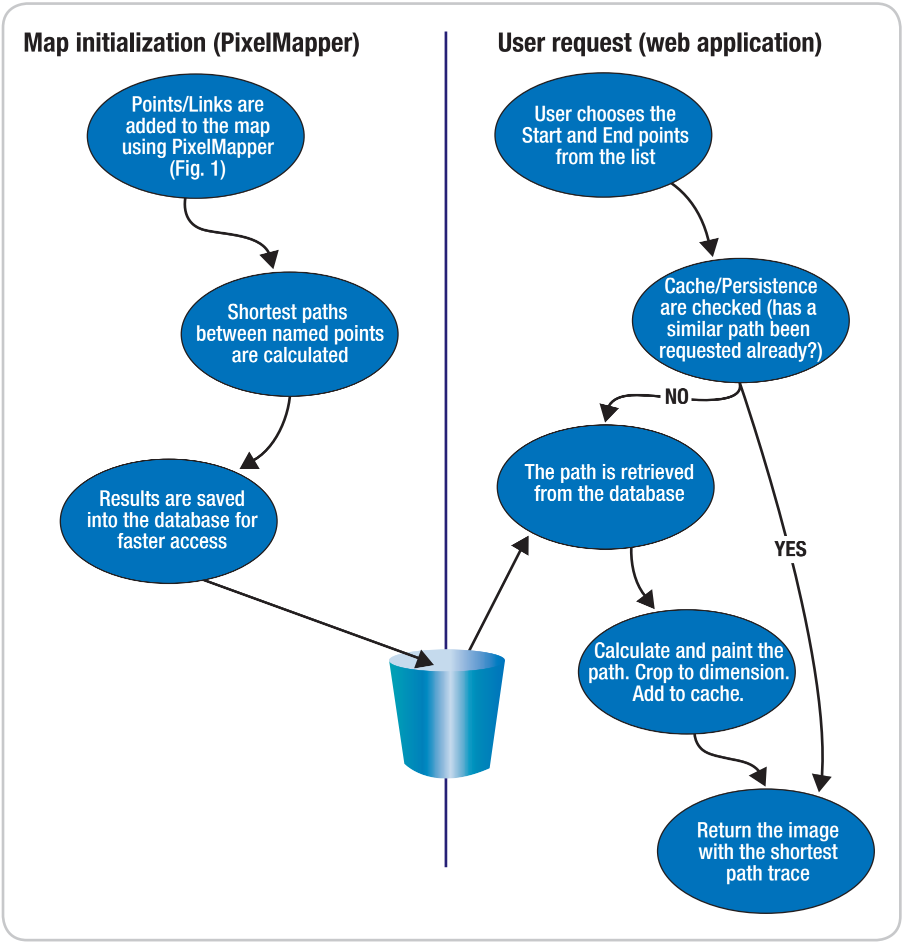
Remember the path is in reverse order so we'll have to traverse it backwards and return 1 – 2 – 4 – 5 (see Figure 10).

The code in Listing 1 is an extract from *Dijkstra::calculateShortestPaths* function (see *Dijkstra* in the code section) and it shows the Greedy loop that finds the shortest distances from one fixed point to all the others. It's enclosed in an outer loop that fixes another named point (startVertex) in each iteration. It's slightly modified from the original version and edited for clarity (comments and variable declaration in the loops. *distanceMatrix* is a *long[][]* and *pointMatrix* is a *Point[][]*).

Dijkstra::retrievePath is the function that returns the paths given any two named points in the initial set. It uses both *pointMatrix* and *distanceMatrix* and stops when the distance between the two current adjacent points is 0 (meaning the two points are, in fact, references to the same point like cell (0,0) in the table above).

After calculating the shortest paths, the results are saved in the database. There are four tables to hold the data:

- POINT, which contains all the defined points. The PixelMapper tool uses this table to load the points on the map at initialization. It's also used by the "search" function.
- LINK, which holds all the defined links. Like the table above, the PixelMapper tool uses this one to paint the links created in a previous session.
- SHORTEST_PATH, which is populated after running *Dijkstra.calculateShortestPaths*. It only contains a "path id" and the ids of the start and end points.
- SHORTEST_PATH_ELEMENT, the table that is also populated after the algorithm runs. It holds all the path elements, that is, the links between the consecutive points on a path. It has lengths, turns, cardinal points, etc. From this table come the step-by-step directions seen in Figure 3.



Map initialization (PixelMapper)
User request (web application)
Points/Links are added to the map using PixelMapper (Fig. 1)
Shortest paths between named points are calculated
Results are saved into the database for faster access
User chooses the Start and End points from the list
Cache/Persistence are checked (has a similar path been requested already?)
NO
YES
The path is retrieved from the database
Calculate and paint the path. Crop to dimension. Add to cache.
Return the image with the shortest path trace

Figure 8 The activity diagram for the map initialization and user request

get WebRenderer™

Java Browser SDK

Desktop Edition

Embed WebRenderer into your Java client side application interface to display rich web content.

- ✓ Swing and AWT support
- ✓ Extensive DOM
- ✓ Plugin support
- ✓ Powerful API
- ✓ HTTP, SSL, TLS
- ✓ JavaScript
- ✓ Complete Event model

Server Edition

Automate Server Side web content reporting, printing and processing with WebRenderer Server Edition.

- ✓ Headless server optimized
- ✓ J2EE support
- ✓ No AWT or Swing dependencies
- ✓ Complete DOM support
- ✓ HTML to thumbnail image
- ✓ Print support
- ✓ Standards compliant rendering

To download a free 30 day trial visit www.webrenderer.com

JadeLiquid™ Software

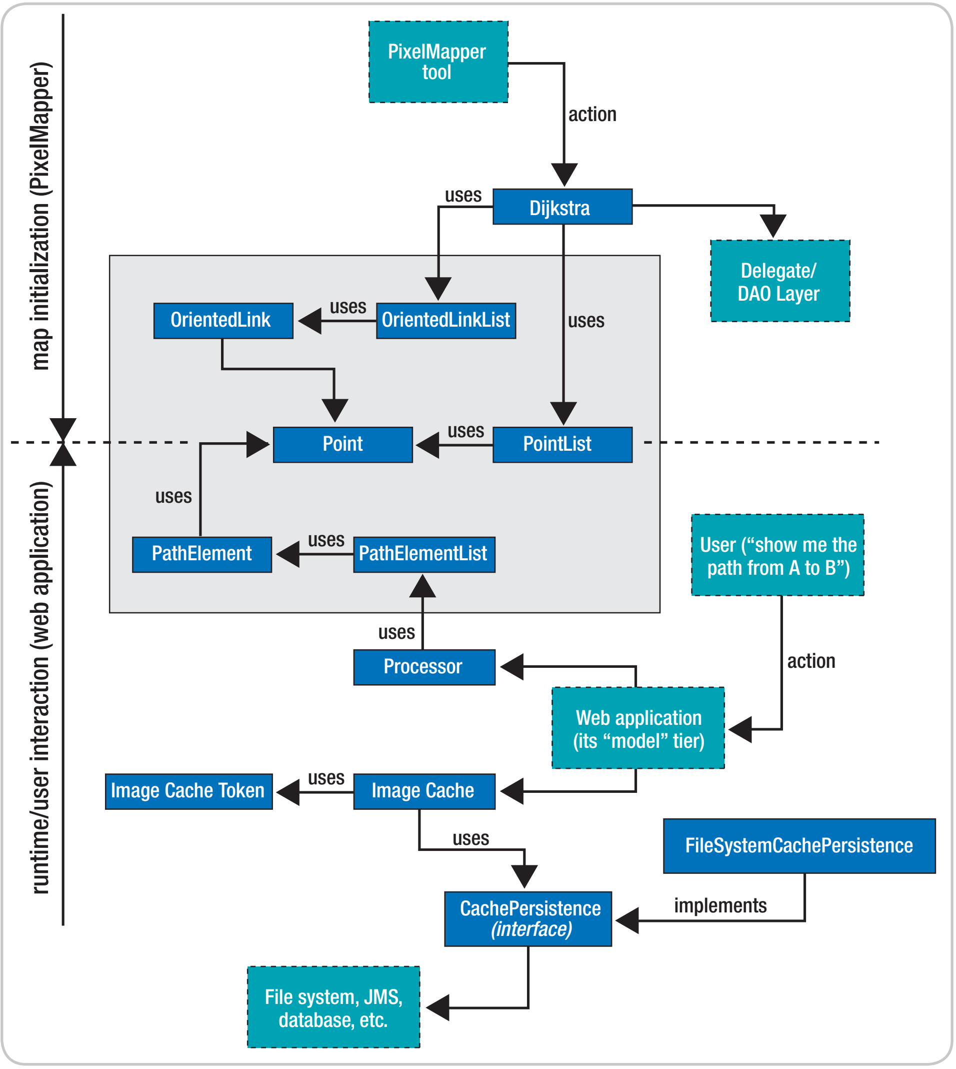
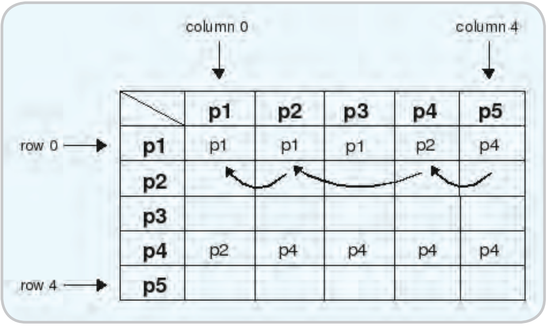

map initialization (PixelMapper)
runtime/user interaction (web application)
PixelMapper tool
action
uses
Dijkstra
Delegate/ DAO Layer
OrientedLink
uses
OrientedLinkList
uses
Point
uses
PointList
uses
PathElement
uses
PathElementList
User ("show me the path from A to B")
uses
Processor
action
Web application (its "model" tier)
Image Cache Token
uses
Image Cache
uses
FileSystemCachePersistence
CachePersistence (interface)
implements
File system, JMS, database, etc.

Figure 9 The class diagram for the two application modules


column 0
column 4
row 0
row 4

	p1	p2	p3	p4	p5
p1	p1	p1	p1	p2	p4
p2					
p3					
p4	p2	p4	p4	p4	p4
p5					

Figure 10 The pointMatrix table partially filled

An interesting fact to note is that one of the hard-to-do parts in the implementation process (well, harder than expected, anyway) was to come up with a way to calculate the turns correctly ("make a right" or "keep going straight").

I had to consider every two adjacent links (or path elements) as vectors in the plane, take the cross-product and use the right-hand rule to find whether the result pointed up or down and, as a function of that, see if the turn was to the right or left. For collinear vectors, I used their direction (if same direction then the "turn" would be a "go straight" and if the opposite direction the turn would be a "U-turn"). (See *PathUtil::getTurn.*)

Strictly speaking, oriented links would have been unnecessary if the application only gave "walking directions." But it can be set up to show driving directions too. So having "one-way" path elements begins to make sense.

The Web application has been implemented around the MVC (model-view-controller) paradigm using Struts. When a user asks for directions, the database is queried, and the data access layer returns the path between the start and the end points with all the details required.

To further improve the response time, the application implements a caching mechanism. After a path is generated as the result of a user request, an *ImageCacheToken* is created and added to the in-memory cache. The *ImageCacheToken* contains the path image as a stream of bytes (see Figure 2) and the list with the step-by-step directions (see Figure 3). Retrieving from the cache is almost instantaneous.

The coordinates of all the points involved are passed to the *Processor::tracePath*. This function draws on the map, creates the step-by-step directions, and feeds all these to a new *ImageCacheToken* instance.

While points are added to the map(*Processor::tracePath*), the "visible area" is permanently enlarged to accommodate all the points that comprise that path. A "band" (of configurable size) is then added around the cropped image. (See the *Processor* source in the code section.)

In fact, the *ImageCacheToken* contains two images (as byte arrays.) If the start and end points are far apart, the resulting image can be large. A scaled-down image is shown first with a "show full image" button painted on it (in the lower-left corner). The maximum size of the image before it gets scaled down is a configurable parameter and can be set by the administrator.

After the *ImageCacheToken* is instantiated and populated with data for a specific path, it's added to the *ImageCache* instance, so next time the system gets a request for the same directions, the same *ImageCacheToken* is returned without querying the database or processing the image map. If the age of the ImageCacheToken goes over a configurable limit, the instance becomes available for persistence.

Once an ImageCacheToken is obtained (either from cache, persistence, or newly created), the image is sent to the Web application as a byte array.

However, since the application can potentially be used by tens of thousands of people, there's a good chance that the servlet container will throw an OutOfMemoryException

IBM RATIONAL PRESENTS

YOU ★ VS ★ THE INCREDIBLE SHRINKING DEADLINE

MAIN ATTRACTIONS

KNOCKOUT INNOVATION | INTEGRATED DEVELOPMENT TOOLS SUPPORTING ASSET-BASED DEVELOPMENT ★ BASED ON ECLIPSE™ ★ RUNS ACROSS MULTIPLE PLATFORMS INCLUDING LINUX®

POWER TO CREATE BETTER SOFTWARE FASTER

IBM MIDDLEWARE. POWERFUL. PROVEN. FIGHT BACK AT WWW.IBM.COM/MIDDLEWARE/TOOLS

AND DOWNLOAD TRIAL VERSIONS OF RATIONAL SOFTWARE MODELER & RATIONAL SOFTWARE ARCHITECT

"The hard part (well, harder than expected, anyway) was to come up with a way to calculate the turns correctly"

after a large number of path images have been generated and cached in-memory. To prevent that from happening, a persistence layer has been added. A thread periodically checks the in-memory cache and moves the "expired" *ImageCacheToken* instances to the persistence implementation. An *ImageCacheToken* has a configurable attribute (*age*) that tells the persistence thread whether it can be persisted or not.

Once it's been determined that an *ImageCacheToken* instance is available for persistence, an *ImageCacheTokenSerializable* type is instantiated and data from the original *ImageCacheToken* is copied over. The only difference between the two types is that *ImageCacheTokenSerializable* holds the image bytes as Byte64 encoded strings, which are serializable (*java.io.ByteArrayOutputStream* type is not serializable). Then the new instance is passed to the *CachePersistence* by calling *addToPersistence*.

Out-of-the-box, the application comes with a file system implementation of the persistence layer (*FileSystemPersistence*) that serializes the *ImageCacheToken* instances and writes them off to the file system (using the *java.io.ObjectOutputStream::writeObject*). A set of interfaces and classes has been exposed that can be used to produce custom implementations for the cache persistence. JMS or DB cache persistence implementations shouldn't be too hard to write.

Conclusion

The application could be used by universities or cities to offer visitors directions to their points of interest. Cities may also want to make their downtown maps more interactive and give people an easy way to find their way to buildings or attractions.

An application such as this could be installed in kiosks equipped with touch-screens, where people could search for points of interest and print out step-by-step directions. It could be enhanced to "link" maps together and show directions between different floors of a building or between different campuses. Another improvement could be showing "points of interest" along a path such as bookstores, restaurants, and coffee shops.

References

1. Knuth, Donald, *The Art of Computer Programming (vol 1, 3rd edition)*. Addison-Wesley.
2. Andonie R., Garbacea I., *Fundamental Algorithms (a C++ perspective)*. Libris, 1995.
3. *Wikipedia*, http://www.wikipedia.com

Listing 1

```
 /* (THE GREEDY LOOP)
  Obtaining a list of all links from the startVertex to all the other
  vertices. For the non-adjacent  vertices (virtual links - where a
  path is made up of multiple links) the initial value is INFINITY.
  The collection contains "n-1" OrientedLink instances, where "n" is
  the total number of vertices (candidates)*/
  OrientedLinkList linksFromStartVertex =
  orientedLinkList.getAllLinksFromPoint(startVertex,tempCandidates);
  while(linksFromStartVertex.getSize() > 0) {
   /* Finding the shortest distance from the startVertex, consi-
     dering all the points left in the linksfromStartVertex
     (this list reduces by one after every loop.)
     The secondStartVertex is the point with the shortest
     distance to the startVertex in the current loop (current
     content of the allLinksFromStartVertex list) */
  OrientedLink shortestLink = linksFromStartVertex.
                              removeShortestLink();
  Point secondStartVertex = shortestLink.getEnd();
  tempCandidates.removePoint(secondStartVertex);
  long linkDistance = shortestLink.getLength();


 /* Calculating the distances from the secondStartVertex to
     all the other points that are still in the tempCandidates
     list (less startVertex and itself) */
  OrientedLinkList linksFromSecondStartVertex =
  orientedLinkList.getAllLinksFromPoint
                       (secondStartVertex,tempCandidates);
  while(linksFromSecondStartVertex.getSize() > 0) {
      Point secondEndVertex = linksFromSecondStartVertex.
                                   removeShortestLink().getEnd();
      int secondEndVertexIndex = candidates.
                                   getPointIndex(secondEndVertex);

    /* If the total distance "dist" (startVertex ->
        secondStartVertex -> current point (secondEndVertex)) is
        smaller than the original distance (startVertex ->
        current point) then "dist" is kept. Also the current point
        gets saved into the pointMatrix */
     long secondLinkDistance = orientedLinkList.
                                   calculateLengthBetweenPoints
                                  (secondStartVertex, secondEndVertex);
     if(secondLinkDistance != PathConstants.INFINITY &&
            (linkDistance + secondLinkDistance <
                                distanceMatrix[i][secondEndVertexIndex])) {
          distanceMatrix[i][secondEndVertexIndex] = linkDistance +
                                                        secondLinkDistance;
          pointMatrix[i][secondEndVertexIndex] = secondStartVertex;


    /* modifying the length so that it will be picked up in the
      correct order in the outer while loop */
    linksFromStartVertex.getLink(startVertex,secondEndVertex).
    setLength(linkDistance + secondLinkDistance);
   }
  }
 }
```

IT'S IN THERE SOMEWHERE

MAKE ANSWERS TO PERFORMANCE PROBLEMS COME TO YOU.

OPNET Panorama offers powerful analytics for rapid troubleshooting of complex J2EE/.NET applications. Panorama quickly identifies how application, web, and database servers are impacting end-to-end performance. With Panorama, you can pinpoint the source of a problem, so time and money aren't spent in the wrong places.

The most successful organizations in the world rely on OPNET's advanced analytics for networks, servers, and applications.

OPNET Panorama™
Real-Time Application Analytics

www.opnet.com/pinpoint

OPNET®
Making Networks and Applications Perform™

OPNET Technologies, Inc. 7255 Woodmont Avenue, Bethesda, Maryland 20814 phone: (240) 497-3000 • e-mail: info@opnet.com • NASDAQ: OPNT

Super-Charge JSF AJAX Data Fetch

Harnessing managed beans

by Jonas Jacobi and John Fallows

In our last article – "JSF and AJAX" *(JDJ*, Vol. 11, issue 1) – we discussed how JavaServer Faces component writers can take advantage of the new Weblets Open Source project (http://weblets.dev.java.net) to serve resources such as JavaScript libraries, icons, and CSS files directly from a Java Archive (JAR) without impacting the application developer.

In this article we'll address the need to fetch data using AJAX with JavaServer Faces (JSF) components. The most common use cases for fetching data with AJAX are to populate dropdown lists and add type-ahead functionality in text fields. In contrast to using AJAX postbacks for events, fetching data shouldn't affect the surrounding components on the page. And if fetching data isn't affecting other parts of the DOM tree, then you don't have to go through the full JSF lifecycle just to get the data, right?

This article introduces for the first time a new Open Source project called Mabon hosted on the Java.net Web site (http://mabon.dev.java.net). Mabon stands for Managed Bean Object Notation and its goal is to provide component writers of AJAX-enabled JSF components to access JSF managed beans outside the scope of the standard JSF lifecycle by using a JSON-syntax communication channel.

In essence, Mabon provides application developers with a standard and easy way to provide data to AJAX-enabled components using the managed bean facility provided by the JSF specification.

Fetching Data with AJAX

Fetching data the conventional way versus using AJAX follows the same basic concept – it shouldn't have the side effect of changing the state of surrounding components.
Figure 1 shows an AJAX sequence diagram using the HTTP GET method. The W3C recommends that you use the HTTP GET method to fetch data when there are no side effects requested by the user (for example, Google Suggest).

Different JSF AJAX Approaches

If you get no side effects, then there's no change to the JSF component hierarchy; so there's no need to go through the JSF lifecycle. But, if you want to reuse a managed bean method, the easiest way to get to it is via the JSF MethodBinding facility. Three solutions exist to support this – adding functionality to the Renderer, using a PhaseListener, and providing a new JSF Lifecycle.

The Renderer Approach

The Renderer approach adds functionality to the Renderer to detect the AJAX request. The JSF default lifecycle first restores the component hierarchy during the Restore View phase and the Renderer takes control during the Apply Request Values phase. After the AJAX request has been processed, the Renderer calls responseComplete() on the FacesContext to terminate processing of remaining phases of the Lifecycle. On the surface this may seem like the preferred approach, but it has some severe drawbacks.

A component hierarchy is required, which can incur additional overhead for each request, especially when client-side state saving is used. Calling the responseComplete() method will take effect only after this phase is done processing. The Apply Request Values phase calls the decode() method on all Renderers in the view, which can cause undesired side effects that are out of your control, such as a <h:commandButton> set to immediate="true" by the application developer. This causes the application logic to be called before the Apply Request Values phase is complete.

This approach also usually requires HTTP POST to send the state string back to the server.

The PhaseListener Approach

The PhaseListener approach adds a PhaseListener (PhaseId.RESTORE_VIEW) that short-circuits the Lifecycle and does all the processing in the PhaseListener itself. When it's done, it calls responseComplete() on the FacesContext.

For this approach to work, it has to render a reference containing information about the managed bean used in

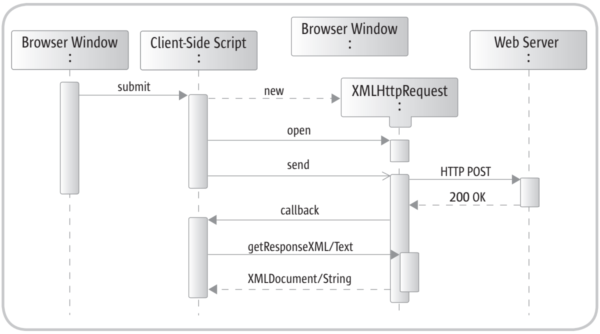


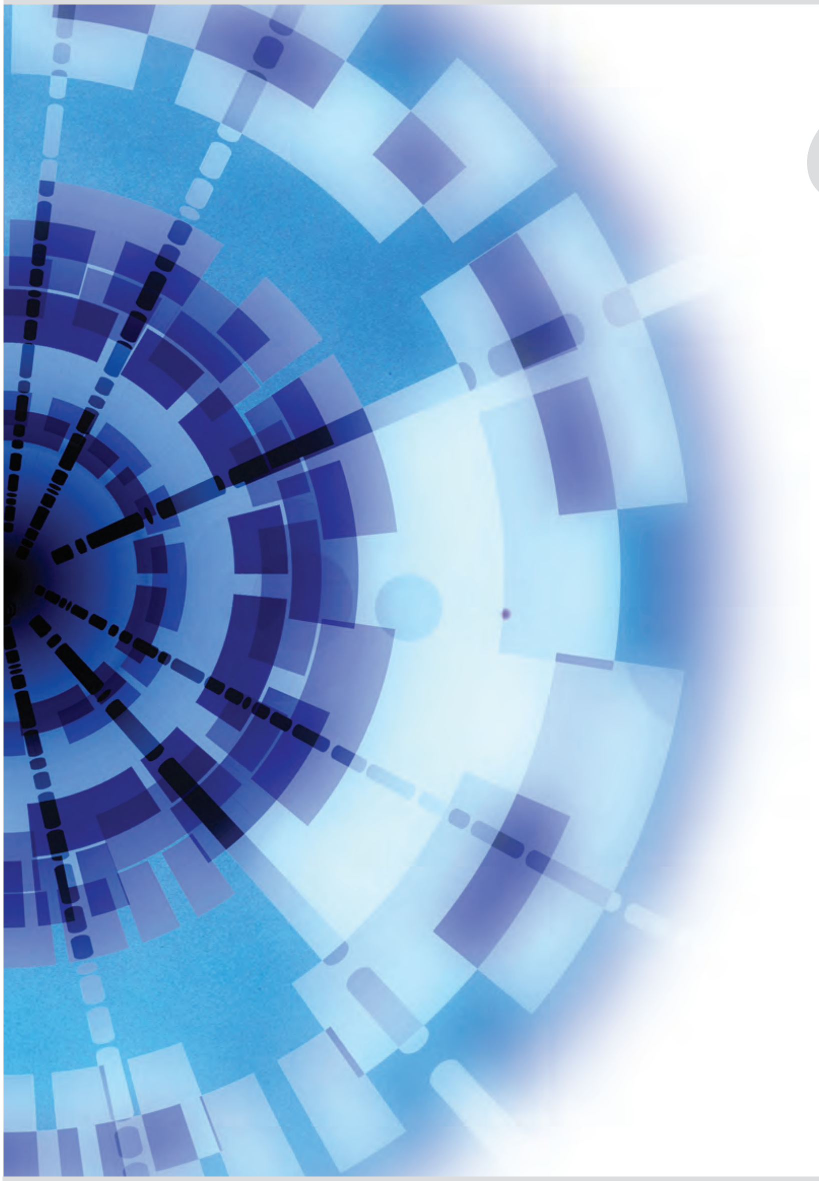
Figure 1 Sequence diagram of an XMLHttpRequest using the HTTP GET method

Jonas Jacobi is a principal product manager and evangelist for Oracle's Java/J2EE tool offering, JDeveloper, and over the past three years has been responsible for JavaServer Faces, Oracle ADF Faces, and Oracle ADF Faces Rich Client development features within Oracle JDeveloper. Jonas has been in the software business for 15 years. Prior to joining Oracle, he worked at several software companies in Europe, covering many roles including support, consulting, development, and project team leadership.

jonas.jacobi@oracle.com

Best AJAX Toolkit* TIBCO General Interface

Build 100% Pure Browser Rich Internet Apps with TIBCO General Interface™



"TIBCO General Interface is a friendly, capable toolkit for building sophisticated JavaScript Web applications that run in a browser."

Download today
at **http://www.tibco.com/mk/gi**

TIBCO®
The Power of Now®

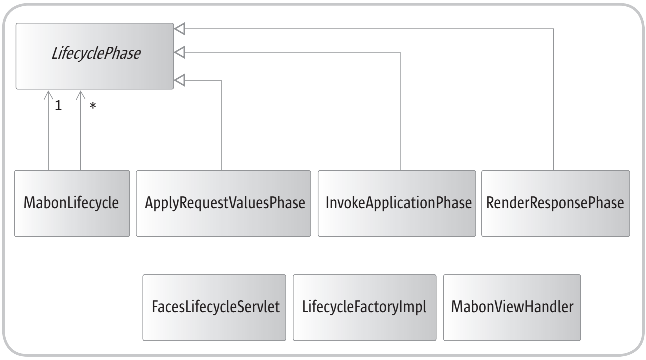


Figure 2 Class diagram of Mabon

the initial request. The PhaseListener uses this information during postback to create a MethodBinding that can then be used to invoke a method on the managed bean and return data to the client. Since no component hierarchy is created, and thus no Renderers, there's no risk that command components with immediate set to true will cause any side effects.

But this approach has one issue; there's no way to prevent application developers from attaching additional PhaseListeners at the same phase, which can cause undesirable side effects. You also have no way of knowing in which order these PhaseListeners will be executed.

The Lifecycle Approach

This Lifecycle approach adds a new Lifecycle that's mapped to an AJAX request and contains only the lifecycle phases needed to process the request, invokes the application logic defined by a MethodBinding, and renders the response. This eliminates the overhead of creating and restoring the component tree, and so no Renderers are required. You also won't encounter any issues with immediate="true".

Another positive side effect of using a custom Lifecycle is that any PhaseListener added by the application developer will have no impact on this solution; application developers can even add PhaseListeners to this custom Lifecycle. However, if a custom PhaseListener is used to put additional managed beans on the request, you can run into issues, unless they're registered for the custom Lifecycle as well.

Select a JSF AJAX Approach

Here we use the Lifecycle approach, since it has no application logic side effects and low overhead. It's here that the Mabon Open Source project can help you focus on the design of your AJAX-enabled component.

Let us explain in a little about what Mabon is and what it can provide component writers interested in AJAX data fetch.

What Is Mabon?

Mabon offers a convenient way to hook in a specially designed lifecycle that's ideal for AJAX-enabled components that have to fetch data directly from a backing bean, without the overhead of a full JSF lifecycle. It also provides a Mabon protocol – mabon:/ – used to reference the backing bean and a JavaScript convenience function used to send the target URL and any arguments needed and then get data asynchronously from the managed bean.

Mabon and JSON

As you know, the XMLHttpRequest provides two response types — responseText and responseXML – that can be used to fetch data. The question to ask is; when should I use which? Answers to this question can differ depending on whom you ask, but we can recommend one guideline. Ask yourself if you control the syntax of the response.

The responseXML type returns a complete DOM object (which gives you ample ways of walking the DOM tree), letting you find the information needed and apply changes to the current document. This is useful when your component will impact surrounding elements and you don't control the response (for example, when you're communicating with a Web Service).

The MabonLifecycle Class

The MabonLifecycle consists of three phases – ApplyRequestValuesPhase, InvokeApplicationPhase, and RenderResponsePhase. The MabonLifecycle is responsible for executing these three phases. It's also responsible for handling any PhaseListeners attached to the MabonLifecycle.

The LifecyclePhase Class

The Mabon LifecyclePhase is the base class for all lifecycle phases.

- ***The ApplyRequestValuesPhase, InvokeApplicationPhase, and RenderResponsePhase Classes –*** Since you're only fetching data and not modifying the component hierarchy or the underlying model in any way, you don't need to include the Restore View, Process Validations, and Update Model phases. The Mabon phases are performing similar operations to the default lifecycle equivalents, such as decoding an incoming request, invoking application logic, and rendering the response.
- ***The FacesLifecycleServlet Class –*** This is a reusable servlet that will initialize the FacesContextFactory and look up the MabonLifecycle in its first request. It will create the FacesContext then invoke the three lifecycle phases that are part of the MabonLifecycle. The servlet mapping defined by the Web application will direct Mabon requests to this FacesLifecycleServlet.
- ***The LifecycleFactoryImpl Class –*** The only purpose of this class is to add a second lifecycle – the MabonLifecycle.
- ***The MabonViewHandler Class –*** During the initial rendering, a custom Renderer has to provide a path to the backing bean that can be intercepted by the FacesLifecycleServlet and used during InvokeApplicationPhase to call the referenced backing bean. By using the Mabon protocol, a component author can get a unique path from the MabonViewHandler that can be rendered to the client. If the component writer passes the string shown in Code Sample 1 with the path argument of the ViewHandler.getResourceURL() method, the MabonViewHandler will return the string shown in Code Sample 2 that can be written to the client.

