Advanced Java 3

Sun Stewardship

Java is controlled by Sun, which is not as appealing as control by a non-profit such as the W3C

However, there is precedent -- C and C++ were controlled by AT&T without harm

The history of Sun's guidance of Java in the last 7 years has been pretty prudent and reasonable, so they have earned some trust.

Hopefully, Sun is happy to develop Java as OS agnostic platform -- app writers may code to Java, and their apps will run everywhere.

There is a stereotype that Sun is run by engineers, and Java may be an outgrowth of that.

Microsoft has the most to lose from applications not being OS specific

By the same token, every other vendor (Sun, IBM, Oracle, ...) benefits from a Java as a healthy, non Microsoft-specific platform for development. The EBM "Everyone But Microsoft" alliance.

Java Open Development

How to find about Java future directions?

Sun actually does Java development very much out in the open

Get a free account on java.sun.com, then...

1. Read the top 25 bugs on the buglist
2. Read the top 25 Request For Enhancements (RFE)

   You can vote for your favorite issues.

Go the JCP (Java community process) site (http://www.jcp.org/) and look at the proposals in the various stages of development. You can observe movements and arguments for a couple years as features make their way into the language.

Things which are showing up now in Java have been visible in some form or other above for years.

Don't be discouraged by the complaining tone in the forums above -- Sun gets a lot of credit for making their bug database and its arguments public. No complex system can exist in the open like that without a lot of flaming, complaining, and posturing (this seems to be a truism of online communities).

Java Development Themes

Major themes in Java...

Backward compatible -- old code continues to run, even as new features are added

Portable -- write once, run everywhere

Large library -- more and more off-the-shelf features get added to the library

Elegant/Structured style vs. "quick 'n dirty" like Perl

Slow progress -- Sun's guidance has tended to be slow and prudent

Sun seems to have a bias toward the "Elegant, Full-featured" solution instead of the "Simple but fast" solution. Time will tell if this is a good strategy. I suspect it is, considering the pace of hardware improvement vs. 20+ year lifetime of a popular computer language

Java Niches -- Server vs. Client

Niche: server-side internet apps -- Java is very popular here already -- portable, secure, programmer efficient -- show well in this niche

"Business logic" applications using Java and its JDBC library to connect to the database and fiddle around with the data. Note: possibly no GUI, just strings, ints, dates, etc.

Niche: "custom" applications

A custom GUI application that is part of a larger custom system -- e.g. the "View Order Status" application used by the foo.com customer service people

Possible niche: Client side java

Possible niche: Small devices -- palm pilots, TVs, ...

J2EE

Java 2 Enterprise Edition (J2EE)

J2SE is "standard" desktop java, and J2ME is the "micro" java for small devices

"Enterprise" is the niche of large, corporate information technology (IT) projects, typically featuring databases. web sites, business processes, ...

There's a lot of money spent here.

Java is doing very well in this niche.

J2EE is standard by which java objects interact with each other

If the various parts of an IT solution are J2EE compliant, then it helps avoid vendor lock-in, since the parts of more interchangeable.

In reality, J2EE is fairly complex, so it adds some complexity to a project.

HTML Forms Are A Hack

Currently, almost all net services, (e.g. Amazon, yahoo email, ...) are presented through HTML forms.

This has the huge advantage of compatibility -- it works with most any client OS, and such platform-independent compatibility has been the key ingredient in the growth of the internet.

Note that the Internet did not develop exchanging proprietary .doc files, even though 90% of users have MS Word -- the Internet explosion really kicked in with 100% portable, standard formats such as HTTP and HTML.

However, HTML forms do not present a great interface -- the user sees a state, they can click a button, there is a 2 second delay, and they see the next state.

Contrast this to a real GUI program -- you move the mouse or scroll a list, and 1/100th a of a second later you get the visual feedback.

We are so used to HTML forms, we have grown blind to how lame they are for constructing a good UI.

Future: Real Client GUI

Imagine Amazon client program
Runs on the client side
Communicates back to the server as needed
Presents a responsive GUI to the client -- lists, text fields, selections etc.
Still limited by networking speed, but can be far better than the HTML form

Applets

Run in a security "sandbox" in the browser -- prevent the applet from touching the local file system, etc.
Applets have not caught on too much
Performance problems
Running inside the browser created inevitable reliability problems
    Microsoft is not, shall we say, enthusiastic about making applets work correctly in the browser.
Original applets used AWT
With the latest Java 1.2 or later installed on a machine, the Swing JApplet may be used -- the browser must be set up to support Java.
Sun's "java plugin" is a browser plugin that provides applet support.