John Fallows is a consulting member of technical staff for server technologies at Oracle Corporation, and has been working in distributed systems for over a decade. During the past five years, he has focused on designing, developing, and evolving Oracle ADF Faces, and is now lead developer for Oracle ADF Faces Rich Client.

john.r.fallows@gmail.com

Fiorano SOA™ 2006

The Quickest Path to an SOA

Fiorano provides the industry's first framework enabling developers to rapidly create SOA applications in well-formed J2EE code.

Key Features:

- JCA-compliant Service-component Framework
- JMS based messaging
- BPEL support for composite components and applications
- Synchronous and Asynchronous Services in a single framework
- Support for multiple languages : Java, C, C++, C# and others

With Fiorano you can:

- Directly deploy Service Components and BPEL processes to any standard J2EE container
- Rapidly deliver ACID transactions across front-end (JSP) and backend processes (Databases, ERPs, etc.)
- SOA enable your existing J2EE application (EJB, JCA, JMS) without reengineering or code-generation
- Easily debug complex flows spanning multiple applications and middleware
- Perform runtime service deployment, service updates and application changes

Download your copy of Fiorano today!

http://www.fiorano.com/downloads

Fiorano®

Enabling Change at the speed of thought

Code Sample 1: The Mabon Protocol:

```
mabon:/managedBean.getValidDates
```

Code Sample 2: String returned after Mabon has evaluated the Mabon Protocol:

```
/<context-root>/<mabon-servlet-mapping>/managedBean.getValidDates
```

During an AJAX request, this URL is sent on the request and intercepted by the FacesLifecycleServlet.

Mabon: Initial Request

The Mabon implementation is designed specifically for AJAX requests and implements a communication channel using JSON syntax. This solution lets AJAX components that use managed beans fetch data and communicate with the server without having to go through a full JSF lifecycle. So how does it work? At application start-up Mabon will add the MabonLifecycle as part of the JSF LifecycleFactory context.

On the initial request, Mabon is just delegating through to the underlying JSF implementation and is active only during the Render Response phase, if needed.

In the Figure 3 sequence diagram, a page that contains a custom AJAX component is executed. To work, the AJAX component has to get data from an underlying backing bean. During encodeBegin(), the AJAX Renderer for that component will use the Mabon protocol – mabon:/ – to write out a target URL that references the backing bean. To get this URL, the Renderer will call the getResourceURL() on the ViewHandler. It will pass a string matching the method binding expression for the backing bean (for example, mabon:/managedBean.getSuggestions). The getResourceURL() method in MabonViewHandler will return a full path – /<context-root>/<mabon-servlet-mapping>/managedBean.getSuggestions – that can be written out to the document.

Mabon: Data Fetch Request

After the page has been rendered to the client, it contains a target URL to the backing bean that's needed by the AJAX component to fetch data (for example, /<context-root>/<mabon mapping>/managedBean.getValidDates). In subsequent AJAX requests, this string will be intercepted by the Mabon implementation and used to invoke the backing bean and return the result to the client.

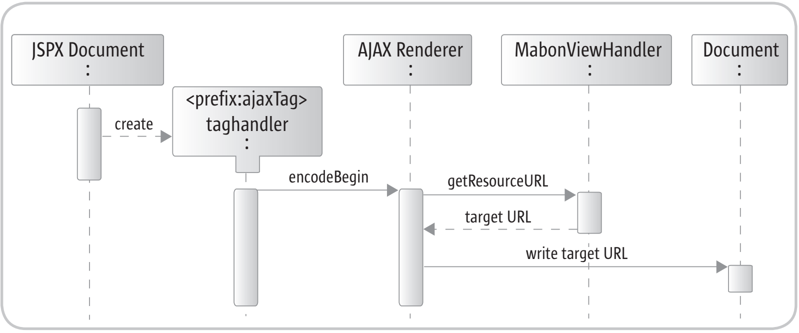


Figure 3 Sequence diagram of the Mabon initial request

On submit an AJAX-enabled component creates a new XMLHttpRequest object, which communicates asynchronously with the server to get data from the managed bean. This request is intercepted by the FacesLifecycleServlet, which routes the request through the Mabon Lifecycle instead of the default JSF Lifecycle.

When the FacesLifecycleServlet intercepts the request, the request processing starts by calling each Mabon lifecycle phase in sequence. First, you execute the ApplyRequestValuesPhase, which will decode the request and get the managed bean reference and method arguments needed for the managed bean off the request. Second, you execute the InvokeApplicationPhase that will create a MethodBinding based on the managed bean reference, invoke this MethodBinding passing any arguments, and return the result. Third, the RenderResponsePhase takes the result and writes it back to the client.

Mabon APIs

The following sections cover the available APIs and how to register Mabon with an application.

Mabon Servlet Configuration

If you're planning to use Mabon for your AJAX-enabled components, you should be aware that it adds an extra step for the application developer using your JSF component library. The application developer needs to add the entry shown in Listing 1 to the Web application configuration file web.xml.

The servlet class net.java.dev.mabon.webapp.FacesLifecycleServlet and the initialization parameter (for example, net.java.dev.mabon) is part of the Mabon contract. The application developer can decide to set the mapping to the same URL-pattern(s) as defined by default (for example, /mabon/*) or override the default URL mapping in case it's colliding with resources used by the Web application. Mabon automatically consumes this URL mapping change without requiring any code changes.

Mabon JavaScript APIs

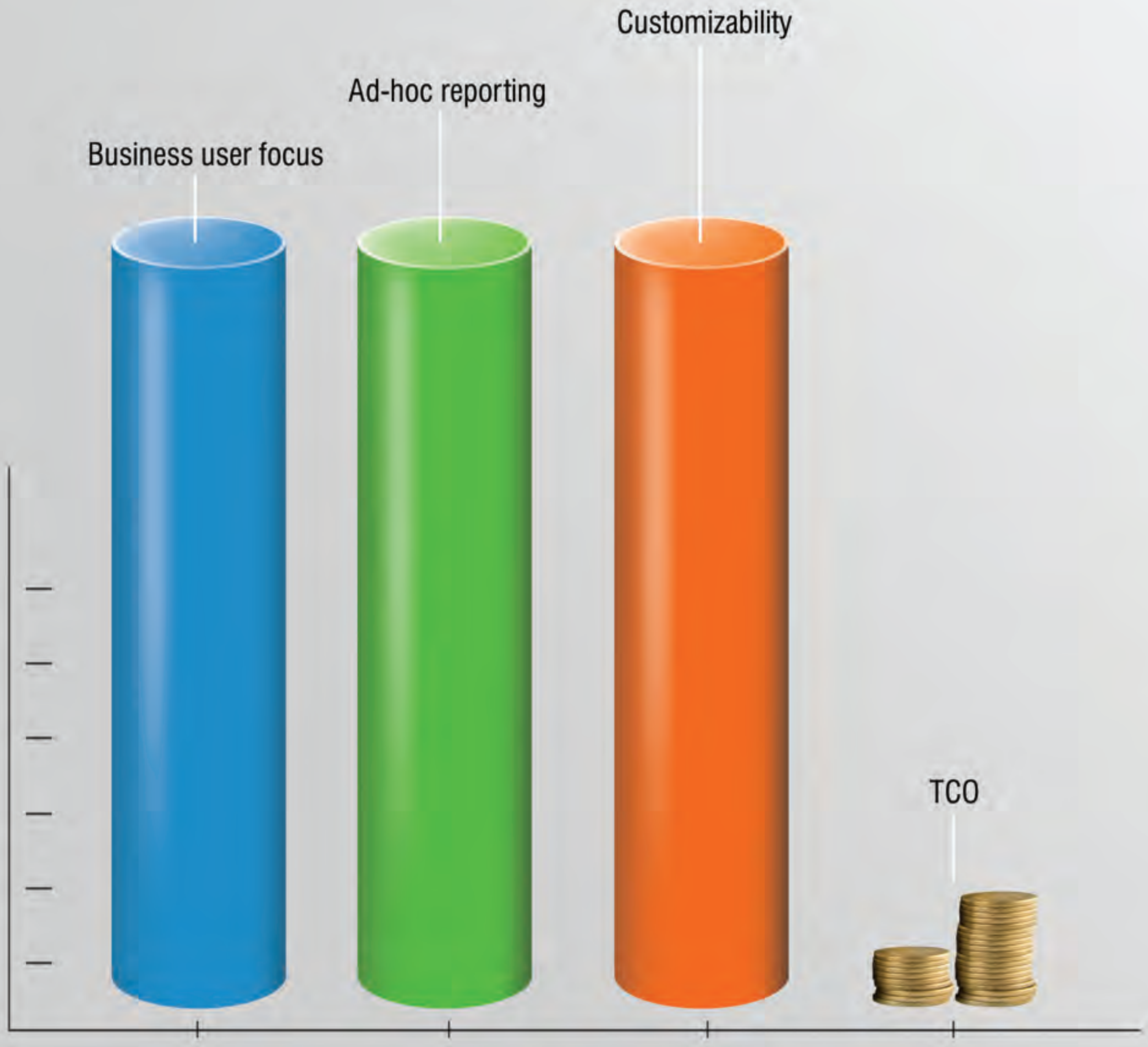
The Mabon project provides a convenience JavaScript library that you can use to send your request to the server. The Mabon send() function leverages the Dojo toolkit's bind() function to communicate asynchronously with the server. For more information about the Dojo Toolkit visit the Dojo Web site at http://dojotoolkit.org/.

Listing 2 shows the source of the Mabon JavaScript library.

The Mabon send() function takes one argument – a Map. To call the mabon.send() function from your AJAX implementation, you have to construct the Map using JavaScript Map syntax as shown in the following code:

```
mabon.send(
            { url: targetURL,
              args: [item1, item2],
              callback: callback_function }
          );
```

Why is IntelliVIEW a better choice than other reporting software?



Easy set up	Saves time and valuable resources
Easy report creation	Minimal training required
Ad-hoc reporting	Without IT intervention
Offline analytics	Improved decision making
User-configurable dashboards	Quicker access to important data
Concurrent user licensing	Increased cost efficiency
Support & services	25% of AMC can be used for services
Global delivery model	Tremendous cost advantage

To know more visit **http://www.intelliview.com/jdj**
Email: sales@intelliview.com
Call toll-free 1-866-99IVIEW

*Intelli***VIEW**
Reporting & analytics made easy!

product of Synaptris

"Mabon offers an easy way of providing data to AJAX-enabled components using the managed bean facility in JSF"

The targetURL is the resource URL that's written to the client (for example, /context-root/mabon-servlet-mapping/managedBean.methodName). The targetURL will be intercepted by the FacesLifecycleServlet and deciphered by the Mabon Apply Request Values phase.

Mabon Protocol

Now that you know how to configure Mabon, it's time to look at how you can reference the managed beans needed to fetch data. The Mabon protocol-like syntax is convenient and easy to understand. The syntax starts with mabon:/ followed by the managed bean name and finally the method name, as shown below:

```
ViewHandler.getResourceURL(context,
                        "mabon:/<managed bean name>.<method>");
```

The syntax uses a prefix to indicate this is a Mabon-managed request, the managed bean name, and the method needed. This syntax – mabon:/<managed bean><method> – defined by the Mabon contract is used to return a target URL referencing the managed bean.

Summary

This article discussed how you can use AJAX to fetch data and leverage the JSF managed bean facility as a data source. It also covered the different XMLHttpRequest response types – responseText and responseXML – that you can use to return the result from the server. We also showed you how to use the eval() function to parse JSON-syntax responses efficiently.

We covered a new Open Source project called Mabon that extends JSF to provide a custom lifecycle that invokes a managed bean method remotely and then transfers the result to the client using JSON syntax.

In our next article in this series of building Rich Internet Components with JavaServer Faces, we're going to look at how we can bring the knowledge from the three previous articles together and create a <jdj:inputSuggest> component that leverages AJAX, Mabon, and Weblets.

• • •

This article is based on, and contains excerpts from, the book *Pro JSF and Ajax: Building Rich Internet Components* by Jonas Jacobi and John Fallows, to be published by Apress in February 2006.

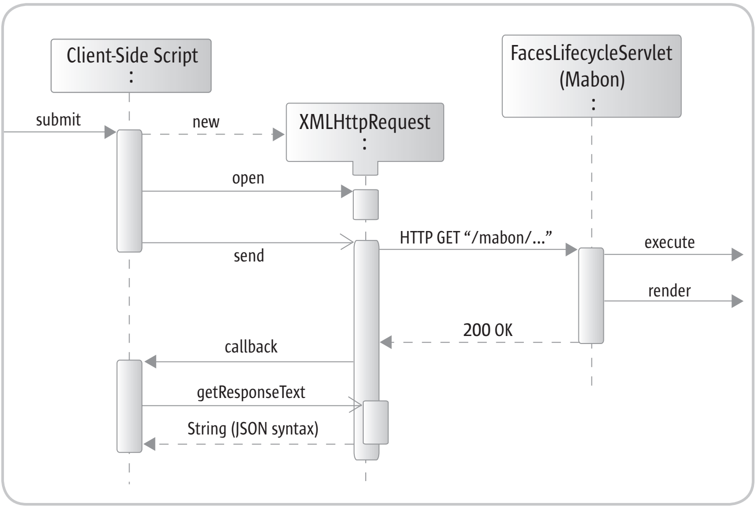


Figure 4 Sequence diagram of Mabon/AJAX data fetch request

Listing 1: Mabon Servlet Configuration

```
<servlet>
<servlet-name>Mabon Servlet</servlet-name>
<servlet-class>
  net.java.dev.mabon.webapp.FacesLifecycleServlet
</servlet-class>
<init-param>
<param-name>javax.faces.LIFECYCLE_ID</param-name>
<param-value>net.java.dev.mabon</param-value>
</init-param>
</servlet>
...
<servlet-mapping>
<servlet-name>Mabon Servlet</servlet-name>
<url-pattern>/mabon/*</url-pattern>
</servlet-mapping>
```

Listing 2: The mabon.js Library

```
dojo.provide("net.java.dev.mabon");

 /**
  * @param  kvparams (map)
  *          {
  *            url        the destination URL
  *            args       the arguments array
  *            callback   the result callback,
                signature (result)
  *          }
  */
 net.java.dev.mabon.send = function (
   kvparams)
 {
   var params = new Array();
   for (var i=0; i < kvparams.args.length; i++)
   {
     var arg = kvparams.args[i];
     if (typeof(arg) == "string")
       params.push("'" + arg + "'");
     else
       params.push(arg);
   }

   var content = {args:'[' + params.join(',') + ']'};

   dojo.io.bind(
   {
     url: kvparams.url ,
     method: 'get',
     content: content,
     mimetype: "text/javascript",
     load: function(type, data, evt) { kvparams.
           callback(eval(data)); },
     error: function(type, data, evt) { alert('Oops!
The server returned an error, please try again.'); }
   });
 }
```

Complex JAVA J2EE Hosting made easy.

WebAppCabaret sm

http://www.webappcabaret.com/jdj.jsp
1.866.256.7973

JAVA J2EE-Ready Managed Dedicated Hosting Plans:

Xeon III SAMEDAY SETUP

Dual 3.2 GHz Xeons
4GB RAM
Dual 73GB SCSI
1U Server
Firewall
Linux
Monitoring
NGASI Manager

$399 *monthly*

Pentium 4 I SAMEDAY SETUP FREE SETUP

2.4 GHz P4
2GB RAM
Dual 80GB ATA
1U Server
Firewall
Linux
Monitoring
NGASI Manager

$199 *monthly* 2nd month FREE

4Balance I

1 Database Server and 2 Application Servers connected to 1 dedicated load balancing device.
Dual Xeons.
High-Availability.

$1724 *monthly*

NEW! Geronimo 1.0 Hosting . Virtual Private Servers(VPS) starting at $69/month

At **WebAppCabaret** we specialize in **JAVA J2EE Hosting**, featuring **managed dedicated servers** preloaded with most open source JAVA technologies.
PRELOADED WITH:
JDK1.4 . JDK1.5 . Tomcat . Geronimo . JBoss . Struts . ANT . Spring . Hibernate
Apache . MySQL . PostgreSQL . Portals . CRM . CMS . Blogs . Frameworks
All easily manage via a web based control panel.
Details:
-All Servers installed with the latest Enterprise Linux
-Firewall Protection
-Up to 60 GB daily on site backup included at no extra charge per server.
-Database on site backup every 2 hours
-Daily off site database backup
-A spare server is always available in case one of the server goes down
-Intrusion detection.
-24x7 Server and application monitoring with automatic self healing
-The Latest Bug fixes and Security updates.
-Tier 1 Data Center. 100% Network Uptime Guarantee
-Guaranteed Reliability backed by industry-leading Service Level Agreements

Log on now at **http://www.webappcabaret.com/jdj.jsp** or call today at **1.866.256.7973**

NGASI POWERED

WebAppCabaret sm

JAVA J2EE Hosting

Putting a Face on Web Services and SOA

It's easier than you think

by Chris Schalk and Michael O'Neill

Service-Oriented Architecture (SOA) is a hot topic among analysts, CIOs, and technology marketers, but the path from high-level architectural principles to programming a functioning, real-world service isn't always clear, especially since, in the end, you still need to create a clean user interface that's as flexible as the services it consumes.

Why SOA?

The concepts underlying SOA aren't a major divergence from standard design principles like abstraction, encapsulation, and reuse. To put it simply, SOA is a software architecture in which application functionality is encapsulated as independent services. These services are in many cases Web Services but can also be services derived from various technology types. Clients in different applications or business processes can then consume these services. More important, an SOA is designed to decouple the implementation of a software service from the interfaces that call that service. This lets clients of a service rely on a consistent interface regardless of the implementation technology of the service.

Instead of building big monolithic applications, developers can build more agile services that can be deployed and reused across an organization for different applications and processes. This allows for better reuse of software functionality, as well as for increased flexibility because developers can evolve the implementation of a service without necessarily affecting the clients of that service.

To this end, the main requirement of an SOA is that the interface to the services is decoupled from the implementation. This separation provides the central benefit to SOA, both from a programmatic perspective and a financial one. Existing systems, both legacy and modern, internal to the company and external, can be exposed and consumed as services, simplifying the task of development.

Chris Schalk is a principal product manager and Java evangelist for Oracle's application server and development tools division. Chris' primary expertise is Web application development and he is responsible for defining the Web development experience for Oracle JDeveloper.

A Face for SOA

A discussion of adopting an SOA typically focuses on Web Services reliability, interoperability, and messaging — that is, on how to integrate existing software into processes and composite applications, and *not* on how users will interact with those processes. Without a user interface, however, there isn't much of an application.

JavaServer Faces (JSF) complements SOA by allowing for an easy way to add the last piece of the SOA puzzle: a human-usable interface. JSF is a standard J2EE technology providing a component-based approach to user interface development, as well as adherence to the model-view-controller architecture that Jakarta Struts helped promote as the standard Web application architecture.

JSF components can be likened to services in SOA: they encapsulate many of the common requirements and commands of a Web-based user interface. Instead of having to programmatically build HTML tables or write data validation in JavaScript, components can provide all of this functionality, enabling a developer to simply drop a component onto a page and set its properties. Even more compelling, the more comprehensive JSF component sets, like Oracle's ADF Faces, are built with multiple sets of renderkits that allow the components to be consumed by many different clients and devices, including PDAs, mobile phones, Telnet, and IM.

Service-Oriented Architecture Details

With Web Services providing an implementation-agnostic messaging protocol, they are the standard messaging protocol for building SOA applications. Combined with standards for reliable messaging, security, and interoperability, Web Services are the ideal SOA backbone.

The services that comprise the building blocks of an SOA application have a common characteristic: a public interface made up of exposed methods that can be consumed by other parts of the application. The public interface can be implemented in any technology (.NET, J2EE, COBOL, PL/SQL, etc.).

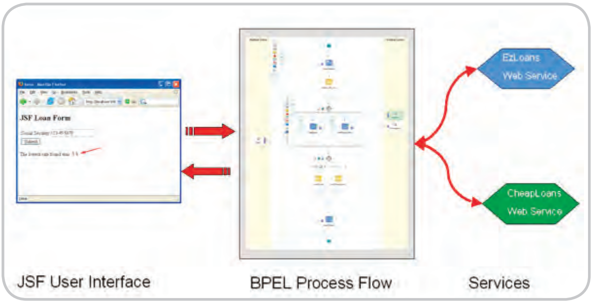


Figure 1 Architecture of our JSF-BPEL SOA Application

can your ecommerce software do this?

NEW v4.1

Introducing the pioneer of flexible ecommerce architecture.

Elastic Path is a revolutionary "developer friendly" Java-based ecommerce platform for building sophisticated and evolving online stores. The first to leverage open source frameworks such as Struts and Hibernate, Elastic Path is refreshingly simple to customize, easy to integrate, quick to deploy, and surprisingly cost effective.

Request your FREE 30 day trial at:
www.elasticpath.com/download

elasticpath

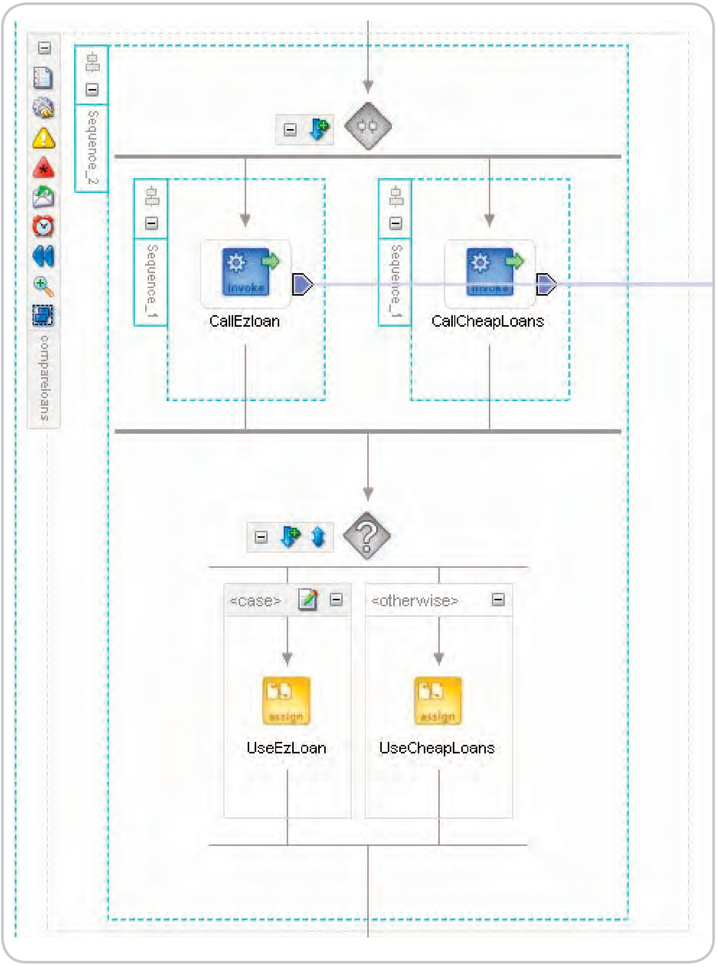


Figure 2 The BPEL Process to check a loan rate amount

By far the most common type of service used in SOA is Web Services. However Web Services by themselves don't provide the basic infrastructure you'd expect in a programming language such as exception handling and state maintenance. To orchestrate a set of Web Services into a business process requires either writing a great deal of low-level code or the use of the Business Process Execution Language (BPEL). BPEL is an emerging OASIS (Organization for the Advancement of Structured Information Standards) standard that enables the orchestration of Web Services. While the BPEL language is represented by a somewhat complex XML schema, there are visual BPEL tools, like Oracle's BPEL Designer, that simplify the orchestration of Web Services into business process flows using a drag-and-drop, declarative development environment.

JavaServer Faces and SOA - A Simple Example

To get a feel for how composite SOA applications, which include both the BPEL process flow as well as a user interface, a simple example SOA/JSF application will be introduced here and more importantly the exact steps required to build the application will be presented. By considering the steps needed to build an SOA application that uses the latest J2EE user interface standard, one can see that building composite SOA applications doesn't have to be an overly complex ordeal.

Consider an example application that consists of both a JSF user interface and a backend BPEL process flow. The JSF user interface includes an entry form for personal information, most importantly a Social Security number. The JSF application then submits the user-provided Social Security number to the BPEL process that shops for the lowest loan rate from a set of independent financial institutions that offer loan rate quotes via Web Services. Once the lowest rate is determined, it's returned back to the JSF application and reported to the user.

For simplicity our example BPEL choreography consists of two *Partner Links* based on the Web Services that provide two competing loan rates.

Building the BPEL Process Flow

Our example BPEL process flow consists of an orchestration of different services (referred to as Partner Links), which compares the loan rate values returned from the services to determine the lowest one. To prototype this example, a set of simple Web Services can easily be constructed to return hard-coded values as loan rates. The following steps show how to build these using Oracle JDeveloper 10g (version 10.1.3).

Building the Loan Rate Services

The prototype Web Services for this example application are based on simple Java classes that return hard-coded values. They are:

- EzLoans – When supplied with a Social Security number, returns a current loan rate.
  – Return method: getDailyRate
- CheapLoans – Also returns a current loan rate when supplied a Social Security code.
  – Return method: getLoanRate

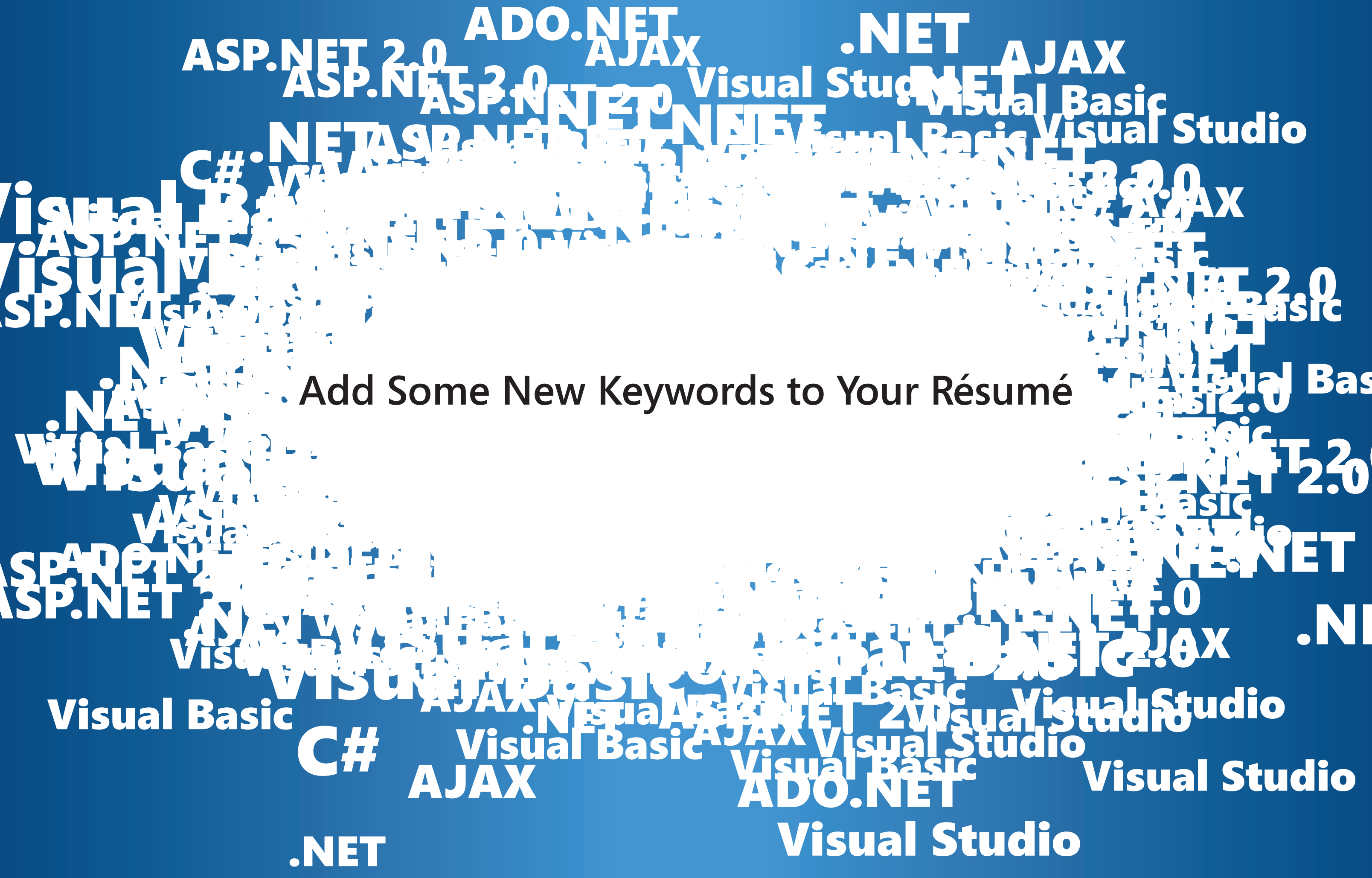
In the interest of conveying the concepts behind the services without getting bogged down in the actual implementation of these services, Web Services will be auto-generated in JDeveloper based on simple Java classes that contain public methods that return current loan rate information and can be exposed as Web Services.

As an example, here is the core source code for the EzLoans Web Service:

```
package com.jdj.ezloan;
```

"More important, an SOA is designed to decouple the implementation of a software service from the interfaces that call that service"

Michael O'Neill is manager of developer programs at Oracle, with six years of experience in Web and UI development utilizing both Java and Microsoft technologies.



Add Some New Keywords to Your Résumé

Learn new skills by attending the ASP.NET 2.0 Webcast Series.