Jar files

.jar file is an archive file that contains directories of .class files + misc images, sounds, and other support files.
Double click on the .jar runs the application (works on windows, Solaris, and MacOSX)
Users need to install Java first -- the Java Runtime Environment from Sun (JRE)
Code does not run in a "sandbox"
It's easy to package your java application into a .jar file -- then you can distribute it as simply as a PDF. Users just download the file and double click it.

Java Web Start

Replacement for applets and jar files
http://java.sun.com/products/javawebstart/
Client installs the JWS loader on their machine once (like installing Acrobat). Installing the Java Runtime Environment installs JWS automatically.
Package app in a .jar
Put a link to the app on a web page -- when the user clicks the link, JWS downloads the appropriate .jar files if needed and launches the application.
The convenience of an applet (access through URLs) but without the problems of running in the browser.
For example, the little DiceMachine java application I wrote at
http://www-cs-students.stanford.edu/~nick/dice/
can be accessed through Java Web Start and as a plain .jar file.
Here's the .jnlp file for DiceMachine -- it's based on the Sun example...

```
<?xml version="1.0" encoding="utf-8"?>
<!-- trying to make a simple, working jnlp for DiceMachine.jar -->
```

```
<jnlp
   spec="1.0+"  <!-- can be omitted -->

   <!-- where other things are found -->
   codebase="http://www-cs-students.stanford.edu/~nick/dice/"

   <!-- where the .jnlp file itself lives -->
   href="dice.jnlp"
>


<information>
   <title>DiceMachine</title>
      <vendor>Nick Parlante</vendor>
         <homepage href="http://www-cs-students.stanford.edu/~nick/dice/"/>
        <description kind="one-line">Dice rolling application</description>
        <description kind="short">Dice rolling application that graphs the
distribution or rolls. Perfect for the game Settlers of Catan.</description>


        <icon href="dice-small.jpeg"/>

   <!-- this allows the app to be run without a net connection -->
        <offline-allowed/>
</information>

<resources>
   <j2se version="1.2+"/>
        <jar href="DiceMachine.jar" main="true" download="eager" />
</resources>

<!-- what's the main class -->
<application-desc main-class="DiceMachine"/>

</jnlp>
```

Unsigned code runs in a sandbox

The client just downloads the .jnlp file which points to enough info for the client to download and run the java code.

Can run with or without a net connection once downloaded.

Can check for updates automatically

The point: You send someone just a URL, and they can just click it to run the program on their machine. Updates can happen automatically.

Will JWS Catch On?

Like Flash catching on -- chicken-and-egg problem that works best if many clients have it pre-installed.

This will be hard since Microsoft controls the dominant OS and browser, and Microsoft hates Java

Enterprises love it internally -- easy way to distribute and update little custom apps -- just send out the URL

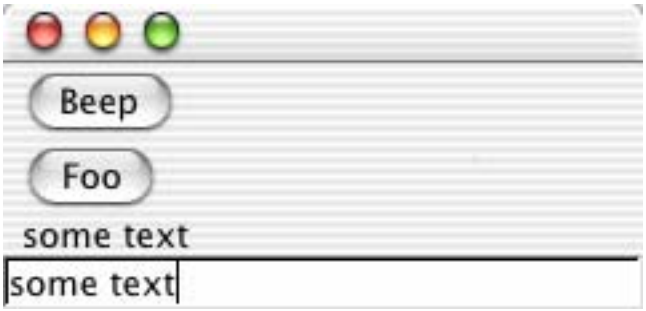
J2ME/MIDP

Mobile Information Device Profile

Allow you to write small apps that work on cell phones, Palm, Windows CE, ...
http://java.sun.com/j2me/
http://java.sun.com/products/midp/
Write a "midlet" that runs on a small device with limited GUI facilities
Works on PalmOS 3.5
Subset of Java for small devices -- not as heavyweight as Swing
Also, Connected Limited Device Configuration -- CLDC -- phones, etc.
Many vendors are excited about the "small device" space -- a new frontier vs. the desktop
Many cell phones now support this -- java is used to construct the internal "applications" of the phone (phone log, etc.)
May also be used for downloadable games, etc.
Some providers let you install your own MIDP apps on the phone, while some have a "captive" strategy which only allow java apps approved by the service provider. The lesson of the Internet is that the captive strategy tends to lose to the wide-open strategy.

New 1.4 EventHandler Style

Removes the need for creating lots of ActionListener objects
Instead, use EventHandler.create(...) to make a little handler -- specify what object to notify, and what message to send
EventHandler uses introspection heavily
In the future, the Sun BeanBuilder (not yet released) project may allow you to construct your GUI like a draw program.
BeanBuilder can write out the EventHandler glue for you
http://java.sun.com/j2se/1.4/docs/api/java/beans/EventHandler.html




```
// Swing2
/*
 Demonstrates a little use of the EventHandler class.
*/
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;

import java.beans.*;

public class Swing2 extends JFrame {
   JTextField field;
   JLabel label;
```

```
   public void beep() {
      System.out.println("beep!");
   }

   public Swing2() {
      JComponent content = (JComponent) getContentPane();
      content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));

      JButton b1 = new JButton("Beep");
      content.add(b1);
      b1.addActionListener(
         // Send msg to: this
         // Message to send: beep
         (ActionListener)EventHandler.create(ActionListener.class, this, "beep")
      );


      JButton b2 = new JButton("Foo");
      content.add(b2);
      b2.addActionListener(
         (ActionListener)EventHandler.create(ActionListener.class, this, "foo")
      );
      // When clicked, this looks for a foo() message, which does not exist

      JLabel label = new JLabel("label");
      content.add(label);

      field = new JTextField(20);
      content.add(field);

      field.addActionListener(
         // send msg to: label
         // msg to send: setLabel
         // value to send: event.getSource().getText()
         (ActionListener)EventHandler.create(ActionListener.class, label, "text",
"source.text")
      );

      pack();
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true);

   }

   public static void main(String[] args) {
      new Swing2();
   }
}
```

Java Beans

Actually really simple -- like an ADT

Bean

- Has an empty ctor
- Has getFoo and setFoo methods for each of its public properties

Unit of exchange

- Module A wants to package information for others to use
- Set up a "bean" class that uses getters and setters in the standard way
- Then other programmers can use it easily

Bean tools

Tools can understand the create/get/set nature of the bean to allow people to manipulate it without writing code.

Old Serialization

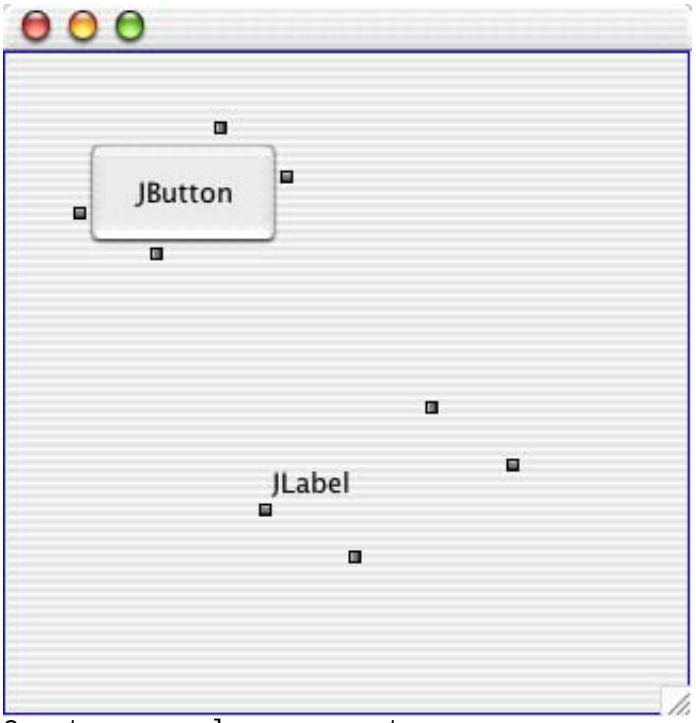
Design -- how to you serialize off a Java class?
Old serialization: write out its ivars
Problem: what if the class changes impl?