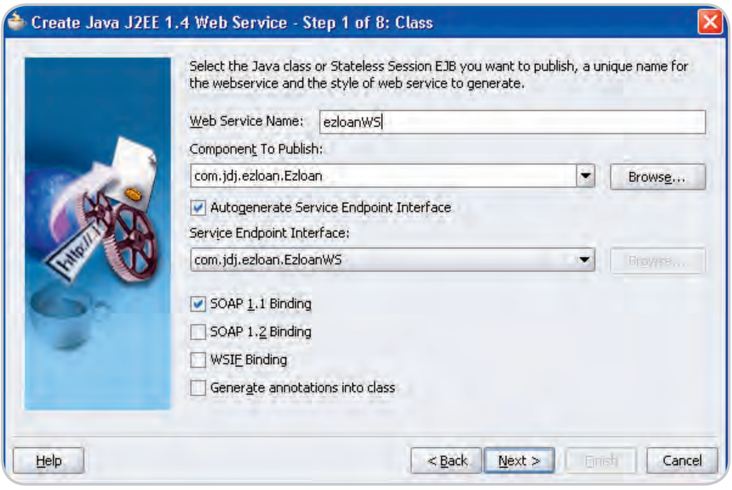
Attend 3 webcasts and you'll receive:

- Microsoft® Visual Studio® 2005 Standard Edition (Not for Resale)
- Programming ASP.NET 2.0 Core Reference (5 Chapters)
- And more—a $400 value*

> Register now at **www.learn2asp.net/webcasts.**

Microsoft

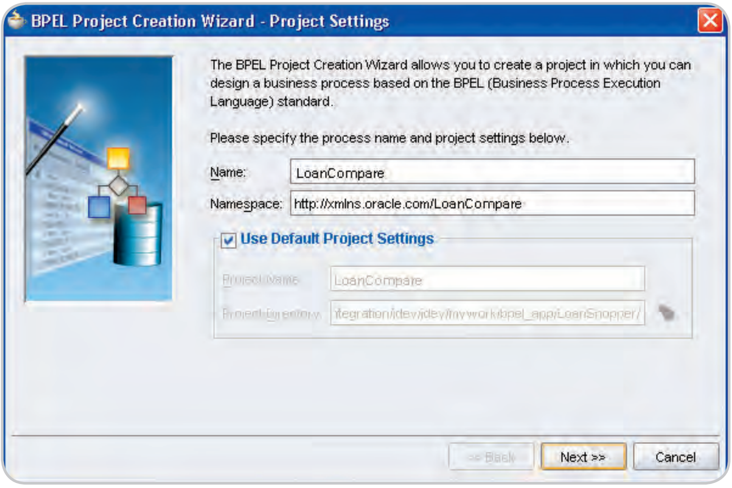
*Offer good in the United States and Canada only through April 17, 2006, while supplies last. This offer is available only to professionals employed in the area of software or Web development who watch three live or on-demand webcasts as part of the ASP.NET 2.0 Webcast Series. This offer is nontransferable. Microsoft is not responsible for nondelivery of gift due to incorrect postal information provided by respondent. Limit: one per customer. Please allow 6–8 weeks for delivery.


Create Java J2EE 1.4 Web Service - Step 1 of 8: Class

Select the Java class or Stateless Session EJB you want to publish, a unique name for the webservice and the style of web service to generate.

Web Service Name: ezloanWS
Component To Publish: com.jdj.ezloan.Ezloan
☑ Autogenerate Service Endpoint Interface
Service Endpoint Interface: com.jdj.ezloan.EzloanWS
☑ SOAP 1.1 Binding
☐ SOAP 1.2 Binding
☐ WSIF Binding
☐ Generate annotations into class

Figure 3 JDeveloper Web Service wizard


BPEL Project Creation Wizard - Project Settings

The BPEL Project Creation Wizard allows you to create a project in which you can design a business process based on the BPEL (Business Process Execution Language) standard.

Please specify the process name and project settings below.

Name: LoanCompare
Namespace: http://xmlns.oracle.com/LoanCompare
☑ Use Default Project Settings
Project Name: LoanCompare
Project Directory: ...tegration/dev/dev/myworkbpel_app/LoanShopper/

Figure 4 Creating the LoanCompare BPEL project

```
public class EzLoan {

  public float  dailyRate = (float)5.6;

    public EzLoan() {
    }

    public float getDailyRate(String soc_security) {
      System.out.println("EzLoans returning loan rate " + dailyRate
+ " for " + soc_security);
        return elRate;
    }
}
```

As you can see, the method **getDailyRate( )** simply returns the value of the **dailyRate** that is hardcoded to a value of 5.6. To expose this method as a Web Service, an IDE can quickly generate the necessary code, which will expose the method as a Web Service when deployed to an application server.

In a similar fashion the **CheapLoans** Java class with a method **getLoanRate( )** can also be exposed as a Web Service.

Once the necessary *plumbing* code for the Web Services has been created, the EzLoans and Cheaploans Web Services can be deployed to a J2EE app server such as Oracle Containers for J2EE (OC4J). After successful deployment, the services can be viewed and tested to make sure they can process requests.

Orchestrating the Services - Building a BPEL Flow

Once the services are created you can then build a BPEL process flow that consumes these services. Fortunately this can be done in an entirely visual manner using the BPEL Designer in Oracle JDeveloper. The process for this is as follows:

1. Create a new synchronous BPEL project in JDeveloper
2. Create Partner Links mappings to the deployed Web Services
3. Create a process flow that invokes the Partner Links (Web Services) and compares their values and returns the lowest loan rate value.
4. Compile and deploy to a BPEL Process Manager

Building a BPEL process flow consists primarily of editing the XML document that defines the flow. Previously mentioned as an alternative to coding directly in XML, it's possible to use Oracle JDeveloper's BPEL Designer feature, which provides a graphical environment for designing BPEL process flows and deploying them to a BPEL process manager.

Creating a New BPEL Project

Creating a new synchronous BPEL project is easily done using JDeveloper's BPEL project wizard. Once the project is created, a set of initial files are generated that include a starter ".bpel" file, which is the XML source of the orchestration, and a ".wsdl" file, which is the Web Services Descriptor document for the BPEL process. This is needed since the BPEL process flow, when deployed, is accessible as a Web Service.

Once the starter BPEL project is generated, the initial (starter) orchestration will appear in the BPEL visual designer. The next step is to create Partner Links that the BPEL process will call (or *invoke)* the loan rate Web Services.

Creating Partner Links and Defining Variables

Creating Partner Links to our loan rate Web Services is done by dragging and dropping a Partner Link component onto the right side of the BPEL diagram and providing the WSDL URL to the dialog.

Once the *EzLoan* and *CheapLoans* Partner Links have been created, they'll be "invokable" in the BPEL process flow. We'll return to these shortly.

Since our JSF application will be supplying a Social Security number, we'll declare a global "soc_security" variable in the BPEL flow. This is done by clicking on the *variables* icon on the upper left of the flow and declaring a new global variable in the dialog.

Building the Flow

Now that we have our Partner Links and a soc_security global variable, we can build our BPEL process flow. We start by dragging and dropping an "Assign" element to just below the *ReceiveInput* node. After the element is dropped, we double-click on it to set its properties. In the dialog we name it "assign_soc_sec" and create a *Copy Rule* and assign the incoming (Input) value to the **soc_security** global variable. Later in the flow we'll use the **soc_security** variable as an argument to the loan rate Web Services.

The next step in building our BPEL process flow is to insert a *scope* element just below the *soc_security* assign element. The *scope* element allows for a collapsible sub-region of the overall flow. Our new scope element will contain our loan comparison process flow so we can name it "compareloans."

In the BPEL Designer, open the new scope region by clicking on its "+" sign. The scope element will expand into an empty box with a "Drop activity here" text message inside of it. Next we'll drag and drop a "Flow" element inside the scope region. The Flow element lets our application invoke more than one Partner Link (Web Services) at the same time.

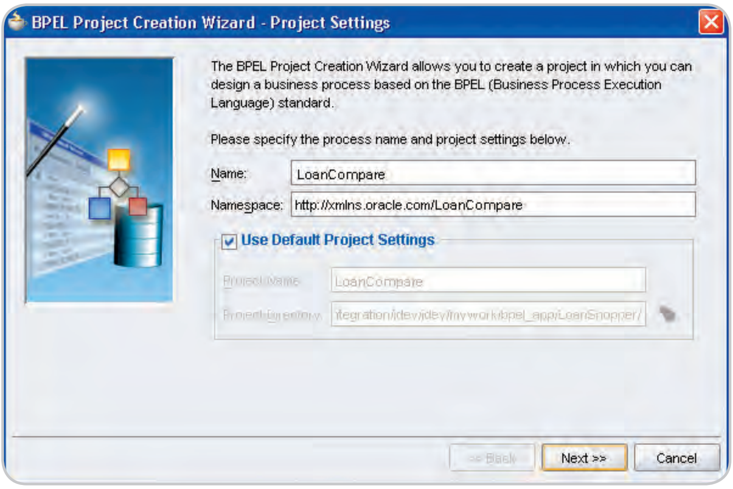
After the Flow box is added, we open it to reveal empty *activity* boxes. Here is where we add the invoke elements from the palette to invoke the services provided by the Partner Links. After dragging an Invoke element into one of the empty flow activity boxes, we double-click it to edit its properties. This Invoke element will call the EzLoans Partner Link so we set the "Partner Link" property with EzLoan. Once selected, we can then select the **getDailyRate** Operation of the service. For the Input variable we select the **soc_security** global variable and the output of the invoked service is stored in a newly created global variable called "**EzLoanOut**."

A separate Invoke method is then dropped into the alternate flow activity box and its properties are set in a similar manner except it will call the **CheapLoans** Partner Link and the global output variable for it will be "**CheapLoansOut**."

After adding the invoke methods, the next step is to insert a "Switch" element that will let the BPEL process compare the different rates returned. In the BPEL Design we drag a Switch element and drop it just below the Flow element.

The BPEL Switch element works by having both a condition expression and two possible operations, based on the evaluation of the condition expression. We'll edit the condition to compare the two loan rates returned from the Partner Link invoked. Editing the Switch condition is done by clicking on the small edit icon at the top of the Switch element to invoke an expression builder dialog that lets us build a logical expression to compare the values of the different loan rate amounts.

By selecting from both the variables tree on the left and the functions window on the right, one can easily build an expression from which to determine which activity to proceed with.


BPEL Project Creation Wizard - Project Settings

The BPEL Project Creation Wizard allows you to create a project in which you can design a business process based on the BPEL (Business Process Execution Language) standard.

Please specify the process name and project settings below.

Name: LoanCompare

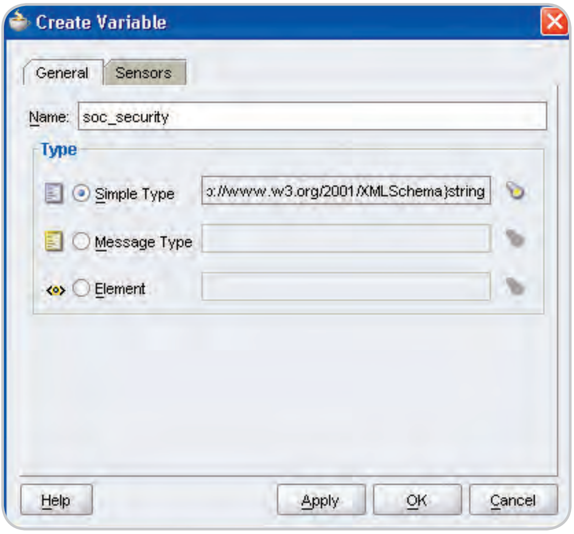
Namespace: http://xmlns.oracle.com/LoanCompare

[✓] Use Default Project Settings

Project Name: LoanCompare

Project Directory: ...tegration\devidev\myworkubpel_app\LoanShopper\

Figure 5 Creating partner links for the loan rate Web Services


Create Variable

General | Sensors

Name: soc_security

Type

[● Simple Type] ...://www.w3.org/2001/XMLSchema}string

○ Message Type

○ Element

Figure 6 Creating a Soc_security global variable

For our application, if the value from the EzLoan rate is less (<) than the CheapLoans Value, then we'll proceed with the activity on the left side (which we'll place in an Assign element). The actual expression used is:

```
bpws:getVariableData('EzLoanOut','parameters','/ns4:
getDailyrateResponseElement/ns4:result') < bpws:
getVariableData('CheapLoansOut','parameters','/ns5:
getLoanrateResponseElement/ns5:result')
```

If the expression evaluates to **true** the application has to return the **EzLoan** value to the end user since it's the lower value, otherwise it

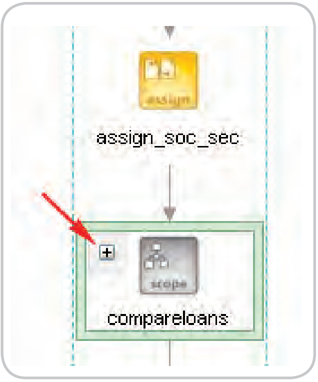
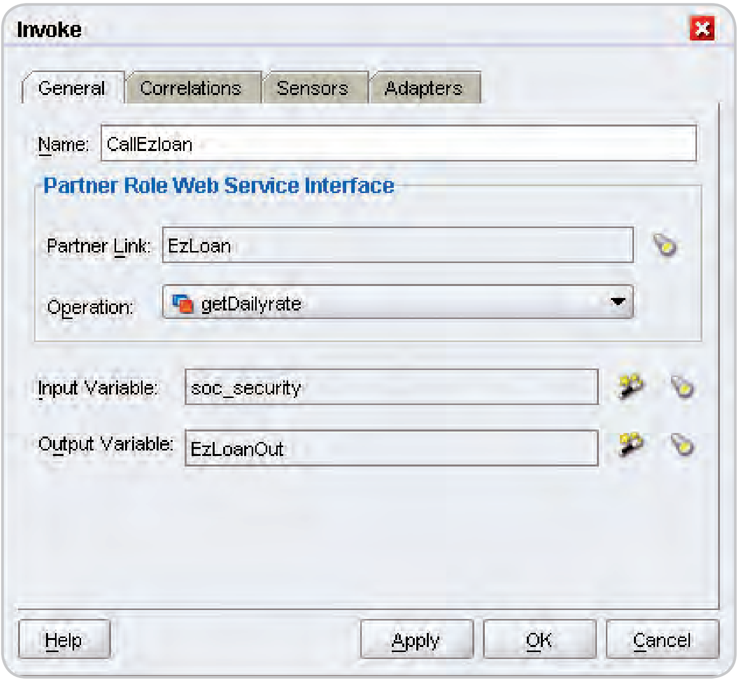


Figure 7 Creating a new "compareloans" scope

"Creating a new synchronous BPEL project is easily done using JDeveloper's BPEL project wizard"



Invoke

General | Correlations | Sensors | Adapters

Name: CallEzloan

Partner Role Web Service Interface

Partner Link: EzLoan

Operation: getDailyrate

Input Variable: soc_security

Output Variable: EzLoanOut

Figure 8 Setting the properties of an invoke method for EzLoan

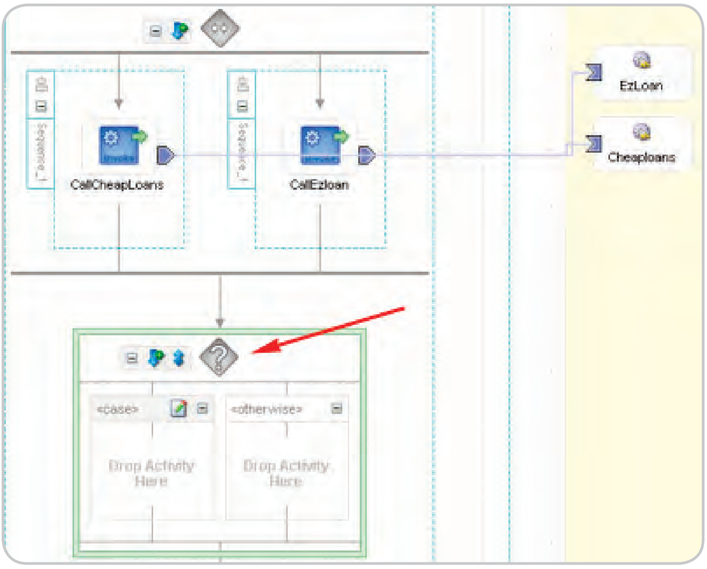


Figure 9 Dropping the Switch element

should return the **CheapLoans** value. Returning the loan rate values is done by using Assign elements.

Inside each Assign element a Copy rule is created to place the value of either the global variable **EzLoanOut** or **CheapLoansOut** into the global **outputVariable** of the overall BPEL process.

Once the Assign elements have their respective Copy Rules inserted, the overall BPEL process flow is complete. At this point the BPEL project can be compiled and deployed to the Process Manager server.

Once deployed to the server, the BPEL process becomes active and can be called as a Web Service. It can also be tested and fully managed from the Process Manager though its Web management console.

In the Console we can also see the final URL of the deployed BPEL process first by clicking on the LoanCompare BPEL process and then clicking on the WSDL tab. The URL for our application is: http://localhost:9700/orabpel/default/LoanCompare/1.0/LoanCompare?wsdl. This is the URL which will be called from the JSF Web application.

Building a JSF Web Application to Call Our LoanCompare Process

This simple JSF example application will contain a form page, which lets the user enter a Social Security number and get the lowest loan rate result returned from the BPEL process.

Building a JSF Loan Application

To build our JSF loan application, we'll again use Oracle JDeveloper 10g (10.1.3).

To get started, we create an empty project named FacesLoanApp.jpr. (File->New..->Empty Project) Once the project is created, we create a JSF-enabled JSP page, form.jsp, using the JSF JSP wizard. (File->New -> Web -Tier -> JSF -> JSF JSP) As the wizard starts, we enter the name of the page "form.jsp" and click Next to continue.

On the Component Binding page of the wizard, we select the option to Autobind the page. The autobinding feature in JDeveloper creates bound declarations of JSF user interface components as we drag and drop them onto JSF pages. For example, if a button is dropped on a page, it will have a corresponding declaration of a Faces button component created in a registered *backing bean* for the page.

The term "backing bean" refers to a special type of JSF managed bean that is created as a *code-behind* Java class, which is used to hold declarations of the JSF UI components as well as other page-based event logic. JSF managed beans are simply Java classes that are registered in the faces-config.xml file that are then usable in the Faces application.

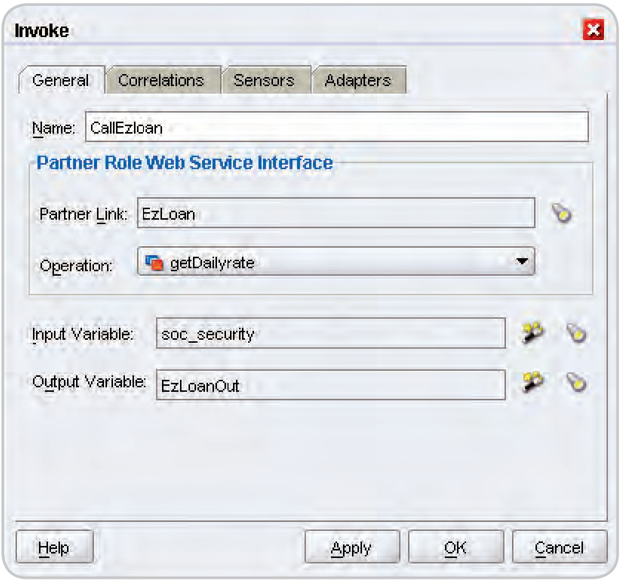
Once the form.jsp page is generated, the visual editor appears. At this point we can add a simple banner "JSF Loan Form" at the top of the page using simple HTML <h2> formatting.

Below the banner, we'll add a simple form that accepts a Social Security entry along with a button to submit the form. For this we'll add JSF UI components to the page by dragging and dropping them from the Component Palette on the right.

The first UI component to drop on to the page is a **Panelgrid** component. After dropping the component, a wizard appears. In the wizard choose the "Create Empty Panel Grid" option and specify that it will contain two columns. Click Finish to insert the empty panelgrid into the page and it will be visible as an empty dotted blue-lined box. This box will contain our form items.

Drag and drop an **OutPut Label** component followed by an **Input Text** component. To complete the example form, drag and drop a **Command Button** from the palette as well.

Now that the components have been added to the page, the label and the button can have their **value** properties edited with values "Social Security" and "Submit" respectively. You'll see the value changes in the visual editor once they are made.


Invoke

General | Correlations | Sensors | Adapters

Name: CallEzloan

Partner Role Web Service Interface

Partner Link: EzLoan

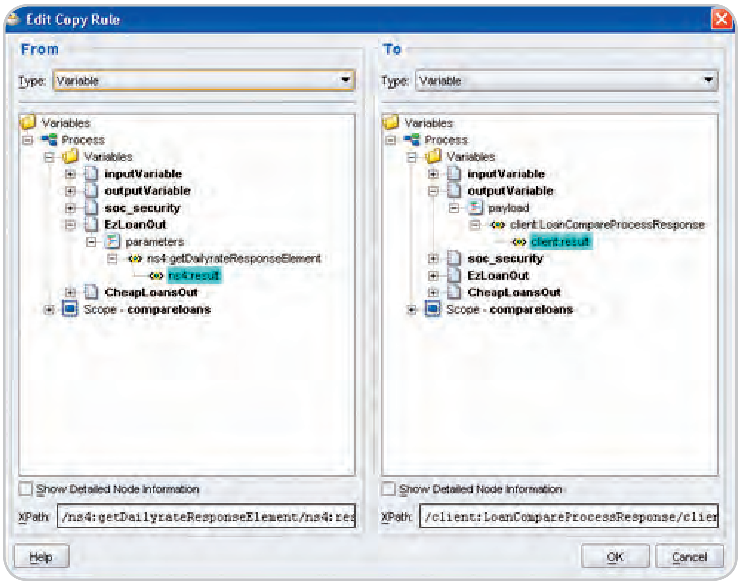
Operation: getDailyrate

Input Variable: soc_security

Output Variable: EzLoanOut

Help | Apply | OK | Cancel

Figure 10 The expression builder dialog


Edit Copy Rule

From — Type: Variable

```
Variables
  Process
    Variables
      inputVariable
      outputVariable
      soc_security
      EzLoanOut
        parameters
          ns4:getDailyrateResponseElement
            ns4:result
      CheapLoansOut
    Scope - compareloans
```

Show Detailed Node Information

XPath: /ns4:getDailyrateResponseElement/ns4:res

To — Type: Variable

```
Variables
  Process
    Variables
      inputVariable
      outputVariable
        payload
          client:LoanCompareProcessResponse
            client:result
      soc_security
      EzLoanOut
      CheapLoansOut
    Scope - compareloans
```

Show Detailed Node Information

XPath: /client:LoanCompareProcessResponse/clie

Help | OK | Cancel

Figure 12 Assigning the EzLoanOut value to the outputVariable using a copy rule

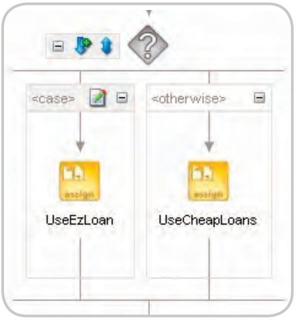


Figure 11 A Switch element with two Assign child elements

The final edit to our page is to add a bit of text below the form, "The lowest rate found was:" along with an **Output Text** UI component that will be used to display the returned results from the BPEL Web Service. Once added, the final page looks like Figure 15.

Creating a JSF Action Method to Call the BPEL Web Service

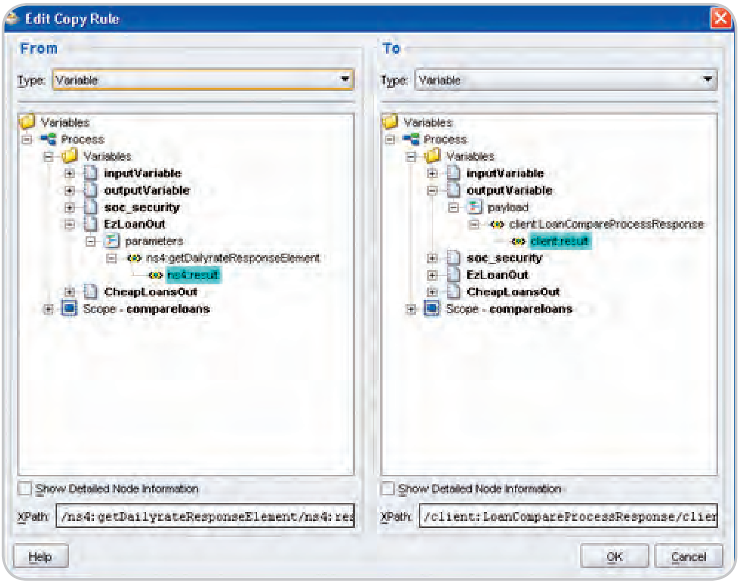
Now our application will be updated so that when the button is clicked, it will call the LoanCompare BPEL Web Service with the value supplied in the Social Security input field. This is easily done by double-clicking on the submit button. This will autogenerate a Faces action method and opens the code editor. At this point we can edit the code that executes as the button is clicked. The code generated is:

```
public String commandButton1_action() {
    // Add event code here...
    return null;
}
```

We'll now modify it to call our BPEL Web Service. Before we can call the service, we must create a Web Service proxy class which will provide a way to call the service from Java.

To create the proxy class, you can use the Web Service Proxy wizard in JDeveloper (File->New…->Business-Tier->Web Services->Web Service Proxy). As the wizard commences, all that is needed to generate the necessary proxy to call the BPEL Web Service is to supply the WSDL URL http://localhost:9700/orabpel/default/LoanCompare/1.0/LoanCompare?wsdl.

Accept the remaining defaults and click Finish to generate the Proxy code. As the code generates you will see a Java Proxy client, LoanCompareWSHttpPortClient.java which is a runnable Java client with a main method. The main method looks like:


Edit Copy Rule

From — Type: Variable

```
Variables
  Process
    Variables
      inputVariable
      outputVariable
      soc_security
      EzLoanOut
        parameters
          ns4:getDailyrateResponseElement
            ns4:result
      CheapLoansOut
    Scope - compareloans
```

Show Detailed Node Information

XPath: /ns4:getDailyrateResponseElement/ns4:res

To — Type: Variable

```
Variables
  Process
    Variables
      inputVariable
      outputVariable
        payload
          client:LoanCompareProcessResponse
            client:result
      soc_security
      EzLoanOut
      CheapLoansOut
    Scope - compareloans
```

Show Detailed Node Information

XPath: /client:LoanCompareProcessResponse/clie

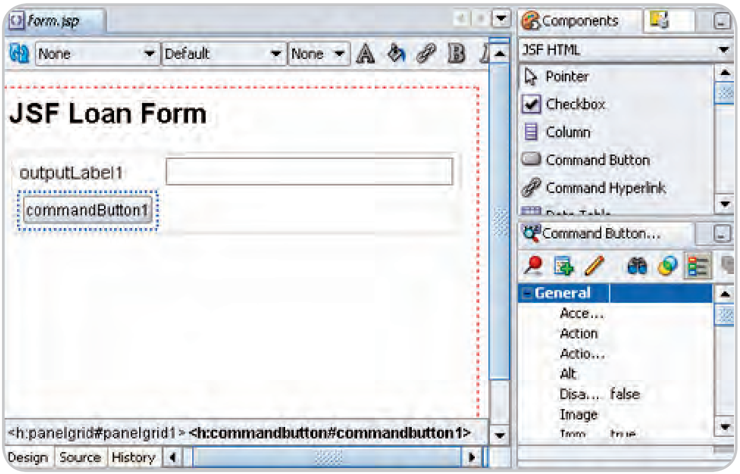
Help | OK | Cancel

Figure 13 The Oracle BPEL process manager's Web console

```
public static void main(String[] args) {
  try {
    proxy.CheaploansWSSoapHttpPortClient myPort = new proxy.
CheaploansWSSoapHttpPortClient();
    System.out.println("calling " + myPort.getEndpoint());
    // Add your own code here
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

You can issue a test call to the Web Service by adding the line:

```
System.out.println(myPort.getLoanrate("123-45-6789"));
```



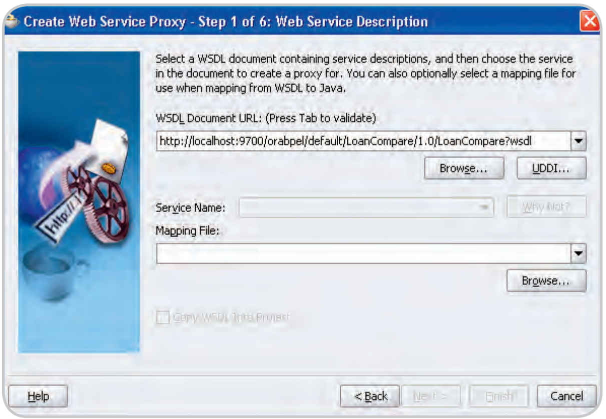
JSF Loan Form

outputLabel1

commandButton1

<h:panelgrid#panelgrid1 > <h:commandbutton#commandbutton1>

Figure 14 The JSF loan form

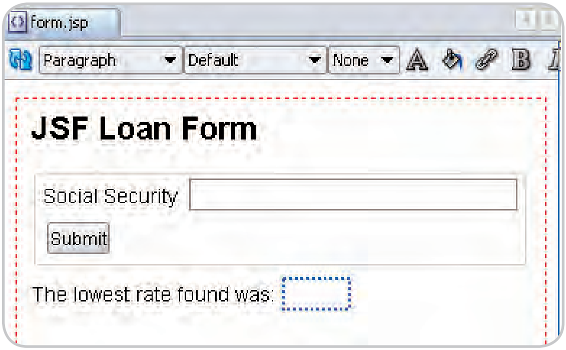

Create Web Service Proxy - Step 1 of 6: Web Service Description

Select a WSDL document containing service descriptions, and then choose the service in the document to create a proxy for. You can also optionally select a mapping file for use when mapping from WSDL to Java.

WSDL Document URL: (Press Tab to validate)

http://localhost:9700/orabpel/default/LoanCompare/1.0/LoanCompare?wsdl

Figure 16 Creating a Web Service proxy for the BPEL Web Service



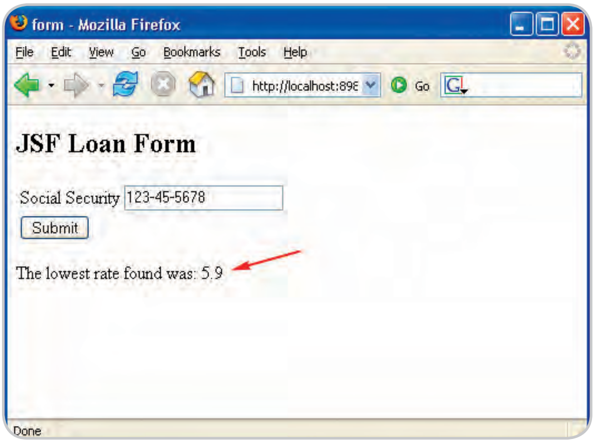
JSF Loan Form

Social Security

Submit

The lowest rate found was:

Figure 15 The JSF loan form



JSF Loan Form

Social Security 123-45-5678

Submit

The lowest rate found was: 5.9

Figure 17 Creating a Web Service proxy for the BPEL Web Service

To the location denoted by the comment "// Add your own code here".

The next step is to modify the action method linked our button with code to call the BPEL Web Service using the Proxy class, "LoanCompareWSSoapHttp-PortClient".

Jumping back into the backing bean code, "Form.java", we will edit the action method, "" as follows:

```
public String commandButton1_action() {
  // Call BPEL Web  service through Proxy Class
  float returnedRate = 0;
  try {
    proxy.CheaploansWSSoapHttpPortClient myPort = new proxy.Cheapl
oansWSSoapHttpPortClient();
    returnedRate = myPort.invokeCompareLoan((String)inputText1.
getValue());
    }
    catch (Exception e) {
        e.printStackTrace();
    };
   // Display the returned value in the outputText field.
   outputText1.setValue(returnedRate);

    return "Success";
}
```

Now when the button is clicked at runtime, the proxy class, **myPort**, will be instantiated and the **invokeCompare-Loan( )** method is invoked and the Social Security value passed from the form is supplied as an argument. The BPEL Web service initiates on the BPEL Process Manager server and determines the lowest loan rate and returns it into the returnedRate float variable. This variable is then applied to the Output Text (outputText1) Faces UI component and a "success" String is returned.

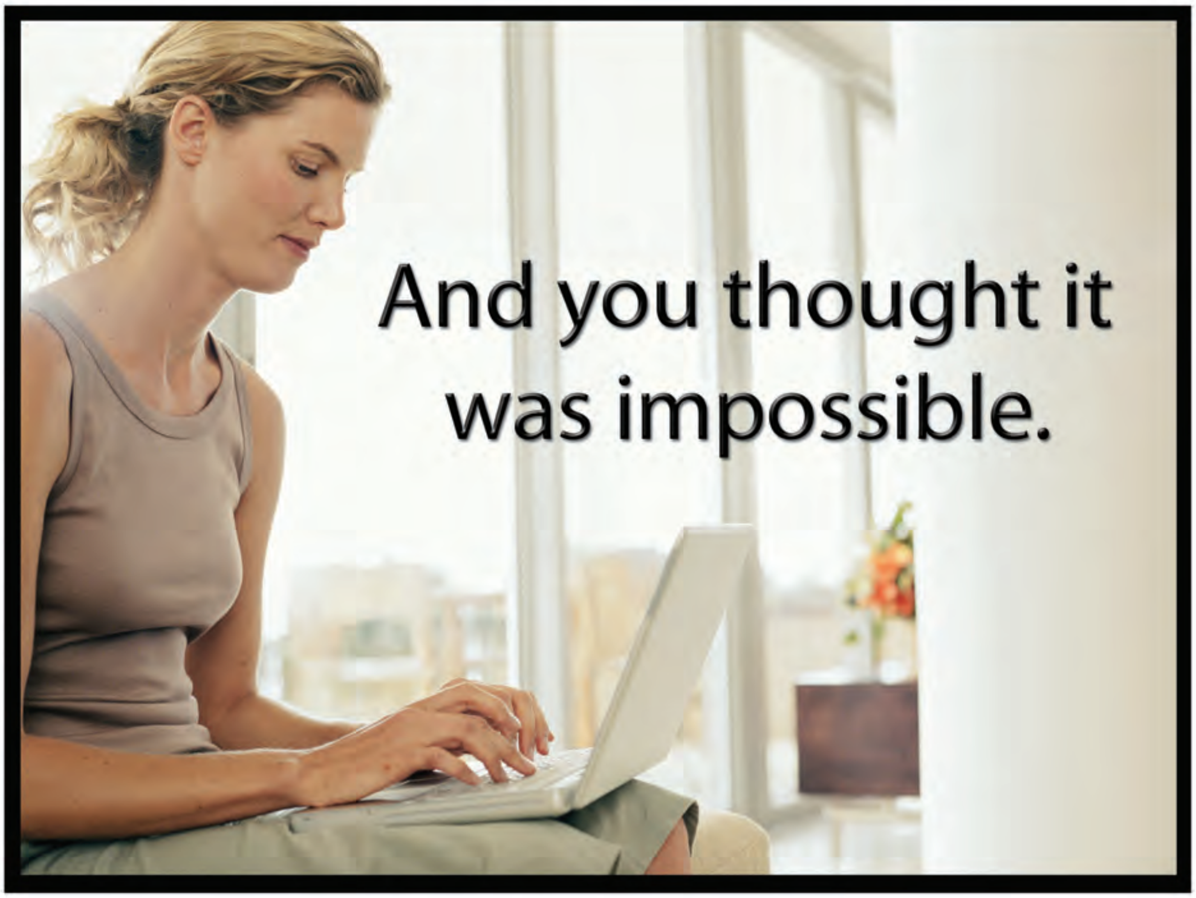
When the JSF application is run, the form will appear in a browser, and when a Social Security number is entered and the button is clicked, the BPEL Process Web Service would be called and the lowest returned rate determined by the BPEL process will be displayed in the JSF form.

Summary

With SOA comes the unique possibility to easily couple vastly different sets of services together into single, cohesive business service. When accompanied with a JSF-based user interface, a BPEL process can then offer its services through a rich user interface.

As our example has shown, building a composite SOA application with an included JSF-based user interface is actually a straightforward process. When a tool such as Oracle JDeveloper 10g is used, it can easily be accomplished in an entirely visual manner.

In five days you'll be programming in JSF, Spring and Hibernate all by yourself.



ArcMind

Public and on-site courses. Online training coming soon.

Phone: 520-290-6588 info@arc-mind.com www.arc-mind.com

Creating Secure Web Applications with Struts

by Alex Smolen

Security principles and how Struts applies

Imagine building a house starting with only a pile of timber and a lump of iron, or making a bowl of spaghetti from a sack of wheat and a bag of tomatoes. The importance of having the right materials makes the idea of building products from scratch seem absurd. Similarly, any software project that doesn't take advantage of the numerous frameworks available for any manner of development activity could be wasting valuable resources and ignoring established best practices.

The advantages of using frameworks are obvious to any developer who has implemented a complex, bug-ridden solution to a design problem that's already been elegantly addressed by a framework. And perhaps the most difficult design problems to get right are those concerning security. With the popularity of Web applications and services on the rise, there has been an increasing move to standardize security-critical tasks, such as authentication and session management, in the container or framework. This way, developers can focus on implementing business processes, rather than specialized tasks like cryptographic algorithms or pseudo-random number generation.

This article will focus on developing secure Web applications with the popular Java framework Struts. It will detail a set of best practices using the included security mechanisms. The first section will provide an overview of both Struts and Web application security as a context for discussion. Each subsequent section will focus on a specific security principle and discuss how Struts can be leveraged to address it.

Alex Smolen is the Security Solution manager at Parasoft, where he oversees the development of the Security Solution and helps companies create strategies to secure their applications. Alex has eight years of experience as a software engineer and is a contributing member of OWASP (the Open Web Application Security Project) and head of the Web Services Project. Alex's speaking engagements include Better Software Conference 2005 and Enterprise Architect Summit 2005 where he spoke on emerging trends in enterprise security. He was graduated from the University of California, Berkeley, with a BS in electrical engineering and computer science.

Alexjdj@parasoft.com

Struts

Struts is a very popular framework for Java Web applications large and small because of the numerous advantages it offers developers. The main goal of the Struts framework is to enforce a

MVC-style (Model-View-Controller) architecture, which means that there is a separation of concerns among different architectural components: the *model* is the representation of the logic, the *view* is in charge of displaying data to the user, and the *controller* is responsible for providing the user with a way to interact with the application and affect the model. A simple analogy for this is a video game, where you have a game console (the model), the television or monitor (the view), and a controller (quite appropriately, the controller). This architectural pattern promotes reuse and stability by reducing the effects of code changes (since the implementation of each component is agnostic to the implementation of the others and the model is isolated from the user).

Although it is approximately an implementation of the MVC pattern, Struts is more accurately based on the "Model 2" architecture specific to the Java servlet technology. Rather than having users access the JSPs directly, Struts applications have a "front controller" servlet that's the initial target of all requests and decides how to process requests and route users. Struts also has two different frameworks, the original (Struts Action Framework) and one based on JSF (Struts Shale). For the purposes of this article, we'll only consider the original framework.

Web Application Security

Web applications (such as those built on Struts) rely on users being able to access potentially sensitive information from all over the world over disparate untrusted networks. It's not exactly a surprise or a secret that many non-secure Web applications have been exploited, making front-page news and causing an enormous amount of problems for the organizations responsible. Application security attacks like SQL injection, cross-site scripting, session hijacking, and cookie poisoning are now mainstays in the toolkit of any hacker worth his salt, and it's becoming increasingly obvious that developers have to put more of an emphasis on security.

Organizations like OWASP (Open Web Application Security Project) and WASC (Web Application Security Consortium) have assembled a great deal of information on how to avoid common pitfalls and create more secure Web applications. These and other resources are invaluable for learning about Web application security, and this article complements them as a guide for best practices in Struts applications with respect to security. Here we'll focus on four specific types of security concerns and how they relate to Struts.

Struts & Input Validation

Input validation refers to the practice of verifying that input from an untrusted source is acceptable and safe to use. This has a significant security impact because malformed data submitted by a malicious user is the direct cause of numerous exploits (including SQL injection and cross-site scripting) and generally causes an application to behave unexpectedly and outside of its security design.

The Struts Validator plug-in lets you cleanly encapsulate all of your validation logic in XML configuration files instead of Java code. The Validator plug-in assists developers by standardizing common types of validations, preventing validation logic duplication, and being easier to verify and change (no recompilation is required). Two things to consider when using the Validator plug-in:

1. There's a mechanism to validate the code of the ActionForm (*org.apache.struts.action.ActionForm*, the Java class in the controller responsible for handling user data). However, this doesn't offer the advantages described above and won't be discussed here.
2. Any business-level validation should be performed in the model, and the controller should be limited to semantic validation (correct length, type, acceptable character set). For instance, in the Validator plug-in you might ensure a credit card number is the right format, but you'd ensure it's a valid card in the business logic.

Here's how the Validator plug-in works:

1. User input is encapsulated in one of the ValidatorForm classes (which extend ActionForm classes):

```
public class UserValidatorForm extends org.apache.
struts.validator.ValidatorForm {
 public String firstName;
 public String lastName;
 public String phoneNumber;
 public String userId;
 …
}
```

2. Validation functions (several standard ones come pre-baked) are defined in validator-rules.xml. This rule calls a validation method from a custom class:

```
<validator name="userId"
   classname="com.jdjexample.validator.UserIdValidator"
     method="validateUserId"
   methodParams="java.lang.Object,
          org.apache.commons.validator.ValidatorAction,
          org.apache.commons.validator.Field,
          org.apache.struts.action.ActionErrors,
          javax.servlet.http.HttpServletRequest"
      msg="errors.userid">
…
</validator>
```

3. Validation.xml maps which fields have to be validated by which rules:

```
<form-validation>
  <formset>
    <form name="userForm">
      <field property="firstName" depends="required">
          <arg0 key="firstName.displayName"/>
      </field>
      <field property="lastName" depends="required ">
          <arg0 key="lastName.displayName"/>
      </field>
      <field property="phoneNumber" depends="required,
      mask">
      <arg0 key="phoneNumber.mask"/>
      <var>
            <var-name>mask</var-name>
            <var-value>
            ^\D?(\d{3})\D?\D?(\d{3})\D?(\d{4})$
            </var-value>
      </var>
    </field>
    </field>
            <field property="userId" depends="required,
            userId">
            <arg0 key="phoneNumber.mask"/>
     </field>
    </form>
  </formset>
</form-validation>
```

So how can we use the Validator plug-in to improve security?

1. ***Think "White List"*** - Most common attacks based on input validation flaws leverage special "meta-characters" to inject user-modifiable data into places it shouldn't be. A typical security mistake developers make is to try to recognize, clean, or escape known "bad" values rather than accept only "good" values. This becomes a problem when a new "bad" value is discovered. With the enormous number of different locales, encodings, syntaxes, and variations thereof, it's likely this strategy will fail, or at least be difficult to maintain. Struts Validator plug-in lets users verify that input data conforms to strict requirements and is well formed. The validation routines Struts uses can leverage regular expressions (with the mask field) to ensure that only safe strings are accepted. Sometimes, it may be necessary to accept some malicious characters, but it's important to handle them very carefully. Furthermore, data can be verified to ensure that it's within reasonable length bounds to avoid denial-of-service-style attacks.
2. ***Be Consistent*** - Using the Validator plug-in, it's significantly easier to verify that there are no holes in the input validation mechanism. This means that all the data from the user should be in a ValidatorForm class and all fields should be specified in the validation framework. This can be checked simply by matching XML elements and it's much easier to discern than having widespread, inconsistent input validation schemes.
3. ***Be Strict*** - Furthermore, there are some cases where it's preferable to let the user select a few values from a finite list than to let him enter arbitrary data (for example, in cases of a yes/no, multiple choice, or filename). When designing input forms, it's good practice to constrain the possible values as narrowly as possible and reinforce these constraints through validation.

Struts and Access Control

One of the main principles of security is access control (also called authorization) – making decisions on who can and can't do what. Operating systems, file systems, databases, and other systems that must enforce security requirements all contain some sort of access control system to prevent malicious and/or inappropriate usage or resources. In Java Web applications using JAAS (Java Authentication and Authorization Service) and similar mechanisms, there are two different approaches for enforcing access control: declarative and programmatic. Declarative access control means specifying *users*, *roles,* and *security constraints* in configuration files, while programmatic access control declares and implements this logic in the code. The decision to use either approach depends on requirements, but generally declarative access control is easier to implement but inflexible, while programmatic access control requires custom coding but is largely customizable.

The Struts framework, as a whole, is built to be extensible, and this is one of its major strengths. Whether an application uses declarative or programmatic access control, Struts has built-in support and extension points for fulfilling access control requirements.

Declarative Access Control

The Struts framework has a role-based access control mechanism for securing actions declaratively. In struts-config.xml, each action has a roles attribute, which accepts a comma-separated list of available roles:

```
<action path="/AdminLogin"
        forward="/admin_login.jsp"
        roles="administrator"/>
```

This allows for a more natural approach to access control based on the named Action classes rather than URL patterns. One important thing to note is that this doesn't secure JSPs, HTML, or other non-Action resources. If all JSPs are only accessible through the controller (which is a Struts/MVC best practice), a recommended countermeasure is to put all JSPs in the WEB-INF folder so they aren't directly available to users. However, this requires that the container stipulates that files in the WEB-INF folder aren't accessible to Web users. Otherwise, JSPs can be protected by standard security constraints in the configuration file.

Programmatic Access Control

The Struts framework offers several different extension points where programmatic access control can be implemented. Specifically, the request processor has a processRoles() method that can be extended to do custom programmatic access control for all requests to the application:

```
protected boolean processRoles( HttpServletRequest
request,
                HttpServletResponse response,
                ActionMapping mapping )
     throws IOException, ServletException {
     …
     if (request.isUserInRole(roleName)) {
     …
}
```

There are other places that access control could occur, including Servlet Filters and ActionServlets, which will be discussed in the struts and Cross-Cutting Security Concerns section.

Additionally, Struts has tag libraries to support programmatic "content-level" access control in JSP pages. Using the following tag, content in JSP pages can be secured according to roles:

```
<logic:present role="administrator"/>
<a href="admin_page.jsp">Administrator Home</a>
</logic:present>
```

In almost all cases, it's recommended that you use the security mechanisms built into the containers and frameworks rather than write them from scratch. Struts provides a granular, integrated set of role-based access control functionalities that help teams design and implement secure applications.

Struts and Error Handling

Application hackers are very resourceful; any piece of information uncovered during an attack will be noted, analyzed, and leveraged for future attacks. One of the most useful (read compromising) pieces of data that a hacker can access is an error messages that says too much about the internal workings of the application. This lets the attacker make informed decisions about the effectiveness of the attack, how to proceed in further attacks, and what resources are accessible by the application.

Struts has a very powerful and extensible exception-handling mechanism built-in. It's very important, for security's sake, to use this carefully to improve the user experience without letting information leak to malicious attackers. Specifically, it's important to have robust error handling, so that expected errors (e.g., bad logins) are handled as appropriately as unexpected errors (NullPointerExceptions) and that there are good logging facilities to verify security-relevant events.

For Java Web applications in general, there are several ways to catch exceptions.

- ***Programmatically***:

1. Use try/catch blocks.

```
try {
 login(username, password);
} catch (BadLoginException) {
…
}
```

2. Use the JSP page directive "errorPage".

```
<%@ page errorPage="bad_login.jsp" %>
```

- ***Declaratively***:

3. In Web.xml, declare error-page and exception type forwards.

```
<error-page>
 <exception-type>
   com.jdjexample.BadLoginException
 </exception-type>
 <location>/bad_login.jsp</location>
</error-page>
```

The Struts framework builds on these methods.

- ***Programmatically***:

1. Use Action Errors to wrap exceptions.

```
try {
 login(username, password);
} catch (BadLoginException) {
   ActionErrors errors = new ActionErrors();
   errors.add("login", new ActionError("bad.
   login"));
   return errors;
}
```

2. Implement processExceptions in RequestProcessor.

```
protected ActionForward processException(HttpServ-
letRequest request,
       HttpServletResponse response,
       Exception exception,
       ActionForm form,
       ActionMapping mapping)
  throws IOException, ServletException {
    if (exception instanceof BadLoginException) {
…
}
```

- ***Declaratively***:

1. Specify action and global exception forwards.

```
<global-exceptions>
    <exception
      key="bad.login"
      type="com.jdjexample.BadLoginException"
      handler="com.jdjexample.BadLoginException-
      Handler"/>
</global-exceptions>
```

Error handling is a vital part of any application robustness and security, and each application must discern requirements about how to deal with errors. Often, these requirements will include logging security relevant errors to provide forensics information after attacks and comply with traceability stipulations in compliance guidelines. Struts lets errors be handled cleanly both in and outside of the code, with multiple points to implement custom code for unique requirements.

Struts and Cross-Cutting Security Concerns

Often, applications have security requirements that affect all resources, such as guarantees about privacy, or auditing requirements. Rather than try to apply the correct logic in each individual component, it's often helpful to have an "intercepting" class that funnels all the invocations and performs a common set of routines. This way, changes to overall functionality can be isolated to a small area. Struts has several different places where cross-cutting security concerns can be implemented, each with its own particular benefits.

ActionServlet

ActionServlet is the primary servlet in the Struts framework, and handles all requests to Struts Actions. In Struts 1.0, ActionServlet was the class to extend to change the handling of all requests to actions; however, much of the functionality of ActionServlet was refactored to the RequestProcessor class. This lets sub-application modules handle requests differently and promotes abstraction between the processing of raw HTTP requests and Struts-specific request processing, which should happen in RequestProcessor. Typically ActionServlet should only be extended if other extension points can't provide the functionality needed.

RequestProcessor

The RequestProcessor class is where most of the per-request processing takes place. There are many different methods that can be overridden here, but some have very direct security relevance:

1. ***processRoles()***: This method should be overridden for any action-level access control that must be programmatic (for instance, if the application doesn't use JAAS).
2. ***processNoCache()***: This method should be overridden to tag the response as "no-cache" to avoid security vulnerabilities associated with locally caching content.
3. ***processException()***: This method should do any special exception handling that can't be done declaratively.
4. ***processValidate()***: This method should do any special validation that can't be done declaratively.

Obviously, more security concerns can be addressed by extending the Request-Processor. This is where we see the Struts framework shine, since it lets us easily implement maintainable cross-cutting security mechanisms in our Web applications. However, it's important to note that any request that doesn't go through the action/controller servlet (e.g., for a static HTML page, JSP, image/multimedia file, etc.) isn't intercepted by the RequestProcessor and won't be subject to these controls (see Servlet Filter if this is a concern).

Base Action

By defining a base Action class that all other actions extend, security concerns can be addressed in the perform method and all other action classes would call the super. The disadvantage here is that all actions must extend this action, but this is a viable alternative to using a request processor.

Servlet Filter

The Filter class (javax.servlet.Filter) isn't Struts-related, but can be used to intercept and handle requests. Servlet Filters have the advantage of intercepting all (or some subset of) requests, not just those based on actions; if this functionality is necessary, then filters may be appropriate. An interesting use of Servlet Filters with respect to security is the security filter project (securityfilter.sourceforge.net) that emulates container-managed security, but alleviates problems with flexibility and portability. However, extending Request-Processor is typically a cleaner solution because it centralizes the application logic in the Struts framework and requires less maintenance than using two separate libraries.

Conclusion

Application security is a problem that affects projects from the business goals down to the code implementation. Despite the increasing amount of knowledge and expertise available to the development community, many homegrown security solutions are used in place of established best practices. The Struts framework offers developers easy and effective security mechanisms that, when applied correctly, can significantly improve an application's overall security. While this article discusses some common issues, it's always important to consider unique security requirements and leverage best practices to suit the needs of the project as well as any included frameworks.

Bridge the Gap Between Windows and Java

Give your users the freedom to use Microsoft Windows based reporting, analysis, and development applications against your Java data source

OpenAccess™ SDK lets you quickly implement a custom ODBC driver with full SQL support for any data source or application.

Key OpenAccess™ SDK Features & Benefits

- Write the ODBC driver with SQL support in just weeks - 99% of the required code is contained in the SDK.
- Use Java to implement the data source specific code.
- Guaranteed to work with ODBC compliant applications – hundreds of organizations have used OpenAccess™ to set their data free.
- Fully Unicode enabled for globalization.
- When required easily add JDBC, OLE DB, or .NET drivers.
- Designed from ground up for reliability, scalability, and portability.

To learn more about OpenAccess™ SDK for Java or to try it out, go to **www.OpenAccessSoftware.com/JDJ** or call **1.888.805.ODBC**

OpenAccess SOFTWARE

OpenAccess Software, Inc. 2001 Gateway Place, Suite 100 San Jose, CA 95110 www.OpenAccessSoftware.com

Copyright © 2006 OpenAccess Software, Inc. All rights reserved. OpenAccess is a trademark of OpenAccess Software, Inc. All other brand names are trademarks of their respective holders.

KISS + Swing = RAD

by Richard Conway

How to rapidly develop enterprise class Swing applications by keeping things simple

Java is a great language for developing enterprise applications. It's powerful, scalable, robust, secure, and typically very complex. As a software developer, I want to solve business problems, not spend man-months building the plumbing for my applications. This article will demonstrate how you can speed up the development and simplify the maintenance of enterprise-class Swing applications by keeping things simple. We'll look at ways to reduce the complexity of your application and the amount of custom code written for it. By limiting the complexity and the amount of plumbing code required, you'll develop more quickly, the application will be easier to maintain, and you can focus on the business logic that provides value to the customer.

In particular, I'll show you how to address three of the primary issues facing developers today:

1. Persistence and object relational impedance mismatch
2. Why too many technologies make development and maintenance difficult
3. What to do when there's more application "plumbing" code than custom business logic

What Is Object Relational Impedance Mismatch and Why It's a Problem

Relational databases are the dominant database technology in the market today and they are well matched for use with procedural programming languages. However, with the move to object-based development languages (C++, Java, Delphi, etc.) it quickly became apparent that objects don't always map easily to relational tables and vice versa. This problem has become known as the object relational or impedance mismatch.

According to the Progress Software Web site, "An R.B. Webber study concluded that coding and configuring object relational (O-R) data access typically accounts for 30% to 40% of total project effort."

There are development and performance repercussions due to this mismatch. Developers must spend time writing code that breaks objects into pieces that can be saved to and re-assembled from relational tables and determining the best way to map objects to relational tables becomes more difficult as the objects grow in complexity. There is also a certain amount of overhead due to this assembly and breakdown of objects that impacts performance. Finally, queries can take longer since the application has to perform multi-table joins when retrieving complex objects from the database rather than just opening the object.

Historically programmers have dealt with this by writing a data access layer that manages the reading/object assembly from the database and object disassembly/writing to the database. Recently there's been an explosion in the number of XML-based mapping tools that attempt to replace the native language data access code with frameworks that define the mapping using XML. The problem with this approach is that while it provides a mechanism for persisting and retrieving objects, letting the developer to deal with objects exclusively, it requires quite a bit of additional work (defining and maintaining the XML mappings), incurs a performance hit when the objects are read from and saved to the database, and often requires the user to query the database with non-standard object query languages that aren't as mature and powerful as SQL.

Eliminating the Object-Relational Impedance Mismatch

One way to eliminate the object-relational impedance mismatch altogether is to use an object database for your application's persistence layer. Object databases are now mature, robust, and fast. More importantly, they provide an easy mechanism for persisting and retrieving objects without the overhead of a mapping framework. This means faster development, better performance, and less code to maintain.

Richard Conway is a software developer and technology consultant with more than 15 years of technology, project management, and information services experience. He started focusing more on Swing-based development at the beginning of 2005 and has just finished a Swing-based client/server asset management project. He lives in Miami with his wife Patricia, is currently working on an EMR application, and plays sand volleyball in his spare time.

reconway@egrok.com

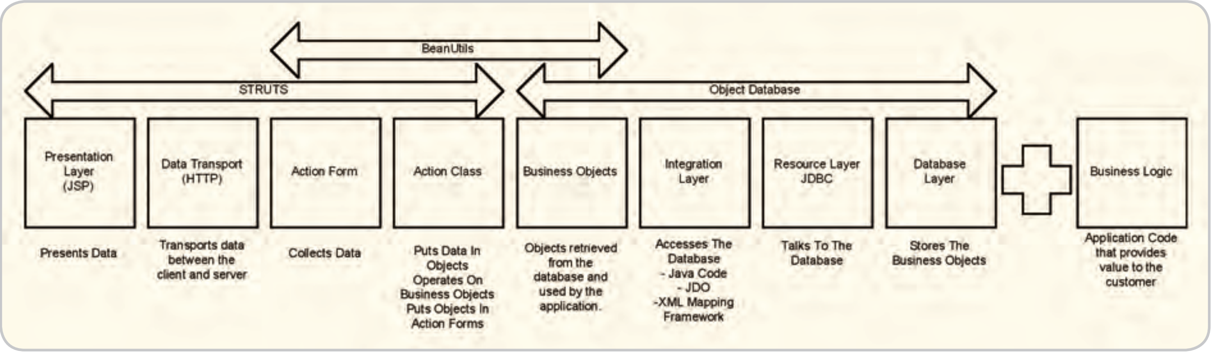
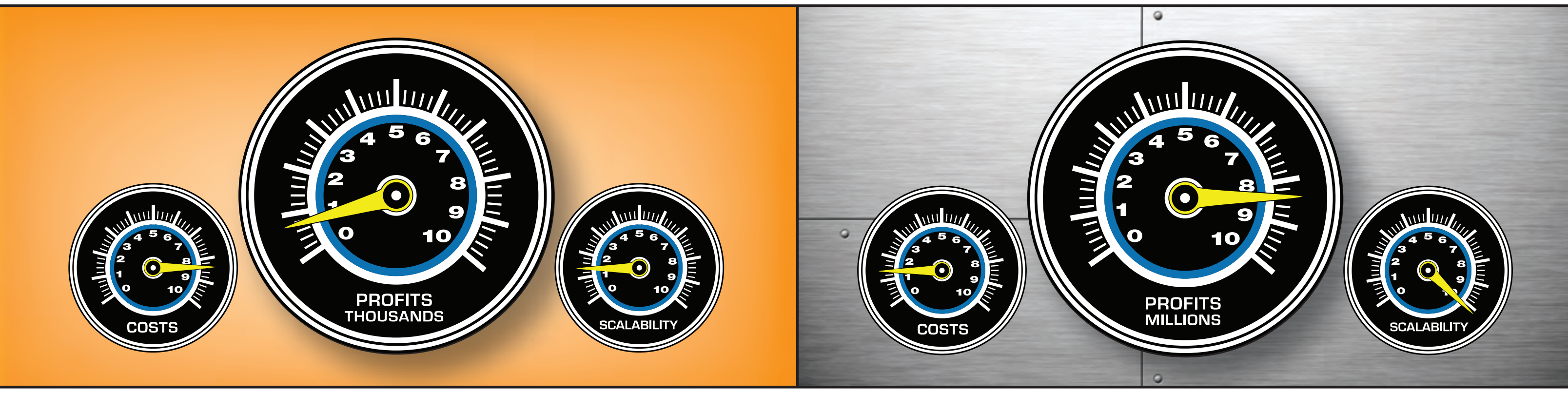

BeanUtils
STRUTS
Object Database
Presentation Layer (JSP)
Data Transport (HTTP)
Action Form
Action Class
Business Objects
Integration Layer
Resource Layer JDBC
Database Layer
Business Logic
Presents Data
Transports data between the client and server
Collects Data
Puts Data In Objects Operates On Business Objects Puts Objects In Action Forms
Objects retrieved from the database and used by the application.
Accesses The Database - Java Code - JDO -XML Mapping Framework
Talks To The Database
Stores The Business Objects
Application Code that provides value to the customer

Figure 1 Web/Struts software stack

A slapdash approach won't get you there...



ExtenXLS will.

"ExtenXLS makes it possible to re-use spreadsheets representing a tremendous investment in time and business logic and to 'connect' those spreadsheets to various SQL data sources to provide access to usable data in a way never before possible."

- Project Leader, AT&T

"We use ExtenXLS for 'pushing' accounting data into an Excel template. A key customer wanted web access to Excel format reports. We tried .NET but had problems with preserving charts & formatting. ExtenXLS provides a flexible, cost effective solution that gives us the ability to stay ahead of user requirements."

- Project Leader, John J. McMullen Company

It's What's behind the Dashboard that Counts!

Slapping a dashboard onto a static spreadsheet file may **look** good, but it's like installing a flashy dashboard on an outdated clunker. Install that same front-end on ExtenXLS — **a scalable, server-based spreadsheet engine** — and watch your flashy dashboard come alive.

Now with Round Trip Reporting™

Reuse your existing spreadsheets as a data-entry tool. Modified data is extracted from your spreadsheets and turned into Java Data Objects for reuse in all your programs.

Reach New Heights without the Learning Curve

With the familiar look and feel of Excel™, ExtenXLS eliminates the learning curve imposed on your end-users by other reporting tools. ExtenXLS unlocks the business logic stored in static spreadsheets throughout your organization, saving time and money.

Escape the Gravitational Pull of Obsolete Reporting!

Output to customized HTML for maximum compatibility or to XML for further processing. Keep your users happy with native Excel™ output which preserves the VB macros, images, charts, and other features that transform live data into actionable reports. You can even embed a familiar spreadsheet component in your Swing applications.

Don't rely on a slapdash approach for your mission-critical reporting needs.
Put a rocket under the hood and achieve escape velocity.

Visit the mother ship at: www.extentech.com/jdj and take your free evaluation copy into orbit for a test flight.

EXTENXLS4™ JAVA | XLS REPORTING TOOLKIT

extentech™ Call Us: 415-759-5292

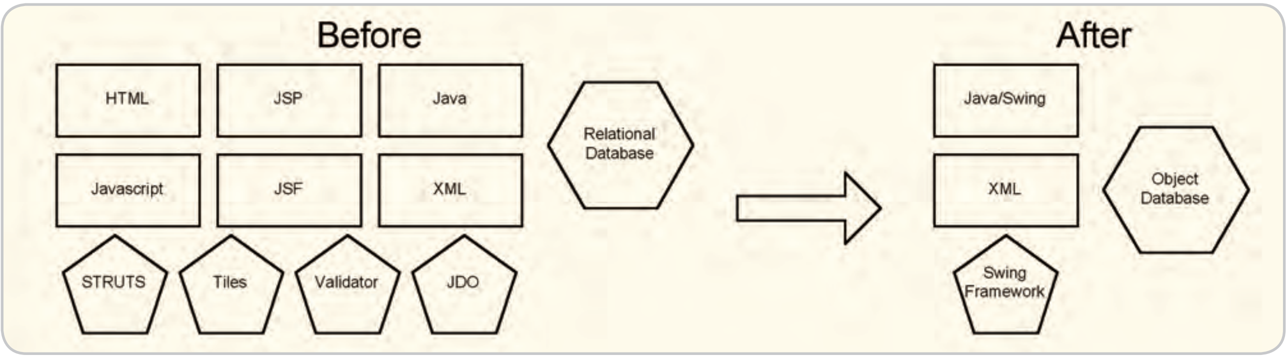


Figure 2 The software simplification process

You can browse a list of object-oriented databases at Service-Architecture.com. As with any technology, every vendor's implementation provides slightly different features and functionalities. It's important to review each one to find the one that best fits your project's needs. Next, I'll demonstrate how using an object database instead of a relational one can dramatically simplify and speed up the development of your entire application.

Example # 1: Defining and Accessing Objects Using An Object-Oriented Database

For these examples, I'll use InterSystems Corporation's Caché Database. The databases from db4objects, Matisse, and Progress also provide Java interfaces and transparent object persistence. However, features and implementations are specific to each vendor's database.

With InterSystems Caché you create your database by defining the objects that it will contain. The objects contain properties that are analogous to standard SQL data types and are mapped to Java datatype equivalents. The objects support aggregation and inheritance *and* can be projected as Java classes that include methods for retrieving, modifying, and persisting the object.

Note that Caché also lets you access your data/objects via an SQL projection, meaning you can access your data in either a relational or object-oriented fashion, whichever is most appropriate for the task at hand.

- ***Aggregation***: When an object is composed of other objects. For example, a car object is comprised of an engine object, a transmission object, four wheel objects, etc.
- ***Inheritance***: An object inherits the attributes and methods of another class (the base or parent class).

Savings # 1: Eliminate the Code and Performance Overhead of Mapping

By using an object-oriented database, you eliminate the need to create and maintain a mapping layer or data access layer. Not only does this speed up development, but you have less code to maintain and one less technology to learn. Look at Figure 1 to see what this means for the typical struts-based Web application.

The object database replaces four pieces of the application software stack. In addition, you will improve performance by eliminating the overhead of any XML mapping layer.

Just so you have an idea of some of the effort being eliminated, here's an example of a JDO XML map for a very simple object with only two fields. When using a mapping framework, you would normally create one of these for each object class in your database. By using an object database, you eliminate this step.