New, XML "Persistence"

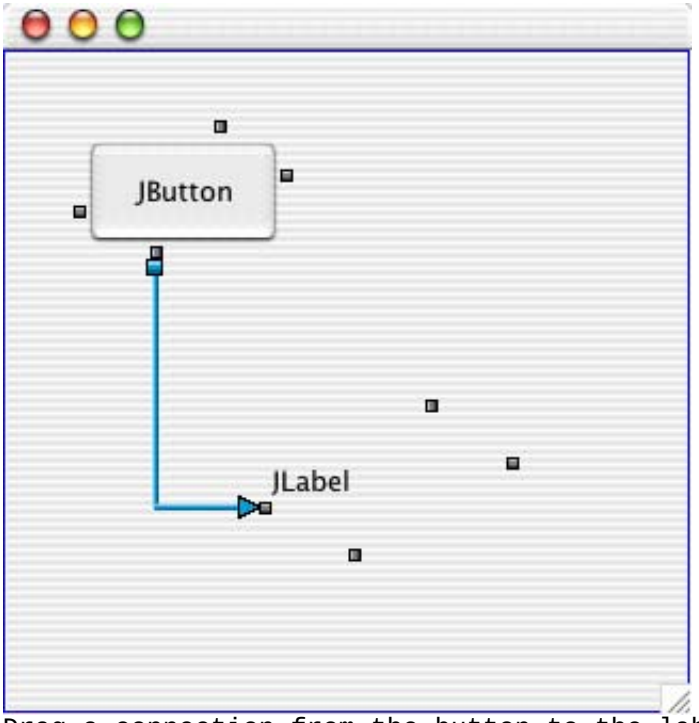
http://java.sun.com/j2se/1.4/docs/guide/beans/index.html
http://java.sun.com/products/jfc/tsc/articles/persistence/
http://java.sun.com/products/jfc/tsc/articles/persistence2/
http://java.sun.com/products/jfc/tsc/articles/persistence3/
Only serialize state that is accessible through public get/set methods (the "bean" view of an object)
This is the technology that underlies the new GUI/Bean/XML layout editor technology (not yet released)
Be smart about constructor defaults...
To serialize Foo f...
1. Construct Foo s;
2. Compute what setXXX() messages are necessary so that s looks like f.
3. Record the arguments for the ctor/setXXX sequence -- that **is** the persistent form of f
Advantages: totally independent of implementation. In fact you could serialize as Foo, and then read back into a different class, say Bar, so long as Bar had the same public ctor/get/set semantics as Foo.

GUI Construction -- Bean Builder (1.4+)

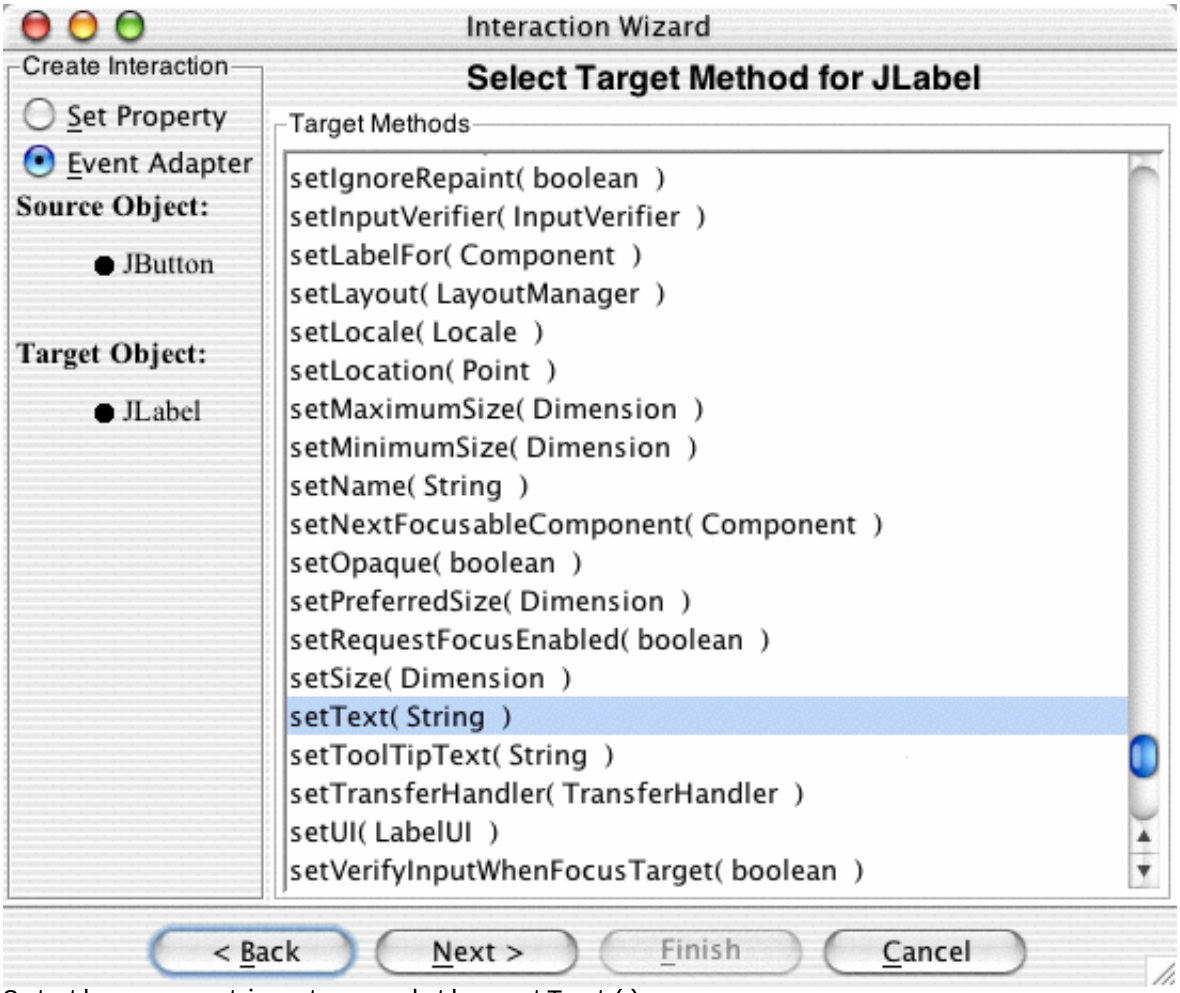
The "BeanBuilder" app lets you draw/edit your GUI.
BeanBuilder is in beta -- it's not done yet.
http://java.sun.com/products/javabeans/beanbuilder/index.html
When you're satisfied, you serialize (dehydrate) down the collection of GUI objects
At run-time, the objects are read in to memory (rehydrated) to re-create the whole GUI and all the listener connections.



Create a couple components



Drag a connection from the button to the label



Set the connection to send the setText() message

Other Java Areas...

RMI

Distributed processing -- make objects that are on "remote" JVMs (on other machines) look like ordinary objects in your local JVM.
Depends on portability to send bytecode around the network.
Depends on serialization standard to move objects around the network.
Depends on "sandbox" security to run the inbound code safely.
Performance is a little slow, since it depends on serialization machinery, however the network itself probably represents most of the delay, so who cares.

JINI

"Federation" layer allowing little devices to cooperate. Everybody thinks this niche is going to be the next big thing, but it doesn't really exist yet.
Example --

> Your CD player sends its GUI code (java bytecode) to your palm pilot. The GUI code understands the CD player. On the Palm, the GUI code presents all the songs that are on the CD player, and you use the GUI to communicate back to the CD player. Your Palm and your CD player interact without being pre-designed for each other by exchanging code.

JDBC

Standard layer to interact with a database...
Write queries...get results

Java Servlets

Used on the server side code ("business logc") for a web application
This is a 2nd generation technology -- perl CGI's were the first generation
(Take CS193i)

Java Server Pages (JSP)

Related to servlets
A more lightweight way to encode an HTML page that calls little bits of java code at strategic points.
Similar to PHP, ASP
(Take CS193i)

Java 2d / Java 3d / Imaging

Image IO -- package for manipulating image data specifically
Advanced Imaging -- manipulatio of large bitmap images (http://java.sun.com/products/java-media/jai/)
Scalable Vector Graphics (SVG) -- W3C standard for vector graphics(similar to PDF) -- SVG will be very useful if it catches on. The Batik project links SVG and java (http://xml.apache.org/batik/)