```
<?xml version="1.0"?>
<!DOCTYPE mapping PUBLIC "-//EXOLAB/Castor
Object Mapping DTD Version 1.0//EN"
              "http://castor.org/mapping.dtd">
<mapping>
  <class name="myapp.ProductGroup"
identity="id">
      <description>Product group</description>
        <map-to table="prod_group" xml="group"
/>
        <field name="id" type="integer" >
      <sql name="id" type="integer"/>
    </field>
        <field name="name" type="string">
          <sql name="name" type="char" />
        </field>
     </class>
</mapping>
```

Example: Object Definition That Demonstrates Aggregation and Inheritance

Now I'll demonstrate how easy it is to create persistent Java objects using Caché by creating a very simple employee database.

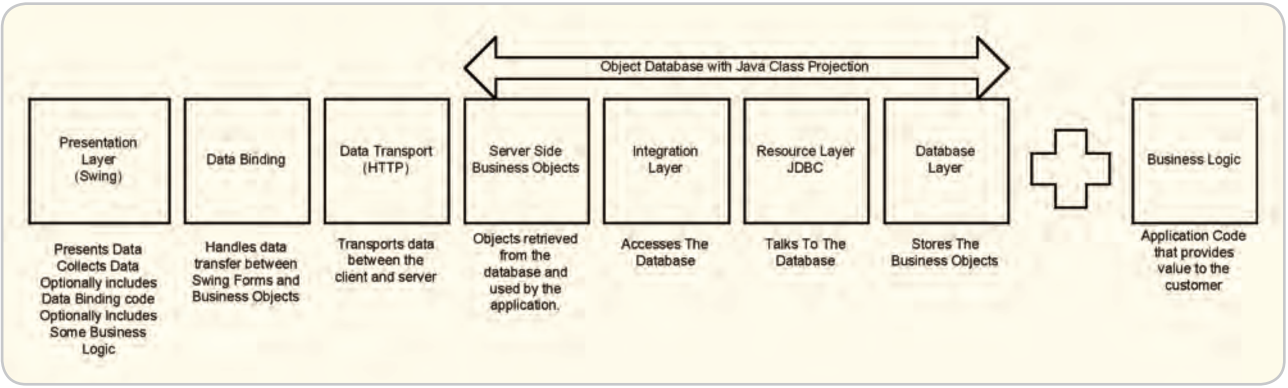

Object Database with Java Class Projection
Presentation Layer (Swing)
Data Binding
Data Transport (HTTP)
Server Side Business Objects
Integration Layer
Resource Layer JDBC
Database Layer
Business Logic
Presents Data Collects Data Optionally includes Data Binding code Optionally Includes Some Business Logic
Handles data transfer between Swing Forms and Business Objects
Transports data between the client and server
Objects retrieved from the database and used by the application.
Accesses The Database
Talks To The Database
Stores The Business Objects
Application Code that provides value to the customer

Figure 3 Framework Coverage: Object Database

Caché comes with a GUI tool called Studio that's used to simplify the definition of classes. You define classes in much the same way you would define relational tables – except when you're done, you can access the objects directly without writing data access code or using an object-relational mapping layer. Much of the class definition text can be generated by wizards in the application – but it can also be manually coded if desired. We'll define four classes including:

1. An address class that can be reused as a property template in persistent classes
2. A person class that represents a simple person object
3. An employee class that extends the person class by adding additional properties

- ***Persistent Class***: Persistent objects can be stored in the database.
- ***Serial Class***: Serial objects can be embedded in persistent objects to create complex, reusable data definitions – such as addresses.

The Address Class

Here's the definition for the address class. Note that it extends a %Serial-Object so it can't be persisted by itself and must be included in a persistent class. Serial classes are great for simplifying the definitions of complex objects. We'll use this one to simplify the design of the Person and Employee classes.

```
/// This is a sample embeddable class representing an address.
Class base.Address Extends %SerialObject [ ClassType = serial, ProcedureBlock ]
{

Projection JavaClient As %Projection.Java;

/// Specify the proper Java Package
Parameter JAVAPACKAGE = "com.egrok.db.base";

/// The street address.
Property Street As %String(MAXLEN = 80);

/// The city name.
Property City As %String(MAXLEN = 80);

/// The 2-letter state abbreviation.
Property State As %String(MAXLEN = 2);

/// The 5-digit U.S. Zone Improvement Plan (ZIP) code.
Property Zip As %String(MAXLEN = 5);

}
```

The Person Class

Note that the Person class is being projected as a Java class in the package "com.egrok.db.base" and that the Person class has two properties that are derived from the serial address class: HomeAddress and WorkAddress. When you access these properties from the Java class, you'll use standard dot notation as follows:

```
Class base.Person Extends %Persistent [ ClassType = persistent, ProcedureBlock ]
{

Projection JavaClient As %Projection.Java;

/// Specify the proper Java Package
Parameter JAVAPACKAGE = "com.egrok.db.base";

Property Name As %String [ Required ];

Property LastName As %String;

Property HomeAddress As base.Address;

Property WorkAddress As base.Address;

Property Notes As %String(MAXLEN = 1000);
Property DateCreated As %TimeStamp;

/// Person is registered with the mailing list
Property IsRegistered As %Boolean;
}
```

```
person.getHomeAddress().getStreet();
```

Swing on the Desktop

There's been a lot of noise about Swing making a comeback on the desktop lately. Performance concerns are a thing of the past and new projects like the Java Desktop Integration Components (JDIC), Java Desktop Network Components (JDNC), Data Binding, and SwingX are adding much needed capabilities to Swing that improve the user experience and simplify development by providing much needed functionality in easy to use libraries. This, continuing our theme of simplification, saves you from writing and maintaining your own code.

Build interactive diagrams easier than you ever imagined

Create custom interactive diagrams, network editors, workflows, flowcharts and design tools. For web servers or local applications. It's easy-to-use and very flexible. We're the first developer of diagramming components and still the best. Find out for yourself: download our FREE fully functional evaluation kit, with full support at www.nwoods.com.

FREE Download With Full Support!

GoDiagram

Interactive diagram components

www.nwoods.com

The Employee Class

```
Class base.Employee Extends base.Person [
ClassType = persistent, ProcedureBlock ]
{

Projection JavaClient As %Projection.Java;

/// Specify the proper Java Package
Parameter JAVAPACKAGE = “com.egrok.db.base”;

/// Specify the required property username
Property UserName As %String [ Required ];

/// Specify the username property must be unique
Index UserNameIndex On UserName [ Unique ];

Property Password As %String [ Required ];

/// The employee’s current work status. Used when
assigning jobs dynamically
Property WorkStatus As %String(VALUELIST = “,Avail
able,Sick,Vacation,UnAvailable”);

/// Defines a one to many relationship between
the Department Object and Employee Objects
Relationship Department As base.Department [
Cardinality = one, Inverse = Employees ];

/// Defines an index on the Department property
Index DepartmentIndex On Department;

/// Specify the Employee’s Manager – Example of
Aggregation
Property Manager As base.Employee;
}
```

Here you can see that the Employee class extends the Person class and thus inherits all the properties and methods associated with the Person class. New properties specific to the Employee class are also added including a one-to-many relationship with the Department class and a reference to another employee, the Manager (an example of aggregation). We won’t go into the details of parent/child and one-to-many relationships here since it’s beyond the scope of this article.

Once again a Java Projection is defined. When you compile your class definitions, Java classes that provide access to the objects in the database will be created in the specified package. Simply by defining your database, you get access to pure Java objects that can be instantiated and persisted without any other code or mapping frameworks.

Tip: As with any Java class, you can extend the Java classes created by Caché (Java Projections) and add custom methods to them. Your application should then access these extended classes. This will insulate your application from most changes made to the class definition and preserve your custom methods, which would be overwritten the next time you compiled the database definition if you modify the Caché projection classes directly.

Example: Instantiating a Persistent Object

Listing 1 shows how to instantiate and access the Employee Java object from your Java code as well as some examples of how to access the data just as you would any other Java object.

That’s it – straightforward Java objects. Just like any other persistence mechanism, you must connect to the database. However, once you’ve done that, you can access the object using standard dot notation. As you can see, you can also access linked objects (department and manager, for example) in the same manner without the need for creating multi-table SQL Joins. And remember – you’ve eliminated the need to define and maintain complex XML mappings for a JDO persistence layer!

Caché Tip: Use Caché’s SQL projection when you are doing complex selection queries and updates to multiple objects at the same time. Use Caché’s Object projection when dealing with a single object or linked objects.

Why Too Many Technologies Make Development and Maintenance Difficult

At this point, you’ve seen how to dramatically simplify and accelerate development on the server side, but you still need to get the data to the client and present it to the end user. Look under the hood of many Web -based applications today and you’ll see a vast array of technologies and frameworks in use. A typical Struts application will require knowledge of HTML, JSP, XML, Java, and perhaps JSF and JavaScript. In addition, you have to learn Struts and possibly Tiles and some sort of Validation framework. (We’ll forget about SQL and JDO for now, since we’ve solved that by using an object database.)

While Struts and Tiles add a layer of complexity, overall they’re an improvement over custom code since they replace custom MVC and Tiling frameworks with proven, tested, and well documented frameworks that are based on industry standards such as Java/JSP/XML. (We’ll explore why frameworks are good in the next section.) However, there are still some serious consequences to using all these technologies:

1. You must have people trained in one or more (preferably more) of these technologies
2. Someone has to understand how each layer interacts with its neighboring layers
3. Each technology/framework will develop at its own pace – possibly introducing versioning and dependency issues with the other technologies

Savings # 2: Reduce the Number of Technologies Used in Development

So, it seems reasonable that by reducing the number of technologies used in an application, you’ll simplify and probably speed up development (see Figure 2). One way to do this is to use Swing for the presentation layer. Swing provides the following benefits:

1. Extremely rich user interface
2. Excellent development and code management tools
3. Easily customizable and extendable
4. Many third-party components and libraries available
5. Reduces the number of technologies you must know to ONE
6. Easy management and distribution with Java Web Start

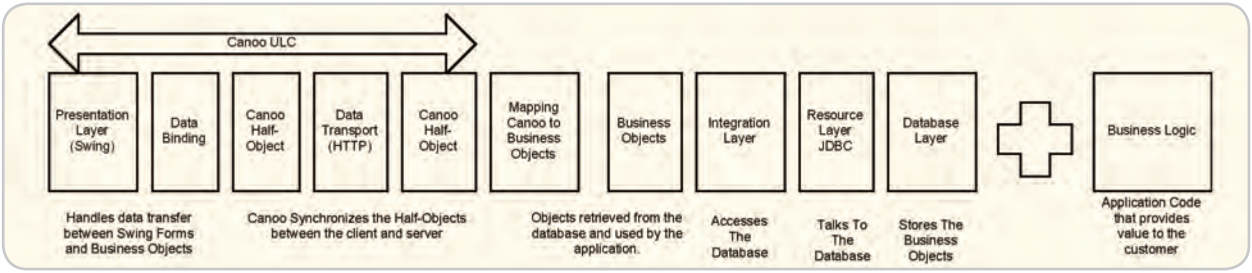

Canoo ULC
Presentation Layer (Swing)
Data Binding
Canoo Half-Object
Data Transport (HTTP)
Canoo Half-Object
Mapping Canoo to Business Objects
Business Objects
Integration Layer
Resource Layer JDBC
Database Layer
Business Logic
Handles data transfer between Swing Forms and Business Objects
Canoo Synchronizes the Half-Objects between the client and server
Objects retrieved from the database and used by the application.
Accesses The Database
Talks To The Database
Stores The Business Objects
Application Code that provides value to the customer

Figure 4 Framework Coverage: Canoo ULC

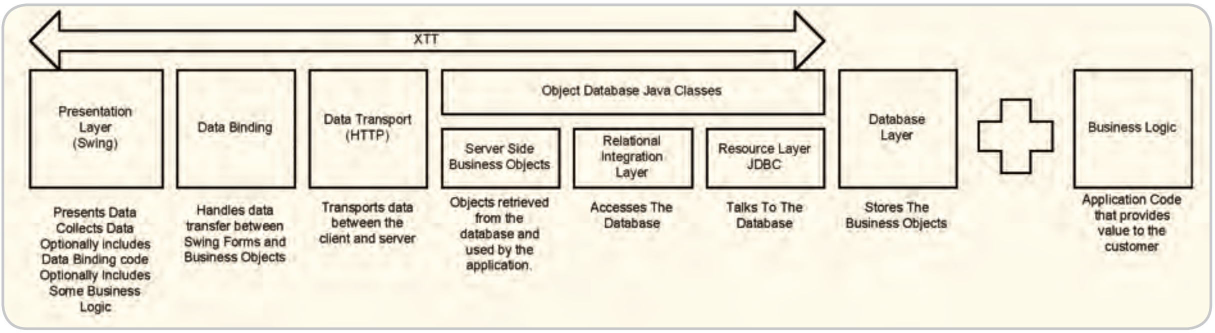
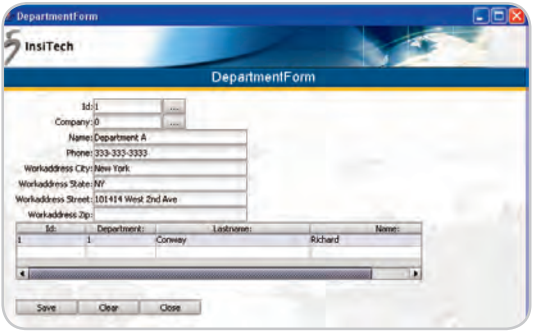

XTT
Presentation Layer (Swing)
Data Binding
Data Transport (HTTP)
Object Database Java Classes
Server Side Business Objects
Relational Integration Layer
Resource Layer JDBC
Database Layer
Business Logic
Presents Data Collects Data Optionally includes Data Binding code Optionally includes Some Business Logic
Handles data transfer between Swing Forms and Business Objects
Transports data between the client and server
Objects retrieved from the database and used by the application
Accesses The Database
Talks To The Database
Stores The Business Objects
Application Code that provides value to the customer

Figure 5 Framework Coverage: Insitech XTT



Id:	1
Company:	0
Name:	Department A
Phone:	333-333-3333
Workaddress City:	New York
Workaddress State:	NY
Workaddress Street:	101414 West 2nd Ave
Workaddress Zip:	

Id:	Department:	Lastname:	Name:
1	1	Conway	Richard

Figure 6 DepartmentForm

By using Swing instead of HTML, JSP, JSF, XML, and JavaScript, you've dramatically simplified your development effort. Code management becomes a breeze. The user interface is much richer and you can do complex validation and processing on the client if you want. Swing provides more efficient communications as well, since, like AJAX, only data is transferred between the client and the server (no UI code). There's only one little problem (well, ok, two) – how will we communicate to the server? And how will we bind the data to the user interface?

The Web stack provides mature and well-understood transport and binding layers via HTTP, JSP, JSF, AJAX, and frameworks Struts. If we are to use Swing to provide a richer user interface to the user, we must also provide a simpler data binding and transport solution that is at least as good as that provided using the typical Web stack. Fortunately the Java space has been maturing in this area and there are a number of options available.

What To Do When There's More Application "Plumbing" Code Than Custom Business Logic

What is plumbing? Communications and binding are part of what I call application plumbing – i.e., required for the application to function, but which aren't seen by the customer and don't provide value to the customer (as the business logic should).

The problem with complex distributed applications (enterprise apps) is that they require quite a bit of plumbing. Frequently you spend more time on the plumbing than on the code that provides value to the customer. This is drudge code. It has to be written, but solves well-understood problems and doesn't require creativity or (typically) differentiate your product from any other product.

Savings #3: Reduce Code by Using Frameworks Based on Open Standards

As previously referenced, just the plumbing needed to save and retrieve information from the database can comprise 30%-40% of your application code. When you add data transport, data binding, and perhaps a Model-View-Controller to the application, your application may consist of 80%-90% plumbing. The goal is to spend as little time on plumbing as possible so you can focus on the business logic that provides value to the customer. What's the solution?

Development Simplification Rule

Use frameworks based on open standards to provide the "plumbing" for your application so you can focus on the custom business logic. In general, the frameworks should:

1. Be based on open standards => Therefore they will be supportable and extendable
2. Replace roll-your-own "plumbing" => Reducing the code you write and maintain
3. Provide advanced features/functionality => Features you may not have the time or expertise to develop in-house.

   Additional things to look for:
4. Wizards => Speed up development by generating code for you

RightOrder, Inc. Santa Clara, CA. Software Engineer
Perform research, design and develop computer software systems. Take a lead role in development team. Be a hands-on developer of proprietary server components & caching subsystems. Must have a B.S. in computer science or equivalent, 7 yrs + SW development of back-end servers & GUI applications in a multi-threaded environment and be fluent in Java, Visual C++, SQL, PL/SQL & SQL Tuning. Must have DBA level experience with Oracle 8/8i/9i/10g . M-F, 40 hr/wk. Send resume to: RightOrder, 404 Saratoga Ave. #200, Santa Clara, CA 95050, Attn: Ginger Burrell

RightOrder, Inc. Santa Clara, CA Senior Software Engineer
Analyze and create software requirements and design, develop and test software systems. Requires excellent design skills & hands-on development experience with a strong background in electronic commerce oriented systems, back-end servers & GUI applications. Requires proficiency at both high and low-level design with a demonstrated ability to plan and develop complex server systems. The candidate must have a BA in computer science or similar degree and preferred MBA in Technology Management. At least 7 years experience programming in OO languages and at least 5 years experience in Java development, Web logic 6.0 on Windows, XML, TogetherJ, JBuilder, InstallAnywhere, Winrunner, Startteam, and J2EE. M-F, 40 hr/wk. Send resume to: RightOrder, 404 Saratoga Ave. #200, Santa Clara, CA 95050, Attn: Ginger Burrell.

RightOrder

RightOrder, Inc.
404 Saratoga Avenue
Santa Clara, CA 95050
www.rightorder.com

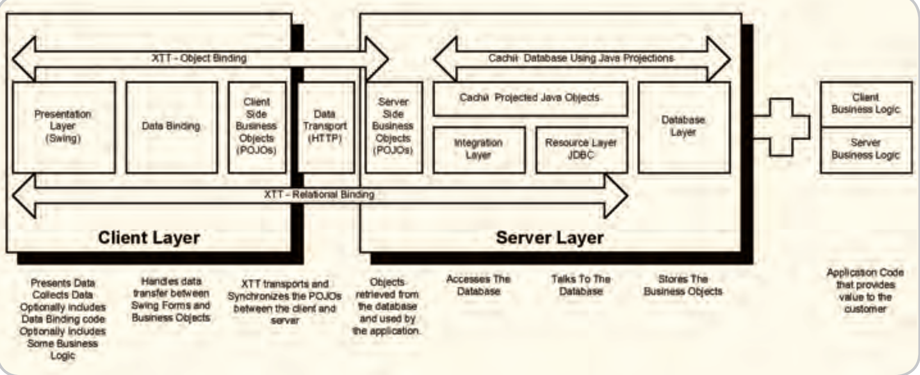

XTT - Object Binding
Cachй Database Using Java Projections
Presentation Layer (Swing)
Data Binding
Client Side Business Objects (POJOs)
Data Transport (HTTP)
Server Side Business Objects (POJOs)
Cachй Projected Java Objects
Integration Layer
Resource Layer JDBC
Database Layer
Client Business Logic
Server Business Logic
XTT - Relational Binding
Client Layer
Server Layer
Presents Data Collects Data Optionally includes Data Binding code Optionally includes Some Business Logic
Handles data transfer between Swing Forms and Business Objects
XTT transports and Synchronizes the POJOs between the client and server
Objects retrieved from the database and used by the application.
Accesses The Database
Talks To The Database
Stores The Business Objects
Application Code that provides value to the customer

Figure 7 Swing Client-Server Software Stack – Using Caché Objects and XTT

5. Happy customers => Talk to people that are using the framework to see what their experiences have been. Browse the framework forums to get a sense for the types of problems developers experience while using the framework and how they have been solved.

A good example of a plumbing framework that provides enhanced functionality while reducing the code you need to write and maintain is Struts. Frameworks are great – but you still need to decide which ones to use (if any) and when to use them. One way to evaluate frameworks (after they've met the above mentioned criteria) is to take a look at how much work they'll save you – i.e., how much of the application scope they cover (see Figure 3). As previously shown, an object database covers quite a bit of the server-side development for you automatically.

In our case, we have to identify frameworks we can use with a Swing client to provide data binding and transport functionality to our application. What are we talking about when we say "data transport" and "data binding"?

Data Transport

Data Transport is how the information is sent between the client and the server. Since Swing is written in Java, you can use just about any kind of transport technology and protocol you want to – Sockets, RMI, XML-RPC, HTTP, etc. It really comes down to which one meets your application's needs. These day's it's common to communicate over port 80 so you don't have to open other firewall ports to use your application. If you're running only in a corporate network, perhaps you'll leverage Swing's strong support for managing Sockets. Alternately, you may want to use HTTP as demonstrated in the article "Struts Meets Swing" (see references) that explains how to connect a Swing client to a Struts application.

Frameworks available that provide data transport include: Canoo's ULC, Insitech's XTT and JDNC. Canoo manages communications between "half-objects" synchronizing the objects on the client with those on the server. Insitech's XTT uses XML over HTTP to pass data and objects between the client and the server.

Data Binding

Data binding is simply a term for synchronizing the presentation layer with the model layer or to put it more simply, tying a form input field to an object property. A robust data binding solution will automatically update the presentation layer if the model layer changes and vice versa. If your presentation layer and persistence layer are on the same machine, this is very straightforward. However, if you're operating across a network, you have to deal with the data transport as well.

The number of data binding solutions has increased significantly in the last year or so – so you'll have to do your homework and review them to decide which one best meets your needs. Here's a sampling: JGoodies, Lazlo, JDNC, Canoo ULC, Insitech XTT, Oracle AD, and Spring Rich. Besides synchronizing the data between the presentation and models, some of these frameworks take care of data transport and persistence as well. By using these binding and transport frameworks, more of the plumbing of an application is taken care of automatically, further reducing the development time and the amount of code that has to be maintained. I'm now going to demonstrate how much work can be saved (and how quickly you can develop) by using some of the available frameworks.

Visualizing the Savings

Let's see how two different binding and transport frameworks assist us in reducing the amount of code we have to develop for the left side of the software application diagram (see Figures 4 and 5).

Canoo's ULC framework simplifies your life by synchronizing data on the server with the presentation layer. It provides a library of components, half-objects, that are placed on your forms and bound to their other half on the server. You don't have to write the binding code or worry about the client/server communications – it's taken care of for you.

Insitech's XTT provides binding and data synchronization between standard Swing components and relational or object data sources. As you can see from the diagram above, not only does XTT bind data with the presentation layer, it also transports it to the server and interacts directly with the persistence layer (either an object or relational database). This further reduces the amount of code you have to write and maintain. XTT ships with a collection of standard Swing components that have been extended to make them XTT-aware, but any Java component can be extended to make it usable by the framework. XTT also provides a simple mechanism for making remote method calls to server-side objects and returning serializable objects to the client.

Using either the Canoo or Insitech framework will dramatically speed up your development.

Development Simplification Rules of Thumb

- Always survey the market before starting a new project to see what new technologies and frameworks can be used to simplify your development effort.
- Use an object database to eliminate the work and overhead of mapping frameworks.
- Limit the number of layers and technologies your application uses. It will reduce complexity and simplify maintenance.
- Use frameworks based on open standards to provide the "plumbing" of your application so you can focus on the custom business logic. The frameworks should:
  - Be based on open standards => Therefore they will be supportable and extendable
  - Replace role-your-own "plumbing" => Reducing the code you write and maintain
  - Provide advanced features/functionality => Features you may not have the time or expertise to develop in-house.

Wizards

Wizards can speed up development even more by generating code for you. However, make sure the code generated is readable and maintainable. Figure 6 shows a fully functional Swing form generated with the XTT Form Wizard that shows a department and all its employees.

It took less than five minutes to generate and since it's just Java Swing, it can easily be customized and extended for use in an application. You can see a detailed demo of XTT on JavaLobby. Tools like these can really kick your development into overdrive.

More Synergies: Combining Swing Frameworks with Object Databases

By combining a Swing framework with an object database, you can cover 80%-90% of your application's plumbing. Now you can focus on developing the user interface and business logic.

Visualizing Code Coverage

Figure 7 depicts using XTT with an object database, in this case InterSystems' Caché. The specific implementation will differ somewhat for other object databases and Swing frameworks. The purpose here is to illustrate what percent of the software stack we can address by using these technologies. Here XTT fully addresses the presentation layer, data binding, and data transport layers of the application. Note that XTT can bind directly to the SQL/relational interface of the database or to POJOs that are then synchronized with objects on the server. In this configuration, developers only have work with objects on both the client and server and data persistence from the client is automatically handled by XTT.

Summary

By using object databases, Swing, and Swing binding/transport frameworks you can reduce the plumbing code you develop by up to 80%-90% for the typical enterprise application. Each database and framework provides its own unique functionality and features and should be evaluated to see which best meets your needs – but most of them can significantly reduce the amount of code you have to develop and maintain. With 80% less code to write, you're well on your way to RAD.

References

Object Databases

- Service Architecture – Object Oriented Databases & Technology: http://www.service-architecture.com/object-oriented-databases
- List of Object Database Vendors: http://www.service-architecture.com/products/object-oriented_databases.html
- InterSystems Caché Database: http://www.InterSystems.com
- db4objects Database: http://www.db4objects.com
- ObjectDB: http://www.objectdb.com/
- Progress ObjectStore Database: http://www.progress.com/realtime/products/objectstore/index.ssp
- Detailed discussion of mapping objects to relational tables: http://www.agiledata.org/essays/mappingObjects.html#ImplementingObjectRelationships

Data Binding

- Javadesktop Data Binding: https://databinding.dev.java.net/
- Getting Started With JDO: http://java.sun.com/developer/technicalArticles/J2SE/jdo/
- Jgoodies Data Binding: https://binding.dev.java.net/
- Data Binding: The Next Big Thing in Desktop Java Apps : http://www.clientjava.com/blog/2005/04/07/1112925660994.html
- Java Desktop Network Components: https://jdnc.dev.java.net/
- Spring Rich Client Project: http://www.springframework.org/spring-rcp
- Insitech XTT: http://www.insitechinc.com/
- Canoo ULC: http://www.canoo.com/ulc/index.html

Networking

- Fundamentals of Internetworking: http://www.oreillynet.com/pub/a/network/2001/02/09/net_2nd_lang.html
- Struts meets Swing: http://javaboutique.internet.com/tutorials/Swing/

Swing IDEs and Visual Editors

- Eclipse IDE: http://eclipse.org/downloads/
- NetBeans: http://www.netbeans.org/products/ide/
- Instantiations Swing Designer for Eclipse: http://www.swing-designer.com/

Swing Components

- Java Desktop Integration Components: https://jdic.dev.java.net/
- SwingX: https://swingx.dev.java.net/

Miscellaneous

- Software Tiers: http://www.opengroup.org/architecture/togaf8-doc/arch/p4/views/vus_sw.htm#Software%20Tiers
- Rich Internet Applications and AJAX - Selecting the best product: http://www.javalobby.org/articles/ajax-ria-overview/

Listing 1

```
import com.intersys.objects.*;          // Required for Caché Object Connection

Database dbconnection = null;
try{
 // Define the database connection and provide a valid username and password
 dbconnection = CachéDatabase.getDatabase(jdbc:Caché://localhost:1972/EGROK,
   dbUserName,dbPassword);

String selectedID = "1"; // Employee record ID – Test Data
 // Open the selected object by referencing the Id of the object
Id id = new Id(selectedID)
 // Open the selected object by referencing the Id of the object
 Employee employee = (Employee)Employee._open(dbconnection,id);

 // Do some stuff with the object here...
 employee.getUserName();  // Get the username – defined in the Employee Class
 employee.getLastName();// Get the LastName – defined in the base.Person Class!!
 employee.setName("Fred"); // Set a property

 employee.getDepartment().getName() // Get the department name!!!

 employee.getManager().getName() // Get the Manager's name!!!

 employee.save(); // Save the changes

 // De-reference the object
 dbconnection.closeObject(employee.getOref());
 employee=null;

 // Close the database connection
 dbconnection.close();
 dbconnection=null;

} // end try block
catch (CachéException e){
 System.out.println("Exception:" + e.toString() + ".");
}// end catch
finally{
 try{
  if(dbconnection != null)
   dbconnection.close();
 }// end try
 catch (CachéException e){
 System.out.println("Exception:" + e.toString() + ".");
 }// end catch
}// end finally
```

Joe Winchester
Desktop Java Editor

We Are Made to Persist. That's How We Find Out Who We Are

In Java's early years, the language received a lot of flak from its opponents over performance. Java turns its .class file bytecodes into machine instructions (MI) at runtime, something that costs cycles and is slower than a fully compiled language that creates the MI as part of the development stage. While to a certain extent this is true, the performance delta has all but been removed with the use of just-in-time (JIT) compilers that cache machine instructions in the VM and do other clever tricks to ensure the JVM runtime speed has very little slack. There was a time when JIT had to be switched off for debugging as it interfered with the ability to map stack and heap information back to the original source. However, even this is no longer true in the newer JVMs that can run in high-performance debug modes with no significant difference between having –Xdebug there or not.

The garbage collector is another area that the Java language has received criticism for. The original concept is that programmers don't have to worry about freeing memory, all they do is create objects at will and let the JVM determine when an object is no longer required. It is certainly a lot simpler than de-allocating heap memory manually in a C program; however, because it's based on presumptive algorithms, frequently it's unfairly blamed for JVM memory bloat or performance problems. Modern garbage collectors though are very efficient and can reclaim memory in small segments with incremental pauses to let the JVM continue uninterrupted (http://www.research.ibm.com/metronome/). They also defragment memory as they go along, and JSR 1 introduces new APIs that allow fully deterministic garbage collection and high-resolution time management, although they do introduce the problem of how to manage immortal memory.

When a Java program is launched, the *java* command contains the *–classpath* that points to a set of directories containing *.class* files and other resources required by the main class. The VM loads classes on-demand when they are first referenced by searching the directories (or .jars if they are zipped up) and loading bytecodes before compiling them. In a textbook "Hello World" program considerably more time is spent loading the JRE classes required for the user's code to run than is spent loading the main class. As the base classes are loaded, they need memory allocated for them; they need just-in-time compilation into machine instructions, and other steps such as bytecode verification and linkage all take JVM cycles to perform. Once the JVM is up and running it can rerun the scenario quickly because the bootstrap work has been done and the VM is fully warmed up. When the user exits the JVM, however, all of the work is thrown away. The next time they rerun the same Java program, all the steps required to load the base JRE classes, allocate memory for them, and so forth takes place de novo. Developers of server-side JVMs like Shiraz (http://www.haifa.il.ibm.com/projects/systems/rs/persistent.html) that run on z/OS solve this by allowing the sharing of classes between JVMs, providing good scalability. Apple's JVMs use a flavor of class-sharing known as Java Shared Archive (JSA) while Java 5 introduced formal Class Data Sharing that allows the sharing of base classes between VMs and, in the future, user-defined classes.

All of this is great news for Java: it has high-performance JITs that can work in debug conditions, garbage collectors that don't freeze the JVM each time a global sweep of the heap has to be done to determine unreferenced objects, and the sharing of data between JVMs to assist scalability. There's one area left though that I'd like to see tackled – serialization and rehydration of a JVM's state. The advantage would be that a program could be exited and re-started on subsequent re-opens as though it had just been temporarily paused. There are problems though that any such solution would encounter.

One is that the Java runtime used by the JVM when the it's re-opened might be different than the one when it was saved. In this case, the classes might have changed physical shape, with instance variables added, removed, renamed, or any number of changes that mean that the serialized instances from the first save can't be mapped to the new class shape. Binary serialization is very brittle and unforgiving when any kind of class shape change occurs. What would be nice is if there was a programmatic way to mutate instances to a new class shape. If the author of a class changes its shape, there could be a method called by the JVM that allowed them to deserialize old shaped instances and map them into new objects.

The second problem is that some objects hold handles to pointers outside the JVM that won't necessarily exist next time round. These could be references to files, sockets, GUI widgets, or anything where the object interacts with the platform in some way. This could be solved by having a clearly defined life cycle to the VM whereby objects were called back on save and load, and something like a GUI widget could keep all of its data on save except the actual window handle, and then re-create itself again from this state when the VM is reloaded. There would still be problems about what to do with something that might not be there any more, such as a file on disk that was no longer present, but if the VM had a clearly defined API cycle through which objects were saved and restored that class authors adhered to, this could be dealt with.

Java has matured in its release cycle since the early days and new versions occur in 18-month rather than weekly periods. For every Java program that runs again and again all year with the same set of unchanged classes, we need to think about how to optimize it for the common scenario: nothing in the JRE has changed, it's on the same computer that ran last time round, and the user wants it to come up as fast as any native program. I think the whole area of VM object persistence needs to be looked at again and seen whether this time around it can be made to work, benefiting programmers with more control and flexibility and users with faster and more reliable programs.

Joe Winchester is a software developer working on WebSphere development tools for IBM in Hursley, UK.

joewinchester@sys-con.com

PLATINUM COSPONSORS

IBM

ORACLE

GOLD COSPONSORS

JUSTSYSTEM

SAP

SILVER COSPONSORS

QUEST SOFTWARE

TERRACOTTA

Don't Miss the 2006 JavaOne℠ Conference

More than 300 technical sessions and Birds-of-a-Feather sessions will be offered at the 11th Annual JavaOne℠ Conference. See and hear from the Industry leaders and technology experts over four content-rich days. Included are:*

Introduction to AJAX

Ben Galbraith, *Consultant* | Dion Almaer, *Adigio, Inc*

This session provides an introduction to AJAX and an orientation to the state of the AJAXian universe. It demonstrates the basic AJAXian techniques through live coding and demonstrates and deconstructs more-advanced examples of AJAX.

Effective Java™ Reloaded

Joshua Bloch, *Google, Inc.*

It has been five years since *Effective Java*™ was released. The Java platform has evolved, and we've learned more about how to use it to best effect. Therefore, a second edition of *Effective Java*™ is being released to coincide with the 2006 JavaOne conference. This presentation covers new material that has been added to the second edition, material that should be useful to every working Java technology programmer.

Spring Framework Update

Rod Johnson, *Interface21*

Rod Johnson, the father of Spring, brings attendees up to date on some of the many significant new features in the Spring 1.3 and 1.4 releases. He discusses what's new and cool in the Spring world and examines the implications of these new features for best practice in developing applications with the Spring Framework. Johnson shows code examples throughout the presentation, leaving attendees ready to try these features out for themselves.

To see more information on the conference offerings visit java.sun.com/javaone/sf

*Content subject to change.

Register by April 14, 2006 and SAVE $100.

java.sun.com/javaone/sf

Sun microsystems

JavaOne℠ Conference | May 16-19, 2006

JavaOne℠ Pavilion: May 16–18, 2006, Moscone Center, San Francisco, CA

JavaOne

Sun's 2006 Worldwide Java Developer Conference℠

BY NOW THERE ISN'T A SOFTWARE DEVELOPER ON EARTH WHO ISN'T AWARE OF THE COLLECTION OF PROGRAMMING TECHNOLOGIES KNOWN AS AJAX!

REAL-WORLD AJAX SEMINAR

DoubleTree Hotel — **April 24, 2006** San Jose, CA

The Roosevelt Hotel — **June 5-6, 2006** New York, NY

How, in concrete terms, can you take advantage in your own projects of this newly popular way of delivering online content to users without reloading an entire page?

How soon can you be monetizing AJAX?

This "Real-World AJAX" one-day seminar aims to answer these exact questions...

Led by "The Father of AJAX" himself, the charismatic Jesse James Garrett of Adaptive Path, "Real-World AJAX" has one overriding organizing principle: its aim is to make sure that delegates fortunate enough to secure themselves a place – before all seats are sold out – will leave the seminar with a sufficient grasp of Asynchronous JavaScript and XML to enable them to build their first AJAX application on their own when they get back to their offices.

[illegible]

Jeremy Geelan
Conference Chair, Real-World AJAX
jeremy@sys-con.com

HURRY! LIMITED SEATING THIS SEMINAR WILL SELL-OUT
CALL 201-802-3022 TO REGISTER!

Register at...
www.AJAXSeminar.com

Early Bird Price (San Jose, CA) (Before March 31 / After March 31 – Price: $995)	$695
Early Bird Price (New York, NY*) (Before March 31, 2006)	$1,295
Discounted Price (New York, NY*) (After March 31, 2006)	$1,495
On-Site Price (New York, NY*) (On June 5-6, 2006)	$1,695

*Includes SOA Web Services Edge plus Enterprise Open Source Conference & Expo

LIVE SIMULCAST!
AROUND THE WORLD ON SYS-CON.TV

CORPORATE SPONSORS

Adobe telerik BACKBASE TIBCO Parlicient Microsoft

AND XML ASYNCHRONOUS JAVASCRIPT AND XML ASYNCHRONOUS JAVASCRIPT AND XML ASYNCHRONOUS JAVASCRIPT AND XML ASYNCHRONOUS JAVASCRIPT AND XML ASYNCHRONOUS JAVASCRIPT AND XML ASYNCHRONOUS JAVASCRIPT AND XML ASYNCHRONOUS JAVASCRIPT AND XML ASYNCHRONOUS JA-

What you'll walk away with...

A New understanding of AJAX and how to make it work for you, plus:

A) **Conference Proceedings Binder**
Full-conference proceedings in a self-contained commemorative binder.

B) ***Ajax in Action* Book**
A copy of the Best-Selling AJAX book by Dave Crane.

C) ***Real-World AJAX "Secrets of the Masters"* Book + DVD**
New AJAX book edited by Dion Hinchcliffe (Release Date: Summer 2006)

D) **Notebook**
Executive style keeps you looking sharp for note-taking wherever your business might take you!

E) **Computer Back-Pack**
A hip new way to conceal your laptop! The Urban Wonder Compu-Pack is an ingenious creation - a 600-denier polycanvas and nylon knapsack!

F) **T-Shirt**
100% preshrunk 6.1 oz. heavyweight cotton tagless shirt features shoulder-to-shoulder tape, and double-needle sleeves and bottom.

G) **Porcelain Mug**
Lightweight and durable, you can use it as a teacup, coffee mug or keep it as a pencil holder.

H) **Satin Silver Contemporary Pen**
Satin-silver plastic barrel with the look and feel of brass. Rubber comfort grip improves writing control and comfort.

I) **Real-World AJAX Seminar on DVD**
Watch complete seminar video and slide presentations (Release Date: Summer 2006)

FACULTY

Featuring... Real-World AJAX Rock Stars!

Jesse James Garrett
Father of "AJAX" Who Coined the Term in 2005

Jesse James Garrett is the Director of User Experience Strategy and a founding partner of Adaptive Path, the world's premier user experience consulting company. He is author of The Elements of User Experience (New Riders), and is recognized as a pioneer in the field of information architecture. Jesse's clients include AT&T, Intel, Crayola, Hewlett-Packard, Motorola, and National Public Radio. Since starting in the Internet industry in 1995, Jesse has had a hands-on role in almost every aspect of Web development, from interface design and programming to content development and high-level strategy. Today, information architects around the world depend on the tools and concepts he has developed, including the widely acclaimed "Elements of User Experience" model. He is co-founder of the Information Architecture Institute, the only professional organization dedicated to information architecture. He is also a frequent speaker and writer whose work has appeared in numerous publications, including New Architect, Digital Web, and Boxes and Arrows.

Scott Dietzen
One of the Fathers of WebLogic and J2EE
Ph.D., President and CTO, Zimbra

Scott is widely credited with helping put together the J2EE standard, launching the Web application server category, launching the Java Community Process, and driving the Web services collaboration with Microsoft and IBM. Prior to Zimbra, Scott was CTO of BEA Systems where he was the principal architect of the technology strategy for the WebLogic product family.

Bill Scott
AJAX Evangelist of Yahoo!

Bill Scott, one of the top AJAX experts in the country, is an Interaction Designer and AJAX Evangelist at Yahoo! He is part of the newly formed Design and Practices Team working with teams throughout Yahoo! to create a rich experience on the web. Before joining Yahoo! Bill founded the User Experience Team at Sabre Airline Solutions, part of Sabre Holdings. During that stint he also co-founded Rico, an open source JavaScript framework for creating AJAX & DHTML web applications. Over the past 20 years Bill has been involved in designing and creating user interfaces for video games, military war games, 3D graphics, oil and gas research, software development environments, supply chain planning, and various other scientific and business domains. He posts his musings about user experience on his blog.

David Heinemeier Hansson
Creator of Ruby on Rails

U.S.-based since November, David Heinemeier Hansson is the development lead of Rails, also known as Ruby on Rails, which the official Ruby on Rails website irreverently describes as "a full-stack, open-source web framework in Ruby for writing real-world applications with joy and less code than most frameworks spend doing XML sit-ups." He has had help from a lot of contributors. David is the creator of applications like Instiki, Basecamp, and Ta-da, and has now joined the Chicago-based team of 37signals.com, a privately held company founded in 1999 and committed to building the best web-based software products possible with the least number of features necessary. David will give his first presentation on "AJAX in Rails" at the "Real-World AJAX" one-day seminar.

Adam Bosworth
Vice President of Engineering, Google
One of the Fathers of XML & the Creator of MS Access

Adam Bosworth is Vice President of Engineering, Google. He joined Google in 2005 from BEA Systems, where he was Chief Architect & Senior VP of Advanced Development. Prior to joining BEA, Bosworth co-founded Crossgain, a software development firm acquired by BEA. Known as one of the pioneers of XML, he previously held various senior management positions at Microsoft, including General Manager of the WebData group, a team focused on defining and driving XML strategy. While at Microsoft he was also responsible for designing and delivering the Microsoft Access PC Database product and assembling and driving the team that developed the HTML engine of Internet Explorer 4.0.

Rob Gonda
Bestselling AJAX Author, CTO, iChameleon Group
Editor-in-Chief, AJAX Developer's Journal

Rob Gonda, newly appointed to the helm of SYS-CON Media's AJAX Developer's Journal, is the CTO for iChameleon Group. He is an Advanced Certified Coldfusion Developer, holds a BS in computer science and engineering and an MBA with a specialization in entrepreneurship. He recently wrote a two-part feature article on AJAX for ColdFusion Developer's Journal that in one month became the most-read article in the magazine's history.

Dion Hinchcliffe
Cofounder & CTO, Sphere of Influence Inc.
Editor-in-Chief, Web 2.0 Journal

Dion Hinchcliffe, newly appointed editor-in-chief of SYS-CON's pioneering Web 2.0 Journal, is cofounder and chief technology officer for the enterprise architecture firm Sphere of Influence Inc., in McLean, Virginia. A veteran of software development, Dion works with leading-edge technologies to accelerate project schedules and raise the bar for software quality. He is highly experienced with enterprise technologies and he designs, consults, and writes prolifically. Dion actively consults with enterprise IT clients in the federal government and Fortune 1000. He is a frequent speaker on AJAX, Web 2.0 and SOA and is currently the top-read SYS-CON.com blogger.

Paul Rademacher
Google, Creator of HousingMaps.com

Paul Rademacher is the creator of HousingMaps.com, which combined Craigslist and Google Maps for the first web mashup. Paul holds a Ph.D. in Computer Science from UNC-Chapel Hill, and worked as an R&D Engineer at Dreamworks Animation on such movies as Shrek 2 and Madagascar. Since creating HousingMaps, Paul is now at Google.

Ross Dargahi
Well-known AJAX Evangelist
Co-founder and VP of Engineering, Zimbra

Ross Dargahi, a co-founder of Zimbra, is presently the company's VP of Engineering. Prior to Zimbra, he was Director of Engineering and Director of Product Management with the Messaging Products Group at Openwave Systems - where he built and led the engineering team that designed and architected large-scale messaging subsystems and was responsible for unified messaging, multimedia messaging (MMS), and voicemail. Ross joined Openwave as part of the 1999 acquisition of Onebox where he was a founding engineer and senior architect. Prior to Onebox, Ross was at Sun Microsystems' JavaSoft division where he was a lead engineer with the Java Server Group responsible for network computer and embedded server products.

Dave Crane
Co-author of Bestselling AJAX Book 'AJAX in Action'
Dave will autograph his bestselling book for all delegates!

Dave Crane is senior developer and architect for Historic Futures Ltd., a UK firm specializing in web-based supply chain management solutions. He is also lead author of the bestselling 'AJAX in Action', and has been using the technologies that we now know as AJAX in production for several years. He has worked extensively in the IT industry over the last ten years, in areas as diverse as home automation, finance and national policy planning, for companies ranging from start-ups to FTSE 100 market leaders, using technologies as diverse as J2EE, Python, Ruby, PHP, Linux and .NET. He holds degrees in environmental science, parallel computing, and a Ph.D. in simulation modelling and philosophy. He is active as a writer, trainer and mentor in the AJAX commun

Christian Cantrell
Coauthor of JavaScript/Flash Integration Kit
AJAX/Flash Integration Guru

Christian Cantrell is a Product Manager for Developer Relations with Adobe and a big fan of AJAX. He has been developing large-scale, web-based applications in ColdFusion, Java, JSP, and Flash for more than six years. He is the author of numerous tutorials and white papers, and is coauthor of Flash Enabled: Flash Design & Development for Devices as well as the JavaScript/Flash Integration Kit. Christian is a leading expert on AJAX to Flash integration.

Sahil Malik
telerik Tech Evangelist, Microsoft MVP
Author of Bestselling ADO.NET 2.0 book

Sahil Malik is an independent consultant, trainer and mentor in various Microsoft Technologies and has been closely involved with telerik as a technology strategist. He has worked for many large clients across the globe including a good deal of Fortune 100 companies and US government organizations. He is currently leading the office of Emerging Technologies at a prominent government office where he is in charge of reviewing, assessing and recommending various technologies to support the organization. Malik frequently speaks on a variety of .NET related topics at local user group meetings and industry events. For his community involvement and contribution, he has been awarded the Microsoft MVP award. He can be reached at www.winsmarts.com.

Jouk Pleiter
Co-Founder & CEO of Backbase

Jouk Pleiter is the CEO of Backbase, a leader in the field of Rich Internet Applications and AJAX development software. Backbase's clients include ING, ABN AMRO, TNT, KPN, Comsys and Heineken. Backbase operates globally with offices in San Mateo (North America) and Amsterdam (Europe). Since 1995, Jouk has been an entrepreneur: he founded three successful software companies. Prior to Backbase, Jouk was part of the founding team at the web content management company Tridion, where he led the product management operations, and was driving the company's efforts to become a leader in the European WCM software market. Jouk previously was part of the founding team at the Interactive Agency Twinspark where he grew the company to a leading market position in Europe and was instrumental in the sale of Twinspark to Agency.com. He has an MBA from the University of Groningen.

Kevin Hakman
Director of Product Marketing for TIBCO
General Interface TIBCO Software

Kevin Hakman is the director of product marketing for TIBCO General Interface, the award winning AJAX and Rich Internet Application framework and toolkit. Kevin Hakman pioneered AJAX in the enterprise co-founding General Interface in 2001. Since that time General Interface (aka 'GI') has been powering Web applications that look, feel and perform like desktop applications, but run in the browser at Fortune 500 and U.S. Government organizations. General Interface was also the first to use its own toolkit to provide full visual tooling for AJAX when it released it's 2.0 Version in 2003. TIBCO acquired General Interface in 2004 to extend its vision for service oriented applications to the end user. Kevin is a contributor to the SOA Web Services Journal and the AJAX Developer's Journal.

Shanku Niyogi
Product Unit Manager of the UI Framework
and Services Team Microsoft Corporation

Shanku is Product Unit Manager of the UI Framework and Services (UiFX) team, which is responsible for delivering high-productivity UI framework technologies for the .NET platform, including ASP.NET, Atlas, Windows Forms, and frameworks for smart clients. Prior to his current role, Shanku was Group Program Manager of the Web Platform and Tools team on the Whidbey release of ASP.NET and Visual Web Developer. Shanku joined Microsoft in 1998 as a developer, having spent several years shipping products in the Windows ISV industry. Shanku holds a Bachelor of Mathematics degree in Computer Science from the University of Waterloo.

For more information...

Call 201-802-3022 or email events@sys-con.com

SYS-CON EVENTS **For more great events visit www.EVENTS.SYS-CON.com**

NOTE: SPEAKER LINE-UP SUBJECT TO CHANGE WITHOUT NOTICE
VISIT WWW.AJAXSEMINAR.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

Struts and JavaServer Faces

Design patterns for list selection

by Heman Robinson

Jakarta Struts[1] provides a standard framework for Web applications, and JavaServer Faces[2] offers a component-based framework for user interfaces.

At the user interface, a common task in both frameworks is selecting items from lists. Over the years, standard design patterns have been developed for selection lists. For developers, the advantage of implementing these standard patterns is that they are readily available and offer a familiar user experience.

This article describes the following patterns:

- One selection from few
- One selection from many
- Multiple selections from few
- Extended selections from many
- Multiple selections from many

These are small idioms for simple UI controls, but it's not always obvious how to implement them to maximize usability. This article describes best practices recommended by usability experts for these patterns and compares their implementations in Struts and JavaServer Faces.

Background

Both Struts and JSF architectures are based on the Model-View-Controller design pattern. This article focuses on the View, implemented using JavaServer Pages. JSPs are typical in Struts architectures[3]. They're not required by JavaServer Faces, but are used here for ease of comparison.

In both frameworks, JSPs are backed by JavaBeans, for which the interfaces are nearly identical in Struts and JSF. In the Struts bean, List properties are stored as LabelValueBeans; in the JSF bean they are SelectItems. Both JavaBeans are initialized in their constructors, again for ease of comparison.

JavaBean interfaces and JSP comparisons are shown in this article. JavaBean variables and constructors are shown in Listings 1 and 2. Complete JSPs are in Listings 3 and 4. The patterns that follow show controls that might be used in a Web search application. (Listings 1, 2, and additional source code can be downloaded from the online version of this article at http://jdj.sys-con.com.)

Heman Robinson is a senior developer with SAS Institute in Cary, N.C. He holds a BS in mathematics from the University of North Carolina and an MS in computer science from the University of Southern California. He has specialized in GUI design and development for 15 years and has been a Java developer since 1996.

hemanrobinson@yahoo.com

Pattern 1: One Selection from Few

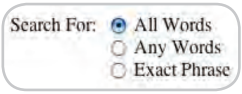
Use when:

- One selection is needed.
- There is room to display all available items.

The best design pattern for one selection from few items is a list of radio buttons, also known as option buttons. This pattern shows all available items and the selection requires only one mouse-click.

As pointed out in *GUI Design Essentials*[4], vertical alignment makes the text easier to scan. This also makes the radio buttons easier to click, as it places them all in the same area. It's not always possible to align buttons vertically, but if you can, it's a service to your users.

Radio buttons are mutually exclusive, so one button should be selected by default[5, 6]. This costs your users nothing and may save them a step.



Search For: [✓ All Words] Any Words Exact Phrase

Figure 1 One selection from few

The Windows Interface Guidelines for Software Design recommends this pattern for lists with seven items or fewer[7]. If you have more, or screen real estate is limited, consider Pattern 2.

In the JavaBean, the "forList" property stores the list of available values. The "forValue" property stores the selected value as a string.

```
public List getForList() ...
public void setForList( List list ) ...
public String getForValue() ...
public void setForValue( String id ) ...
```

In the Struts JSP, the <logic:iterate> tag loops over the buttons specified by <html:radio> and <bean:write> tags. The "Search For:" label is provided by the <fmt:message> tag from the JSP Standard Tag Library[8]. Struts tags and JSTL tags are often used in combination.

```
<tr valign="top">
  <td align="right">
    <fmt:message key="searchFor"/>
  </td>
  <td>
    <logic:iterate name="exampleForm" property="forList"
      id="item">
      <html:radio property="forValue" idName="item"
        value="value">
    &nbsp;<bean:write name="item" property="label"/><br />
      </html:radio>
      </logic:iterate>
    </td>
</tr>
```

In the JavaServer Faces JSP, the <h:selectOneRadio> tag loops over the list specified by the <f:selectItems> tag. The “Search For:” label is provided by the <h:outputText> tag.

```
<h:panelGrid columns="2">
    <h:outputText value="#{bundle.searchFor}" />
    <h:selectOneRadio value="#{example.forValue}"
        layout="pageDirection" id="for">
        <f:selectItems value="#{example.forList}"/>
    </h:selectOneRadio>
</h:panelGrid>
```

In the JavaServer Faces JSP, code size is reduced by using more compact tags for both controls and layout. The layout tags <tr> and <td> are replaced by <h:panelGrid>. For brevity, layout and label tags are omitted in succeeding patterns.

Pattern 2: One Selection from Many

Use when:

- One selection is needed.
- There is not enough room to display all available items.

The standard pattern for one selection from many items is the listbox. A disadvantage of this pattern is that not all available items are visible. An advantage is that it requires only a small amount of space[9].

This control can display several items or one. When several items are displayed, *GUI Design Essentials* recommends a minimum of three items and a maximum of eight.

When only one item is displayed, the control appears as a dropdown menu. The drop-down variant conserves more space but requires more user interaction[7].

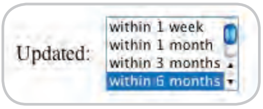


Figure 2 One selection from many (Listbox variant)

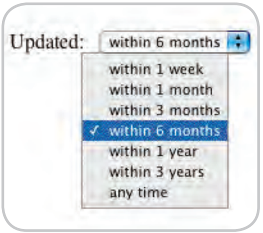


Figure 3 One selection from many (Dropdown variant)

In the JavaBean, the “updateList” property stores the list of available values. The “updateValue” property stores the selected value as a string.

```
public List getUpdateList() ...
public void setUpdateList( List list ) ...
public String getUpdateValue() ...
public void setUpdateValue( String id ) ...
```

In the Struts JSP, the <html:select> tag creates this control. The <html:optionsCollection> specifies the list. The “size” attribute of the <html:select> tag controls the number of items displayed. If “size” is omitted, it defaults to 1, which makes this control a dropdown menu.

```
<html:select property="updateValue" size="4">
    <html:optionsCollection property="updateList"/>
</html:select>
```

In JavaServer Faces, the JSP is similar to that of Pattern 1. The <h:selectOneListbox> tag loops over the list specified by the <f:selectItems> tag. The “size” attribute of the <h:selectOneListbox> tag works similarly to that of the Struts tag. If size=1, the <h:selectOneListbox> tag renders the same dropdown menu as the <h:selectOneMenu> tag, which is clearer.

```
<h:selectOneListbox value="#{example.updateValue}" size="4"
    id="update">
    <f:selectItems value="#{example.updateList}"/>
</h:selectOneListbox>
```

Pattern 3: Multiple Selections from Few

Use when:

- Multiple selections are needed.
- There is room to display all available items.

The standard pattern for multiple selections from few items is a list of check boxes. This pattern shows all the items and requires only one mouse-click per selection.

As in Pattern 1, vertical alignment makes the checkboxes easier to click and the text easier to scan[4].

The Windows Interface Guidelines for Software Design recommends this pattern for lists with 10 items or fewer. If you have more, or screen real estate is limited, consider Patterns 4 and 5.

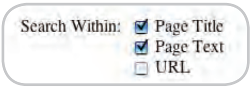


Figure 4 Multiple selections from few

In the JavaBean, the “withinList” property stores the list of available values. The “withinValues” property stores the selected value as an array of strings.

```
public List getWithinList() ...
public void setWithinList( List list ) ...
public String[] getWithinValues() ...
public void setWithinValues( String[] list ) ...
```

In the Struts JSP, the <logic:iterate> tag loops over the checkboxes specified by <html:multibox> and <bean:write> tags.

```
<logic:iterate name="exampleForm" property="withinList"
    id="item">
    <html:multibox property="withinValues">
        <bean:write name="item" property="value"/>
    </html:multibox>
    <bean:write name="item" property="label"/><br />
</logic:iterate>
```

In JavaServer Faces, the JSP is consistent with that of Pattern 1. The <h:selectManyCheckbox> tag loops over the list specified by the <f:selectItems> tag.

```
<h:selectManyCheckbox value="#{example.withinValues}"
    layout="pageDirection" id="within">
    <f:selectItems value="#{example.withinList}"/>
</h:selectManyCheckbox>
```

In Struts, unchecked checkboxes present a problem. Ted Husted provides a good explanation in his “Struts Tips”[10] (http://husted.com/struts/tips/007.html):

The multibox leverages the way HTML handles checkboxes. If the box is not checked, the browser does not submit a value for the control... This behavior is the reason there is a reset() method…

The usual solution is to turn all checkboxes off in the reset() method of your JavaBean:

```
public void reset( ActionMapping mapping,
    HttpServletRequest request )
{     ...
```

```
    setWithinValues( new String[ 0 ]);
        ...
}
```

Bill Siggelkow in the *Jakarta Struts Cookbook*[11] notes this does not handle all cases, and suggests other solutions. In JavaServer Faces, this problem does not occur.

Pattern 4: Extended Selections from Many

Use when:

- Multiple selections are needed.
- There is not enough room to display all available items.
- Users are experienced.

For multiple selections from many items, there are two traditional design patterns. In Pattern 4, items are displayed in a listbox. Key combinations such as shift-click and ctrl-click are used to select multiple items.

The Windows Interface Guidelines for Software Design calls this an "extended selection" listbox to emphasize its efficiency for contiguous selections. For example, all of the list items can be selected with only three mouse actions: click, scroll, shift-click.

One disadvantage of this pattern is that not all users know these multiple-selection techniques. This design pattern is suitable for experienced users.

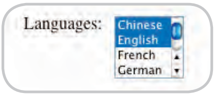


Figure 5 Multiple selections from many

Another disadvantage is that listboxes can be configured for either single or multiple selection[9]. So even for experienced users, it may not be clear when multiple selection is possible.

In the JavaBean, the "languageList" property stores the list of available values. The "languageValues" property stores the selected value as an array of strings.

```
public List getLanguageList() ...
public void setLanguageList( List list ) ...
public String[] getLanguageValues() ...
public void setLanguageValues( String[] list ) ...
```

In Struts, the JSP is the same as that of Pattern 2, except for the addition of the "multiple" and "size" attributes. These specify multiple selection and the number of items visible.

```
<html:select property="languageValues"
    size="4" multiple="true">
    <html:optionsCollection property="languageList"/>
</html:select>
```

In JavaServer Faces, the JSP is similar to that of Patterns 1, 2, and 3. The <h:selectManyListbox> tag loops over the list specified by the <f:selectItems> tag.

```
<h:selectManyListbox value="#{example.languageValues}"
    size="4" id="lang4">
    <f:selectItems value="#{example.languageList}"/>
</h:selectManyListbox>
```

Pattern 5: Multiple Selections from Many

Use when:

- Multiple selections are needed.
- There is not enough room to display all available items.
- Users are not experienced.

The alternative to Pattern 4 is to place the checkboxes of Pattern 3 within a scrollable area. The visually obvious checkboxes provide an easy learning curve for inexperienced users[6].

However, this pattern is not optimized for contiguous selection. Long sequences must be selected by clicking on each checkbox individually. Also, the checkboxes present smaller targets. In Pattern 4, the user can click anywhere on the item[7].

Neither of these design patterns is perfect, and in our applications we must decide between them. If you have experienced users who know the shift-click and ctrl-click techniques, they will appreciate the power of Pattern 4. If your users are not experienced, Pattern 5 is easier to learn.

The implementation of Pattern 5 is similar to Pattern 3. In the JavaBean, the "languageList" property stores the list of available values. The "languageValues" property stores the selected value as an array of strings.

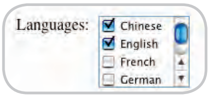


Figure 6 Multiple selections from many

```
public List getLanguageList() ...
public void setLanguageList( List list ) ...
public String[] getLanguageValues() ...
public void setLanguageValues( String[] list ) ...
```

In the Struts JSP, the addition of the <div> tag provides scrolling. The <logic:iterate> tag loops over the checkboxes specified by <html:multibox> and <bean:write> tags.

```
<div style="border:1px solid #7f9db9;
    overflow:auto; height:70px; width:90px;
    font-family:Lucida Grande; font-size:8pt;">
    <logic:iterate name="exampleForm" property="languageList"
        id="item">
        <html:multibox property="languageValues">
            <bean:write name="item" property="value"/>
        </html:multibox>
        <bean:write name="item" property="label"/><br />
    </logic:iterate>
</div>
```

In the JavaServer Faces JSP, the <h:selectManyCheckbox> tag loops over the list specified by the <f:selectItems> tag. Because they're not JSF tags, the <div> tags must be wrapped in <f:verbatim> tags.

```
<f:verbatim>
    <div style="border:1px solid #7f9db9;
        overflow:auto; height:70px; width:110px;
        font-family:Lucida Grande; font-size:8pt;">
</f:verbatim>
<h:selectManyCheckbox value="#{example.languageValues}"
    layout="pageDirection" id="lang5">
    <f:selectItems value="#{example.languageList}"/>
</h:selectManyCheckbox>
<f:verbatim>
    </div>
</f:verbatim>
```

JAVA DEVELOPER'S JOURNAL Advertiser Index

Advertiser	URL	Phone	Page
AJAX Seminar	www.ajaxseminar.com	201-802-3022	48-49
Altova	www.altova.com	978-816-1600	4
ArcMind, Inc.	www.arc-mind.com	520-290-6588	33
Elastic Path	www.elasticpath.com/download	800-942-5282	25
Extentech	www.extentech.com/jdj	415-759-5292	39
Fiorano	www.fiorano.com/downloads	800-663-3621	19
IBM	www.ibm.com/middleware/tools	617-621-0600	13
InterSystems	www.intersystems.com/ensemble12p	617-621-0600	7
IT Solutions Guide	www.itsolutions.sys-con.com	888-303-5282	53
JadeLiquid Software	www.webrenderer.com	+61-3-6226-6274	11
JavaOne Conference 2006	java.sun.com/javaone/sf		47
Microsoft Corp.	www.learn2asp.net/webcasts		27
Northwoods Software Corp.	www.nwoods.com	800-434-9820	41
OpenAccess Software, Inc.	www.openaccesssoftware.com	888-805-ODBC	37
OPNET Technologies, Inc.	www.opnet.com/pinpoint	240-497-3000	15
Parasoft Corporation	www.parasoft.com/jdjmagazine	888-305-0041	Cover IV
Quest Software	www.quest.com/911	949-754-8000	Cover II
RightOrder, Inc.	www.rightorder.com	408-983-4000	43
SOA/EOS Conference	www.soaeosconference.sys-con.com	201-802-3022	60-61
Software FX	www.softwarefx.com	800-392-4278	Cover III
Synaptris	www.intelliview.com/jdj	1-866-99IVIEW	21
TIBCO Software Inc.	www.tibco.com/mk/gi	800-420-8450	17
WebAppCabaret	www.webappcabaret.com/jdj.jsp	866-256-7973	23

Conclusion

This article described standard design patterns for selection lists and compared their implementations in Struts and JavaServer Faces.

These simple patterns provide only a taste of these frameworks. Still, some facts are clear. As shown in the listings that follow, in JavaServer Faces, JSP tags are more consistent and code size is significantly reduced. The backing JavaBean is similar in Struts and JSF, which allows developers an easy migration path.

No matter which technology you use, the recommended best practices for these design patterns will continue to be valuable. They provide familiar, usable interfaces for selecting items from lists.

Resources

1. Apache Software Foundation (2006): http://struts.apache.org/
2. Bergsten, H. (2004). *JavaServer Faces*. O'Reilly.
3. Cavaness, C. (2004). *Programming Jakarta Struts, Second Edition*. O'Reilly.
4. Weinschenk, S.; Jamar, P.; and Yeo, S. (1997). *GUI Design Essentials*. Wiley & Sons. p. 192–194, 197, 206–207.
5. Johnson, J. (2000). *GUI Bloopers*. Morgan-Kaufman. p. 94–95.
6. Cooper, A., and Reimann, R. (2003). *About Face 2.0: The Essentials of Interaction Design*. Wiley. p. 346–347.
7. Microsoft Corp. (1995). *The Windows Interface Guidelines for Software Design*. Microsoft. p. 145-54.
8. Sun Microsystems Inc. JavaServer Pages Standard Tag Library: http://java.sun.com/products/jsp/jstl/
9. Tidwell, J. (2005). *Designing Interfaces*. O'Reilly. p. 212, 235-236.
10. Husted, T. (2006). Struts Tips: http://husted.com/struts/tips/
11. Siggelkow, B. (2005). *Jakarta Struts Cookbook*. O'Reilly. p. 88–90, 147.

Listing 3: Struts JSP

```
 <%@ taglib uri="/WEB-INF/tlds/struts-bean.tld"
   prefix="bean" %>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld"
   prefix="html" %>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld"
   prefix="logic" %>
<%@ taglib uri="/WEB-INF/tlds/c.tld" prefix="c" %>
<%@ taglib uri="/WEB-INF/tlds/fmt.tld" prefix="fmt" %>

<HTML>
<HEAD>
   <TITLE>Selection List Examples</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<fmt:setBundle basename="com.kowaldesign.example.example"/>

<html:form action="/exampleWrite.do">
<table border="0" cellpadding="0" cellspacing="10">
   <tr valign="top">

      <%-- Pattern 1:  One Selection from Few --%>
      <td align="right">
         <fmt:message key="searchFor"/>
      </td>
      <td>
         <logic:iterate name="exampleForm"
            property="forList" id="item">
            <html:radio property="forValue" idName="item"
               value="value">
   &nbsp;<bean:write name="item" property="label"/><br />
            </html:radio>
         </logic:iterate>
      </td>

      <%-- Pattern 3:  Multiple Selections from Few --%>
      <td align="right">
         <fmt:message key="searchWithin"/>
      </td>
      <td>
         <logic:iterate name="exampleForm"
            property="withinList" id="item">
            <html:multibox property="withinValues">
               <bean:write name="item" property="value"/>
            </html:multibox>
            <bean:write name="item" property="label"/><br/>
         </logic:iterate>
      </td>
   </tr>

   <%-- Pattern 2:  One Selection from Many --%>
   <tr>
      <td />
      <td align="right">
         <fmt:message key="updated"/>
      </td>
      <td>
         <html:select property="updateValue" size="4">
            <html:optionsCollection property="updateList"/>
         </html:select>
      </td>
      <td />
   </tr>
   <tr valign="top">

      <%-- Pattern 4:  Multiple Selections from Many --%>
      <td align="right">
         <fmt:message key="languages"/>
      </td>
      <td>
         <html:select property="languageValues"
            size="4" multiple="true">
            <html:optionsCollection
               property="languageList"/>
         </html:select>
      </td>

      <%-- Pattern 5:  Multiple Selections from Many --%>
      <td align="right">
         <fmt:message key="languages"/>
      </td>
      <td>
         <div style="border:1px solid #7f9db9;
            overflow:auto; height:70px; width:90px;
            font-family:Lucida Grande; font-size:8pt;">
            <logic:iterate name="exampleForm"
               property="languageList" id="item">
               <html:multibox property="languageValues">
                  <bean:write name="item" property="value"/>
               </html:multibox>
               <bean:write name="item"
                  property="label"/><br />
            </logic:iterate>
         </div>
      </td>
   </tr>

   <tr align="middle" valign="top">
      <td colspan="4">
         <input type="button"
            value="<fmt:message key='submit'/>" />
      </td>
   </tr>
</table>

</html:form>
</BODY>
</HTML>
```

Listing 4: JavaServer Faces JSP

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<HTML>
<HEAD>
   <TITLE>Selection List Examples</TITLE>
   <link rel="stylesheet" type="text/css" href='<%=
      request.getContextPath() + "/stylesheet.css" %>'>
</HEAD>
<BODY BGCOLOR="white">
<f:loadBundle
   basename="com.kowaldesign.example.example" var="bundle"/>

<f:view>
   <h:form id="form">
      <h:panelGrid columns="1" rowClasses="center">
      <h:panelGrid columns="5">

         <%-- Pattern 1:  One Selection from Few --%>
         <h:outputText value="#{bundle.searchFor}" />
         <h:selectOneRadio value="#{example.forValue}"
            layout="pageDirection" id="for">
            <f:selectItems value="#{example.forList}"/>
         </h:selectOneRadio>
         <h:outputText value="" />

         <%-- Pattern 3:  Multiple Selections from Few --%>
         <h:outputText value="#{bundle.searchWithin}" />
         <h:selectManyCheckbox
            value="#{example.withinValues}"
            layout="pageDirection" id="within">
            <f:selectItems value="#{example.withinList}"/>
         </h:selectManyCheckbox>
      </h:panelGrid>

      <%-- Pattern 2:  One Selection from Many --%>
      <h:panelGrid columns="2">
         <h:outputText value="#{bundle.updated}" />
         <h:selectOneListbox value="#{example.updateValue}"
            size="4" id="update">
            <f:selectItems value="#{example.updateList}"/>
         </h:selectOneListbox>
      </h:panelGrid>
      <h:panelGrid columns="5">

         <%-- Pattern 4:  Multiple Selections from Many --%>
         <h:outputText value="#{bundle.languages}" />
         <h:selectManyListbox size="4" id="lang4"
            value="#{example.languageValues}">
            <f:selectItems value="#{example.languageList}"/>
         </h:selectManyListbox>
         <h:outputText value="" />

         <%-- Pattern 5:  Multiple Selections from Many --%>
         <h:outputText value="#{bundle.languages}" />
         <h:panelGroup>
            <f:verbatim>
               <div style="border:1px solid #7f9db9;
               overflow:auto; height:70px; width:110px;
               font-family:Lucida Grande; font-size:8pt;">
            </f:verbatim>
            <h:selectManyCheckbox
               value="#{example.languageValues}"
               layout="pageDirection" id="lang5">
               <f:selectItems
                  value="#{example.languageList}"/>
            </h:selectManyCheckbox>
            <f:verbatim>
               </div>
            </f:verbatim>
         </h:panelGroup>
      </h:panelGrid>

      <h:panelGrid columns="1">
         <h:commandButton action="#{example.submit}"
            value="#{bundle.submit}" />
      </h:panelGrid>
      </h:panelGrid>
   </h:form>
</f:view>
</BODY>
</HTML>
```

Don't Tell Me **Cause It Hurts**

–continued from page 6

Moving Out?

Modern IT is a moving target and, if you can't keep up with it, you should find something more suitable for yourself. If you are considering opening a business (I hope you're not planning to get into real estate), be prepared to live without any income for at least a year. If you have saved some cash, this might be a good time to try to capitalize on that business idea you've dreamed about for years. If you're not a businessman, stay in IT. Ask yourself this question: "How can I earn $60K a year in a non-IT world?" The most probable answer is to start by investing $50K+ into a business hoping to break even in a year and start earning some money. The other choice is to change your profession. Go back to college and in about two years you'll become a junior engineer or biochemist with a $40K salary and a similar amount of student loans.

Staying In!

Here's another plan:

1. Don't even start polluting the job market with your current résumé.
2. For the next six months sleep six hours a day tops. Spend no more than three hours for food intake, wife/girlfriend, and kids. Spend the remaining 15 hours with books and your PC.
3. Find a couple of reputable courses that teach the programming tools of your choice. These weekly courses are expensive, but they give you a chance to listen and communicate with experts in this field. To make these courses more effective, take them after spending a substantial amount of time studying on your own. For example, if you are planning to learn about WebLogic application server, don't sign up for a class until you tried to learn it on your own. Install the software, read the books, and study the source code of sample applications. This way you'll be better able to absorb the course info in the classroom and will ask the right questions. After taking the first course, continue studying at home and take another (more advanced) course in a month or two. Take any available free online courses on the subject of your choice. Attend professional seminars, user groups meetings, and sign up for each free technical Webinar. Check out the schools of continuing education at your local college. They may be offering evening/weekend classes.
4. Join an open source project (see sourceforge.net).
5. After all of the above is done, include your new skills in your résumé and hit the job market.

Sorry for the cold-blooded coverage of this unpleasant topic, but victims of layoffs might already be sick of hearing that "It's going to get better any moment." Just don't stop fighting – looking for work is a full-time job and has to be done the right way. Keep pushing and they won't have any other choice but to hire you! Come back – you can do it!

SMBs Should Take Note of Sun's Java Studio Creator 2

Reviewed by
Jason Halla

Sun Java Studio Creator 2 – Early Access

Sun's Java Studio Creator 2 is the company's upcoming second release of its somewhat lauded, somewhat maligned visual application development environment. In this review I give Java Studio Creator 2 (JSC2) Early Access edition a fresh look and discuss its merits and limitations.

For the uninitiated, JSC2 is Sun's effort to bring the same visual approach to Java/J2EE Website development that Microsoft Visual Studio or Macromedia Dreamweaver users have enjoyed for years. Sun works to make the visual design approach in JSC2 more comprehensive from a J2EE perspective than what you would expect from Dreamweaver by allowing JSC2 users to tap directly into business logic and data represented by EJBs, Web services, or databases using a drag-and-drop paradigm.

Sun's target audience for this IDE are somewhat-technical employees working at small-to-medium businesses (SMBs). I say "somewhat technical" and not "developer" since by taking a primarily visual approach to application development, JSC2 opens the role of the front-end developer to graphic artists, business and/or system analysts, and other users not traditionally associated with actual J2EE development.

Getting Started

For this review, I loaded JSC2 on an HP laptop running Windows XP Pro, with an AMD Athlon 64 3200+ processor, 1.25GHz RAM and 40GB hard drive. Sun recommends a slightly less beefy machine, running at least Win 2K or XP, with an Intel P4 1GHz, 512MB of RAM, and 325MB of hard disk space. JSC2 is available for Solaris, Windows, Linux (Java Desktop or RedHat), and OSX – that's right, "write once, run anywhere" is alive and well, thank you very much.

Installation did not go smoothly the first time round; I ran into an error with the bundled PointBase database engine and had to reinstall with a later build. I write this instance off as being an "early access" bug; the second installation attempt went off without a hitch.

JSC2 is built atop of NetBeans 4.1 – Sun's open source, all purpose Java IDE – and is able to leverage a few advantages due to its heritage. First, JSC2 gives the developer access to roughly the same set of tools that NetBeans provides – editors for all of the common languages associated with Java/J2EE development (Java, JSP, XML, WSDL, and the like), robust refactoring support, project building, HTTP monitoring, and more. JSC2 currently allows development in JDK 1.3.x and 1.4.x; 1.5 is not yet supported.

Second, NetBeans is a well-liked IDE with a robust community of developers backing it; if you run into a question or have an issue with JSC2, chances are good you'll be able to find assistance from within the NetBeans community online. While Sun doesn't recommend that JSC2 act as your primary IDE (more on this later), should a Java developer need to dig into the code, the tool certainly can support this.

Java Studio Creator 2 is available free for members of Sun's Developer Network; non-members pay a yearly subscription fee of $99. The pricing is very competitive with Microsoft's .NET Visual Studio Standard edition, which retails for $299, and with Macromedia's Dreamweaver 8 at $399.

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
Web: http://developers.sun.com
Phone: 800 555-9SUN

First Impressions

When first starting JSC2, the user is presented with the Welcome screen (see Figure 1), which offers options for opening an existing project, starting a new one, or exploring tutorials on how to best use the IDE.

Setting up your own project is similar to the process in NetBeans. The breadth

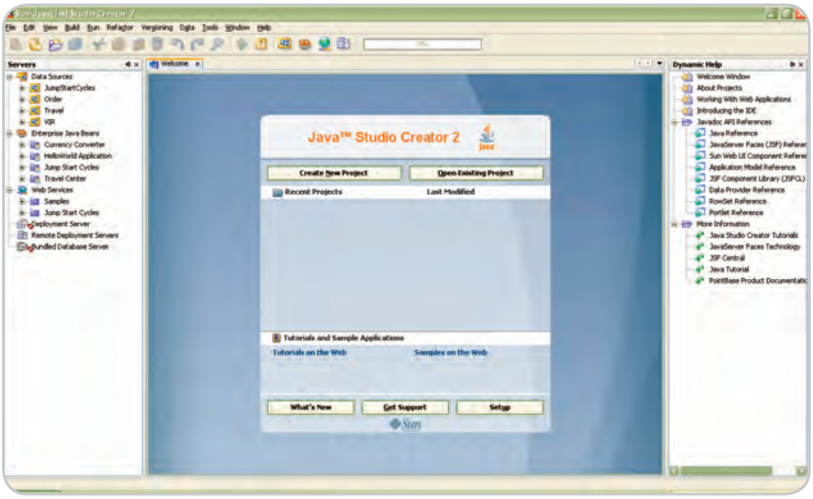


Figure 1 Welcome screen

Jason Halla is an enterprise application architect with a Fortune 500 company in Indianapolis, specializing in J2EE, SOA and LAMP architectures, as well as the moderator of Devshed.com's popular Java, PHP, XML, and software design forums.

drgroove@myway.com

of project types, of course, isn't as comprehensive as NetBeans, Eclipse, or other Java IDEs such as IntelliJ or JDeveloper, but reasonable considering JSC2's focus on J2EE development.

Slow Going

After selecting a demo project to launch, JSC2 hung for quite a long time – several minutes, at least – before coming back to life. I have a reasonably well-endowed machine, and expected JSC2 to be a tad more responsive. Fortunately, subsequent launches of the IDE into an open project were faster affairs. Unfortunately, this wouldn't be the last time I would encounter slow performance from JSC2 during the course of my review; the IDE would sometimes take a few minutes to redraw the interface after having closed the lid on my laptop (see Figure 2). Quickly comparing JSC2 to Eclipse, which has always been rather speedy, the first major difference to come to mind is Eclipse's pioneering use of SWT instead of Swing, which is what NetBeans and therefore JSC2 use. However, without spending time tracing down the exact cause of the sporadic slowness in JSC2, it is difficult to know for certain.

Another minor niggle: running the bundled PointBase database in the background opens a shell window, which is left open as long as PointBase is running. To a battle-hardened J2EE developer, this is immaterial; however, seeing as JSC2 is meant to open the world of Website development to a larger, non-developer audience, it seems a little ghetto, especially in consideration of how polished the rest of JSC2 is. Hopefully, subsequent editions of JSC omit the phantom shell.

Visual Approach

I appreciated the fact that Sun took the visual approach to its logical conclusion in Java Studio Creator, working this concept into each of the main IDE features. The list of features includes (obviously) a visual JSP editor, visual database Query editor, visual CSS stylesheet editor, and visual page navigation editor.

Design, JSP, and Java Modes

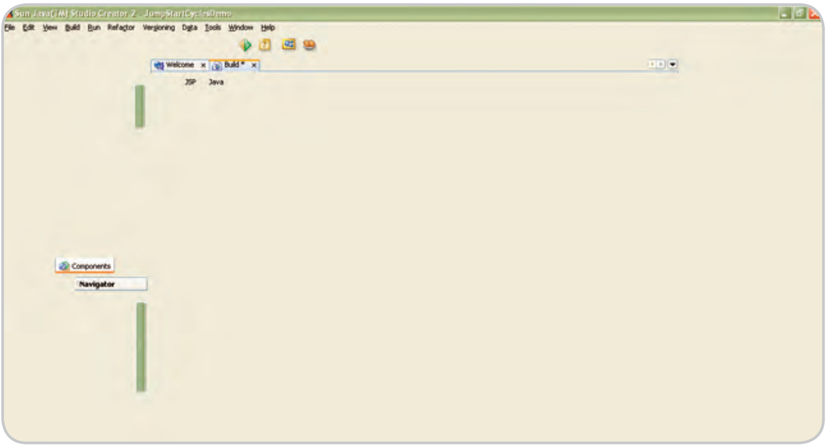
Web pages from a project opened in JSC2 start the user off in Design mode. The user can drag-and-drop user interface (UI) components onto the rendered JSP page, and manipulate and customize from there. Each Web page has an associated Design, JSP, and Java view. The JSP mode allows you to edit the JSF tags underlying a page; the Java mode opens up the code for the bean that drives the page's behavior and content (called a PageBean in JSC parlance). While the JSP and Java editors in JSC2 are top-notch, Design mode is king for this IDE, and it's where you're likely to spend most of your time working (see Figure 3).

Drag-and-Drop Power

JSC2 offers up a comprehensive selection of user interface components, built on top of the JavaServer Faces (JSF) spec. All common form elements are represented (radio buttons, password fields, text areas, drop-menus, etc.). Tables come with the ability to sort each column, calendars, messages (for providing alerts or status updates), and a JavaScript-driven tab set. Also included are so-called composite components, such as a breadcrumb trail or an add/remove list, as well as data validators and converters, and code clips (snippets of code to help power your app). Sun has provided a strong stable of UI components likely to be used in your app and then some. UI components are accessed from the palette pane; these are grouped logically by type (Basic, Layout, Composite, etc.); you can create new categories based on your own organization if desired. JSC2 generates clean, well-formed JSF tags within the JSP for each UI component dragged onto a page in Design mode, as well as Java code for that page's associated PageBean.

The Fear of Generated Code

That brings us to the scary world of generated code – the fear of which is best known by its clinical name "Codiusgeneriphobia" – a black concept in the mind of most developers. It's one thing to have your IDE generate rote code, such as getters/setters, or onerous code like EJB remote/home interfaces; it's quite


Sun Java(TM) Studio Creator 2 - JumpStartCyclesDemo
File Edit View Build Run Refactor Versioning Data Tools Window Help
Welcome
Build
JSP
Java
Components
Navigator
Figure 2 Odd behavior after PC logoff

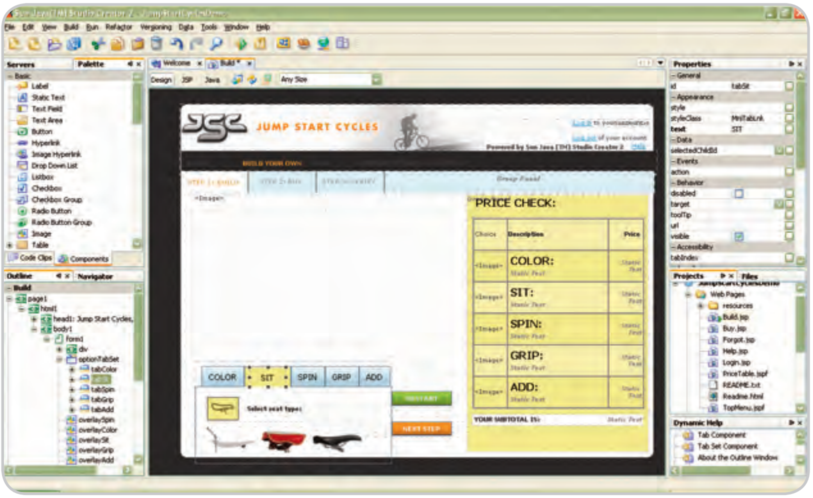


Figure 3 Design view

another for an IDE to generate code that might actually drive some of your apps behavior.

Code generated by JSC2 comes complete with warnings in the comments that any changes are subject to being replaced. I was somewhat shocked when I looked under the covers at the code that JSC2 was generating for the single Web page I had been working on during the course of this review. The PageBean created by JSC2 was replete with attributes and getters/setters; calls to JSF UI component objects – including try/catch/finally blocks for JSF-specific exceptions; calls to other beans; image object configuration details; JDBC connectivity parameters, SQL statements, RowSet handlers, and more. Obviously, this kind of bleeding between application tiers or concerns could cause serious problems in the future, in terms of debugging, maintenance, and future refactoring – especially if the need ever should arise to use this generated code outside of any editor other than JSC.

Another area of concern is the ability to test each unit of code created by JSC2. Given the overt complexity of the PageBeans generated by the IDE in support of each page's functionality, you're basically sacrificing the ability to atomically test each object through JUnit.

Easy Access of Business Services

JSC2 makes short work of accessing information or business logic by providing direct access to database tables and views, EJBs, and Web services (see Figure 4). One omission is the inability to access business logic through a generic POJO as an alternative to working with an EJB for business logic; given Sun's predilection for perpetuating their dubious EJB technology, this is hardly surprising. As Sun's target market for JSC2 are SMBs, the lack of a legacy system (mainframe/midrange) or ERP integration is understandable. JSC2 comes preconfigured with drivers for DB2, Oracle, MS-SQL Server, Sybase, and PointBase; adding another SQL engine such as MySQL or PostgreSQL is quick and painless, and provided with the appropriate JDBC driver.

Leveraging the JSF spec, user interface elements dropped into a page can be bound to any available business service; this is basically a straightforward, point-and-click process in three steps. Simply right-click on the UI component, select the data resource from a drop-menu, choose which elements from that data source you'd like represented in the UI component, and you're done. JSC2 imports the appropriate data and/or logic and automatically updates the Design view with a preview of the information that will be imported.

One unfortunate consequence to the drag-and-drop design view is the slowness associated with undo; removing a component, component customization, or data binding incurs a noticeable delay. Given all of the underpinnings of code required to support the interface, this is understandable; still, I would hope in subsequent releases that Sun would work to ameliorate some of the delay.

Though JSC2 provides access to data and logic sources, and even though it is built atop NetBeans, Sun doesn't recommend using the IDE to actually build any of those business services. Instead, the concept that they're championing is that a team of Java/J2EE developers would have already created the underlying business logic objects or services to drive the functionality of your application; JSC2 is then employed by an analyst or designer to composite those services into the front end of the Web site. Sun envisions JSC2 existing within a development ecosystem, comprised of their own Sun Java Studio Enterprise for the "actual" Java developers building EJB and WS-based business services, analysts compositing these services into a JSF-based front end with JSC2, with the final work deployed on their own Sun Java System Application Server.

Understanding this helps counter some common criticisms of JSC2 – that it lacks integration with modeling tools, profiling support, etc. JSC2 doesn't exist to replace more powerful, general purpose IDEs like Eclipse, IntelliJ or NetBeans; it seeks to augment them – and therefore shouldn't have to support more powerful features like round-trip model engineering or profiling.

Visual Page Navigation

The visual page navigation editor provides an intuitive way to draw relationships between each page in your Website; this is akin to the Struts config visual editor in IBM's WSAD/RAD or Oracle's JDeveloper, for those of you who are familiar with them. Simply select a page, click and drag outward (this creates the relationship arrow, rather than selecting the arrow from a palette, which I think is more intuitive), and drop the arrow on the desired end page. JSC2 handles all of the innards required to link the pages together. Each relationship can take an optional name, letting the designer keep track of the use case associated with that activity. You can also manually edit the underlying <faces-config/> XML configuration file if necessary.

Versioning

JSC2 supports integration with CVS and Visual Source Safe code repositories; it does not support direct integration with Rational ClearCase, CA's Harvest, or the open source Subversion, though this does not necessarily prevent you from using these products with JSC2.

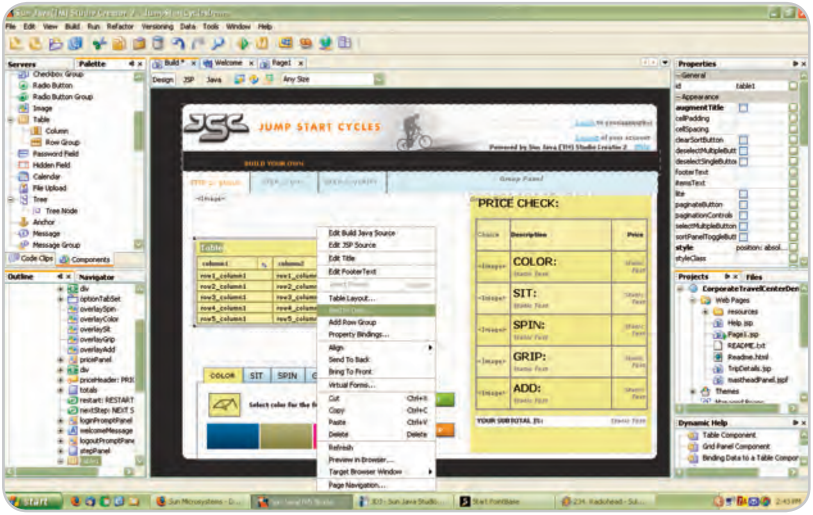


Figure 4 Bind data

Deployment

Applications can be run locally from within JSC2 via the bundled Sun Java System Application Server PE. JSC2 includes support for remote deployment to the Sun Java System Application Server or WebServer; you'll have to export your project as an .ear or .war file to deploy to other application servers, such as IBM WebSphere, BEA WebLogic, JBoss, or Apache Geronimo.

Framework Support

Sun takes the Henry Ford approach to framework support in Java Studio Creator – you can use any MVC framework you want, as long as it's JavaServer Faces.

In my conversations with Sun representatives – including luminary Craig McClanahan – about plans to support other MVC or ORM frameworks, their response was noncommittal. Essentially, the viewpoint is that basing the MVC layer on JSF, the presentation tier on JSP, accessing data through RowSets, and using EJBs for business logic all conform to J2EE specifications, even though doing so at the exclusion of alternative frameworks restricts choice. In this sense, JSC2 becomes a showcase for Sun's work in creating J2EE standards as much as a RAD tool for allowing designers to quickly build Web applications. Unfortunately, my experience is that enterprise Java development does not necessarily revolve around Sun's published best practices on the matter – for instance, there is a general disdain in the industry around using EJBs at all, and most J2EE applications I've worked with manage data through an ORM framework, such as Hibernate or JDO, not through JDBC and RowSets.

The other unfortunate consequence of this restriction is that it works to prevent adoption of JSC2 within enterprise development environments – unless by some chance the company you're working with already uses JSF, encapsulates business logic in EJBs, and has no known plans to ever change this architecture.

As a technologist, I don't let the capabilities of an IDE force my hand in determining which underlying frameworks are best suited to any project – technology decisions should be driven by several factors, primarily TCO and functional/non-functional requirements. In the event you're working in an environment where the underlying MVC framework isn't JSF, JSC2 could still be used by your analysts for rapid prototyping; there is still value in leveraging JSC2 as part of an agile process. With expanded support for Struts Classic, Struts Shale, Spring, Hibernate, JDO, Velocity, etc., I think JSC could become a vital, must-have IDE in most J2EE shops. Without expanded framework support, its role must be necessarily relegated for use only by those businesses that have already standardized on JSF, or those that adopt J2SC for rapid prototyping.

Of course, Sun's target market for JSC2 are SMBs, which don't base their purchasing decisions on architectural flexibility; their primary concern is keeping IT costs down. J2EE development has traditionally been an expensive, complex undertaking, preventing many SMBs from using Java at all. By lowering the technical bar of entry, JSC2 brings J2EE into the arena normally ruled by LAMP and .NET applications, making it a real alternative for many SMBs.

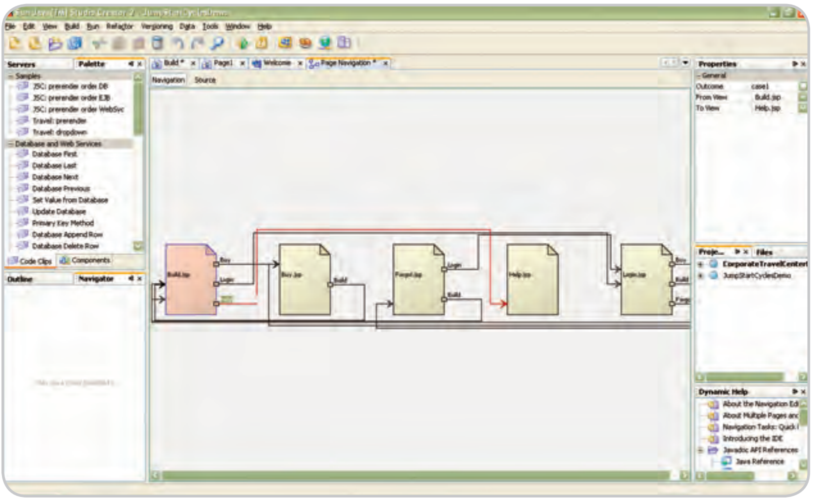


Figure 5 Page navigation editor

Plug It In

For the record, JSC2 does sport a pluggable architecture for UI components; a good example of why you'd want to use this is including AJAX functionality in your site, for which there is a download available on Sun's Java Studio Creator support site. This kind of pluggable architecture is encouraging, as is the existence of a timely library such as one supporting AJAX. It shows both Sun's acknowledgment that the technologies it makes available in its toolset aren't necessarily gospel; it also shows a willingness on Sun's part to respond in a timely manner to new technologies as they appear on the tech landscape. Good show.

Conclusion

Sun has done a successful job of creating an environment that allows J2EE applications to be quickly composited with a visual design approach, using existing business services. Java Studio Creator 2's user interface is well-thought-out, making the tool easy to learn and use, and provides a great selection of user interface components and other tools for rapidly building J2EE Websites. Building JSC2 on top of NetBeans gives users the advantages of a stable, feature-rich platform with a thriving user community. Compared to offerings from Microsoft and Macromedia, Sun's JSC2 pricing model is extremely attractive.

However, in forcing users to build on top of a specific architectural stack, Sun limits the choices available to JSC2 users and limits the kind of customers and markets that will be attracted to the tool. Rather than basing your decisions on the best technology for meeting your company's requirements, deciding whether or not to use Java Studio Creator 2 depends more on your business' budget and developer skill set. Some of the code-generation features would make agile or test-driven development practices hard to sustain. Performance issues surrounding the original JSC release unfortunately remain in the latest edition.

All told, if you're an SMB looking for a cost-effective way to rapidly and visually build your company's Website, Sun's Java Studio Creator 2 is definitely worth considering.

BONUS TRACK ADDED! (FREE $1,295 VALUE!)
"Real-World AJAX" 2-Day Seminar

SOA WebServices & Enterprise OpenSource Conferences

The #1 i-Technology Educational and

Register at... www.SOAEOSConference.sys-con.com

The **10th International SOA Web Services Edge Conference**, colocated with the First Annual **Enterprise Open Source Conference**, will take place on **June 5-6, 2006** at the historical **Roosevelt Hotel in New York City**. SOA Web Services Edge and Enterprise Open Source Conference together deliver the #1 i-technology educational and networking opportunity of the year.

This year, more than 100 distinguished conference faculty members will present **96 cutting-edge SOA, Web Services and Enterprise Open Source topics** in educational classes, panels, and keynotes in six simultaneous tracks for two information-packed days, plus a bonus track, a **"Real-World AJAX"** two-day seminar, a $1,295 value!

The conference program does not merely present a comprehensive view of all the development and management aspects of integrating a **SOA strategy** and an **Open Source philosophy** into your enterprise, its organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

According to analyst firm Gartner Group, **by 2008 more than 60 percent of enterprises will use SOA** as the guiding principle when creating mission-critical applications and processes. **"Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises,"** says Gartner analyst Yafim Natis.

Meanwhile, the **Open Source revolution** has spread into traditional areas of enterprise IT, with all major vendors and a global group of **Open Source entrepreneurs** developing the technology and delivering the solutions that are changing the way the world does business and the way governments serve their citizens.

Receive FREE WebCast Archives of all Entire Conference!

The best news for this year's conference delegates is that your "Golden Pass" registration now gives you full access to all 96 conference sessions. We will mail you the complete content from all the conference sessions in seven convenient DVDs after the live event takes place.

SOA WebServices • OpenSource • AJAX
On-Demand Webcast Archives
DVD

This on-demand archives set is sold separately for $995

Receive 8 Days Worth of Education F

FACULTY

Jeffrey Barr
Amazon
As Web Services Evangelist for Amazon.com, Jeff Barr focuses on creating developer awareness for the Amazon software platform. He has a longstanding interest in Web services and programmatic information interchange. Jeff has held development and management positions at KnowNow, eByz, Akopia, and Microsoft, and was a co-founder of Visix Software. Jeff's interests include collecting and organizing news feeds using his site, www.syndic8.com . He holds a Bachelor's Degree in Computer Science from the American University and has done graduate work in Computer Science at the George Washington University.

Israel Hilerio
Microsoft
Israel Hilerio is a program manager at Microsoft in the Windows Workflow Foundation team. He has 15+ years of development experience doing business applications and has a PhD in Computer Science.

Adam Kolawa
Parasoft
Adam Kolawa, Parasoft co-founder and CEO, is considered to be a visionary in his field. In 1983, he came to the United States from Poland to pursue his Ph.D. In 1987, he and a group of fellow graduate students founded Parasoft to create value-added products that could significantly improve the software development process. Kolawa's years of experience with various software development processes has resulted in his unique insight into the high-tech industry and the uncanny ability to successfully identify technology trends. As a result, he has orchestrated the development of numerous successful commercial software products to meet growing industry needs to improve software quality.

Jason Levitt
Yahoo!
Jason Levitt, Technical Evangelist on creating Flash-based Yahoo! Maps applications.

Duane Nickull
Adobe
As senior standards strategist for Adobe Systems, Duane Nickull is responsible for managing Adobe's participation in OASIS and UN/CEFACT, as well as ensuring that Adobe's enterprise solutions support emerging XML standards. Previously Mr. Nickull co-founded Yellow Dragon Software Corporation, a privately held developer of XML messaging and metadata management software, recently acquired by Adobe. Mr. Nickull currently serves as a vice chair of the United Nations Centre for Facilitation of Commerce and Trade (UN/CEFACT) where he oversees the United Nations Electronic Business strategy and architecture.

Bob Pasker
Azul
Bob Pasker is deputy CTO with Azul Systems. He has been designing and developing networking, communications, transaction processing, and database products for 25 years. As one of the founders of WebLogic, the first independent Java company (acquired by BEA Systems in 1998), he was the chief architect of the WebLogic Application Server. Bob has provided technical leadership and management for numerous award-winning technologies, including the TribeLink series of routers and remote access devices, and the TMX transaction processing system. Bob graduated magna cum laude and Phi Beta Kappa from San Francisco State University and holds a Masters degree from Brown University.

Brian Behlendorf
CollabNet
Brian Behlendorf founded CollabNet, with O'Reilly & Associates, in July 1999. The company provides tools and services based on open source methods. Before launching CollabNet, Behlendorf was co-founder and CTO of Organic Online, a Web design and engineering consultancy located in San Francisco. During his five years at Organic, Behlendorf helped create Internet strategies for dozens of Fortune 500 companies. During that time, he co-founded and contributed heavily to the Apache Web Server Project, co-founded and supported the VRML (Virtual Reality Modeling Language) effort, and assisted several IETF working groups, particularly the HTTP standardization effort.

Marc Fleury
JBoss
Born in Paris in 1968, Marc Fleury got his Ph.D in physics from the Ecole Polytechnique in Paris. He started in Sales at Sun Microsystems France and then moved to the US where he worked on early java enablement of SAP at SAPLabs. Marc started the JBoss project in 1999. An ex-Lieutenant in the paratroopers, Marc holds a degree in Mathematics from the Ecole Polytechnique, a master in Theoretical Physics from the Ecole Normale ULM and was a visiting scientist at MIT during his thesis. Marc's research interest focuses on aspect oriented middleware.

Andy Astor
EnterpriseDB
Andy is President and CEO EnterpriseDB, the world's leading enterprise-class, open source database company. Previously, Andy was vice president webMethods, leading the company's open source, standards, and Web services agendas. Andy was elected twice to the Board of Directors of the Web Services Interoperability Organization (WS-I), and led WS-I's marketing efforts. Prior to joining webMethods, Andy was vice president at D&B, where he led worldwide development of all on-line products. His work at D&B included the development and launch of one of the earliest commercial Web services.

Mike Milinkovich
Eclipse.org
Mike Milinkovich has held key management positions at Oracle, WebGain, The Object People, and Object Technology International Inc. (which subsequently became a wholly-owned subsidiary of IBM), assuming responsibility for development, product management, marketing, strategic planning, finance and business development. Mike earned his MS degree in information and systems sciences and a bachelor of commerce degree from Carleton University in Ottawa, Canada.

Peter Yared
ActiveGrid
Peter Yared is the founder and CEO of ActiveGrid. Most recently, he was CTO of Sun Microsystems's Liberty Network Identity initiative. Mr. Yared was also CTO of Sun Microsystems Application Server Division. Before its acquisition by Sun, Mr. Yared served as CTO of NetDynamics, which pioneered the then-leading J2EE application server. Earlier, Mr. Yared was founder and CEO of JRad Technologies, an enterprise Java company acquired by NetDynamics. Additionally, Mr. Yared was Chief Architect of client/server products at object-oriented tool maker Prograph International and the architect of several mission-critical systems deployed by U.S. government agencies and the GED Testing Service.

David Temkin
Laszlo
David Temkin is Chief Technology Officer of Laszlo Systems, Inc. In this role, he has positioned the company to become the next technology standard for rich Internet applications. Under his direction, Laszlo developed its patent-pending open-source product suite and extended operations to both coasts of the United States. Before founding Laszlo, Temkin was senior director of engineering at Excite@Home where he led a team of 55 engineers, designers and technical writers responsible for developing the company's consumer software. Prior to Excite@Home, Temkin was an engineering manager in the Newton division at Apple Computer and developed enterprise software at EDS.

Kevin Hakman
TIBCO
Kevin Hakman is Co-founder, TIBCO General Interface, TIBCO Software Inc. Prior to TIBCO General Interface, he was the co-founder of Versalent Inc. a leading provider of enterprise client technology. Prior to Versalent, he founded a series of successful emerging Internet technology and ecommerce ventures. He has also written for eBusiness Journal and HotWired.

Coach Wei
Nexaweb
Coach Wei currently serves as CTO for Nexaweb, which develops the leading XML-based rich client technology platform for building and deploying Enterprise Internet Applications. Previously, he played a key role at EMC Corporation in the development of a new generation of storage network management software. Coach is a graduate from MIT, holds several patents, and is an industry advocate for the proliferation of open standards.

Luis Derechin
JackBe
Luis Derechin is CEO and Co-Founder of JackBe. Mr. Derechin has over 12 years of entrepreneurial and management experience. He has been part of the founding team of successful startups, including a catalogue retail company that achieved $15M in sales.

Jouk Pleiter
Backbase
Jouk Pleiter is the CEO of Backbase, a leader in the field of Rich Internet Applications and AJAX development software. Backbase's clients include ING, ABN AMRO, TNT, KPN, Comsys and Heineken. Backbase operates globally with offices in San Mateo (North America) and Amsterdam (Europe). Since 1995, Jouk has been an entrepreneur: he founded three successful Software companies. Prior to Backbase, Jouk was part of the founding team at the web content management company Tridion, where he led the product management operations, and was driving the company's efforts to become a leader in the European WCM software market.

Networking Opportunity of the Year!

SOA WebServices Edge 10th International conference

2006 ENTERPRISE> OPENSOURCE CONFERENCE

The Roosevelt Hotel
JUNE 5-6, 2006
Roosevelt Hotel / New York, NY
For Sponsorship Information ► Call 201 802-3022

Early Bird Price (Before March 31, 2006)
Incl. Golden Pass to Both Conferences* $1,295

Discounted Price (After March 31, 2006)
Incl. Golden Pass to Both Conferences* $1,495

Conference Price (On June 5-6, 2006)
Incl. Golden Pass to Both Conferences* $1,695

*BONUS: "Real-World AJAX" Two-Day Seminar!

Attention Sponsors:
A Forum will be available to display leading Web services, OpenSource, and AJAX products, services, and solutions.

TOPICS INCLUDE...

SOA Web Services Edge:
> Transitioning Successfully to SOA
> ebXML
> Orchestration
> The Business Case for SOA
> Interop & Standards
> Web Services Management
> Messaging Buses and SOA
> SOBAs (Service-Oriented Business Apps)
> Delivering ROI with SOA
> Java & XML Web Services
> Security
> Systems Integration
> Sarbanes-Oxley
> Business Process Management

Enterprise Open Source:
> Open Source Licenses
> Open Source & E-Mail
> Databases
> ROI Case Studies
> Open Source ERP & CRM
> Open-Source SIP
> Testing
> Grid Computing
> Open Source Middleware
> LAMP Technologies
> Professional Open Source
> Open Source on the Desktop
> Open Source & Sarbanes-Oxley
> IP Management

AJAX User Experience:
> Rich Internet Applications
> Flash-AJAX Integration
> Developing Interfaces
> AJAX on Rails
> Best Practices
> Enterprise AJAX
> Technology Demos
> Lessons from the Frontline
> Bringing Desktop Apps to the Web
> AJAX Power Panel
> and more...

SYS-CON EVENTS www.EVENTS.sys-con.com

Packed into 2 Information-Filled Days!

Six New Spec Leads Reach JCP Stardom

Onno Kluyt

The red carpet isn't rolled out just on Hollywood Boulevard this time of the year. It happens in our community too. Six new spec leads recently reached stardom and I'm inviting you to meet them in this month's column. Exceptional spec lead performance gets noticed by the community and acknowledged through awards and the Star Spec Leads distinction as a key ingredient for an effective and smooth JSR development process. It's already an established tradition for the community that occurs two or three times a year to raise its most successful spec leads to stardom. Folks like Danny Coward, Pierre Gauthier, Janna Majakangas, Éamonn McManus, Antti Rantalahti, and Bill Shannon have set the bar yet higher with the timely and quality delivery of their JSRs. I'm delighted to congratulate them on behalf of the community and wish them a good year ahead!

Danny Coward of Sun Microsystems has been involved with the JCP program in a variety of roles from representative and observer to expert and spec lead. He has already led three JSRs and his six years with the community have provided him with valuable experience as spec lead. Danny feels strongly that the longevity of Java APIs and platforms is coming about because of the breadth and depth of community participation. "One perspective is not enough," he says.

Chair of the Operation Support Systems through Java (OSS/J) Architecture Board, editor of the OSS/J Design Guidelines, spec lead and implementation team lead, all these describe *Pierre Gauthier* of MetaSolv Software. But what puts Pierre in the echelon of Star Spec Leads more than anything else is that he took the time to mentor numerous individuals over the past few years, transferring the knowledge and expertise he gained as spec lead and driver of the OSS/J effort through the community.

Onno Kluyt is the director of the JCP Program Management Office, Sun Microsystems.

onno@jcp.org

Jaana Majakangas of Nokia Corporation has quickly acquired her spec lead competencies and put them to work effectively. She attributes part of her success to excellent mentors – other spec leads from Nokia.

She finds the role and work of spec lead deeply satisfying. "I like my current role where we create a standard in some area, and it is interesting to see how it is then implemented in actual products. This gives us feedback on how we have succeeded," she says.

Éamonn McManus of Sun Microsystems is a veteran spec lead with a JSR portfolio to envy. Nicknamed "the JMX guru," he has a reputation for leading JSRs transparently. He has even turned to blogging to expand communication beyond traditional JSR channels. He created the Java Management Extensions (JMX) blog to get as much feedback and input as possible from the community.

For *Antti Rantalahti* of Nokia Corporation, the spec lead–expert group communication is a top priority. He says, "…quick response time and correctness are part of being professional." Experts should know that they have all the same information as the spec lead has and feel that they are the ones who can and will do the work. Antti is pleased with being named a Star Spec Lead, saying, "I see it as a merit for myself, my company, and the Expert Group."

Bill Shannon's (Sun Microsystems) participation in the JCP is synonymous with the Java Platform, Enterprise Edition (Java EE), the development and maintenance of which he has led and overseen through all its stages. He was awarded the distinction of *Most Outstanding Spec Lead for Java Standard Edition/Enterprise Edition 2005* (http://jcp.org/en/press/news/awards/2005award_winners) for his technical acumen, his ability to build consensus among Experts, and his efficiency in guiding the specification development process.

For all of these reasons and more, the six have received the distinction of Star Spec Leads. If you want to find out more about them and the JSRs they lead, visit http://jcp.org/en/press/news/star.

Before I sign off, a word about the 2006 JCP training sessions – great opportunities for anyone with an interest in Java technology to familiarize themselves with the premier Java standards body in the industry. The JCP training season kicked off last month with its first session in the Bay Area. The next one is planned for May at JavaOne in San Francisco. A couple of things these sessions will deliver to you:

- Information about the JCP, what it is all about, and how you can leverage it to best support your Java endeavors
- Familiarity with the JCP JSR review process via interactive exercises in which you can play spec lead
- Tips on spec leads' and expert groups' pet peeves so that you can stay away from common pitfalls and become a successful player in the community
- Walk the talk of standards development through the JCP program
- Become an information resource for your teams as to how the JCP can be of help with your team's projects

Check the 2006 JCP Calendar at jcp.org periodically for the latest info regarding training dates and locations and sign up early for the session closest to your area. And stay tuned to news about the jcp.org home, which promises feature enhancements to make your experience with the site a better one.

Final tips: the JSRs updated at the time of writing (February) were JSR 288, Adaptive Java ME System API from Aplix Corporation; JSR 62, Personal Profile Specification; JSR 109, Implementing Enterprise Web Services; JSR 56, Java Network Launching Protocol and API; JSR 154, Java Servlet 2.5; JSR 270, Java SE 6 Release Contents; JSR 252, JavaServer Faces 1.2 from Sun Microsystems; and JSR 135, Mobile Media API from Nokia Corporation.

Until next time.

Software FX
Bringing Your Data, Business & Strategy Into Focus.

New! version 6.2
Now Includes Maps!

All New Flavors... Same Great Taste of Java!

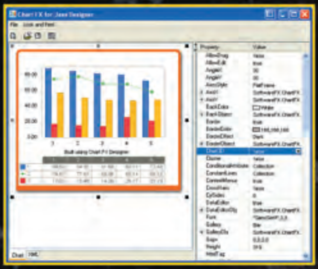


Chart FX for Java Designer

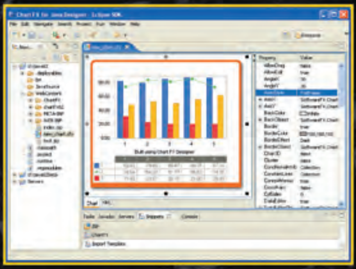


Chart FX for Java - Eclipse

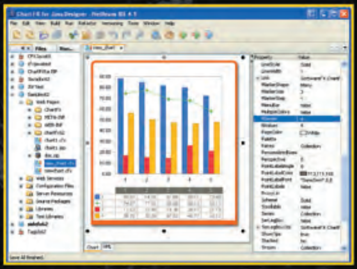


Chart FX for Java - NetBeans

Chart FX for Java 6.2

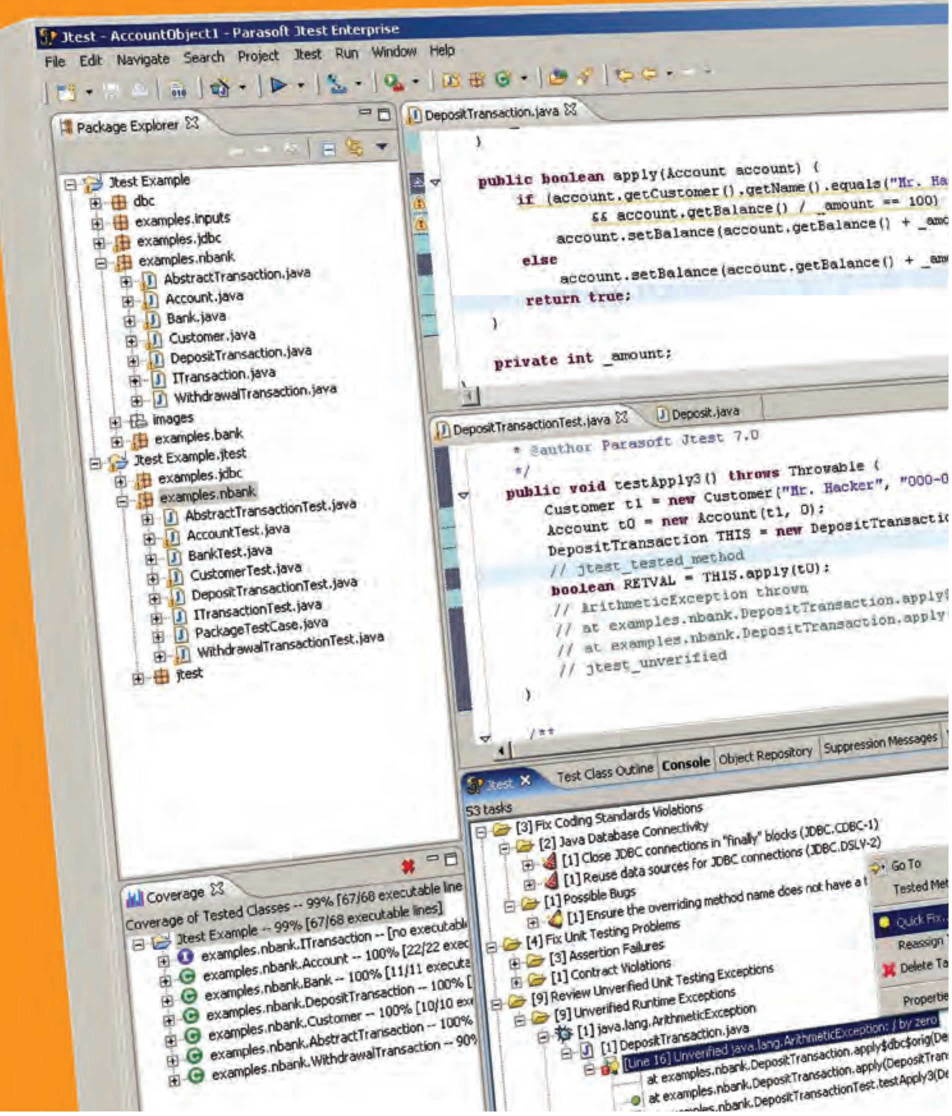
Now offers seamless integration within Eclipse and NetBeans, Chart FX for Java 6.2 brings New features include Smart & dynamic axis labeling, conditional marker and axis attributes, extended URL linking, multiple chart panes and robust annotation objects. One of the most dynamic new features is the inclusion of the Chart FX Maps Extension as a permanent fixture within Chart FX for Java. ➤

Chart FX

US: (800) 392-4278 • UK: +44 (0) 8700 272 200 • Check our website for a Reseller near you! www.softwarefx.com

Code in quality. Test out bugs.

Take control of your code through Automated Unit Testing and Code Analysis...



Jtest - AccountObject1 - Parasoft Jtest Enterprise

```
Package Explorer
Jtest Example
  dbc
  examples.inputs
  examples.jdbc
  examples.nbank
    AbstractTransaction.java
    Account.java
    Bank.java
    Customer.java
    DepositTransaction.java
    ITransaction.java
    WithdrawalTransaction.java
  images
  examples.bank
Jtest Example.jtest
  examples.jdbc
  examples.nbank
    AbstractTransactionTest.java
    AccountTest.java
    BankTest.java
    CustomerTest.java
    DepositTransactionTest.java
    ITransactionTest.java
    PackageTestCase.java
    WithdrawalTransactionTest.java
  jtest
```

```
DepositTransaction.java
    }

    public boolean apply(Account account) {
        if (account.getCustomer().getName().equals("Mr. Ha
                && account.getBalance() / _amount == 100)
            account.setBalance(account.getBalance() + _amo
        else
            account.setBalance(account.getBalance() + _amo
        return true;
    }

    private int _amount;
```

```
DepositTransactionTest.java   Deposit.java
     * @author Parasoft Jtest 7.0
     */
    public void testApply3() throws Throwable {
        Customer t1 = new Customer("Mr. Hacker", "000-0
        Account t0 = new Account(t1, 0);
        DepositTransaction THIS = new DepositTransacti
        // jtest_tested_method
        boolean RETVAL = THIS.apply(t0);
        // ArithmeticException thrown
        // at examples.nbank.DepositTransaction.apply$
        // at examples.nbank.DepositTransaction.apply(
        // jtest_unverified
    }

    /**
```

```
Coverage
Coverage of Tested Classes -- 99% [67/68 executable line
Jtest Example -- 99% [67/68 executable lines]
  examples.nbank.ITransaction -- [no executabl
  examples.nbank.Account -- 100% [22/22 exec
  examples.nbank.Bank -- 100% [11/11 executa
  examples.nbank.DepositTransaction -- 100% [
  examples.nbank.Customer -- 100% [10/10 exe
  examples.nbank.AbstractTransaction -- 100%
  examples.nbank.WithdrawalTransaction -- 90%
```

```
Jtest   Test Class Outline   Console   Object Repository   Suppression Messages
53 tasks
[3] Fix Coding Standards Violations
  [2] Java Database Connectivity
    [1] Close JDBC connections in "finally" blocks (JDBC.CDBC-1)
    [1] Reuse data sources for JDBC connections (JDBC.DSLV-2)
  [1] Possible Bugs
    [1] Ensure the overriding method name does not have a t
[4] Fix Unit Testing Problems
  [3] Assertion Failures
  [1] Contract Violations
[9] Review Unverified Unit Testing Exceptions
  [9] Unverified Runtime Exceptions
    [1] java.lang.ArithmeticException
      [1] DepositTransaction.java
        [Line 16] Unverified java.lang.ArithmeticException: / by zero
          at examples.nbank.DepositTransaction.apply$dbc$orig(De
          at examples.nbank.DepositTransaction.apply(DepositTran
          at examples.nbank.DepositTransactionTest.testApply3(De
```

PARASOFT®

Jtest®

- Automatically analyzes your code, applying 500+ Java coding best practices that surface poor code constructs affecting reliability, security and performance.
- Automatically unit tests your Java code evaluating code behavior and exposing unexpected exceptions, boundary condition errors and memory leaks.
- Generates extendable, high coverage test cases in JUnit format that can be quickly turned into libraries of reusable test assets.
- 100's of Quick Fix options for fast, accurate resolution of errors.
- Integrated code coverage analysis ensures that your code is thoroughly tested.
- Team collaboration support allows you to define, distribute and enforce code quality standards across entire development teams.

Stop bugs before they start:

Jtest automates the valuable, yet often tedious tasks of code reviews and unit testing allowing development teams to adopt a "test as you go" strategy that promotes testing code as it is developed so that quality is built into the code from its inception and bugs eliminated before they infect the rest of the code base.

The result? Cleaner, safer, more reliable and consistent code. Reduced code rework. Faster, more predictable release cycles. Reliable applications and happier, more productive end-users. **To try Jtest today, call 888-305-0041 (x-3501) or go to www.parasoft.com/JDJmagazine.**

Parasoft Jtest clients include: Lehman Bros., Cisco, Disney, Raytheon, ADP, Sabre Holdings, GE and more.

PARASOFT®

We make software work.™

Automated Error Prevention™