JavaOne℠

Sun's 2002 Worldwide Java Developer Conference

Combining Doclets with JDBC™ and JSP™ Technologies to Deliver the Next-Generation Documentation System for the Java™ Platform

Eitan Suez
President
UptoData, Inc.

Primary Purpose

Study the architecture and design of a 3-Tier web application for Java™ API Documentation

Learning Objectives

- As a result of this presentation, you will be able to:
  - Learn techniques for architecting web applications
  - Witness firsthand the power and flexibility of 3-Tier, Java™ technology-based applications
  - Gain further insight into Doclet, Java Servlet, and JSP™ APIs

About The Speaker

- Eitan Suez has been programming in the Java™ programming language since 1995 (1.0 beta)
- Eitan is a Sun Certified Programmer for the Java™ Platform
- Eitan is the architect and author of *dbdoc*, a system for Java™ Technology API documentation

Question to Audience

Do you have a
javadoc feature wish list?

My Wish List

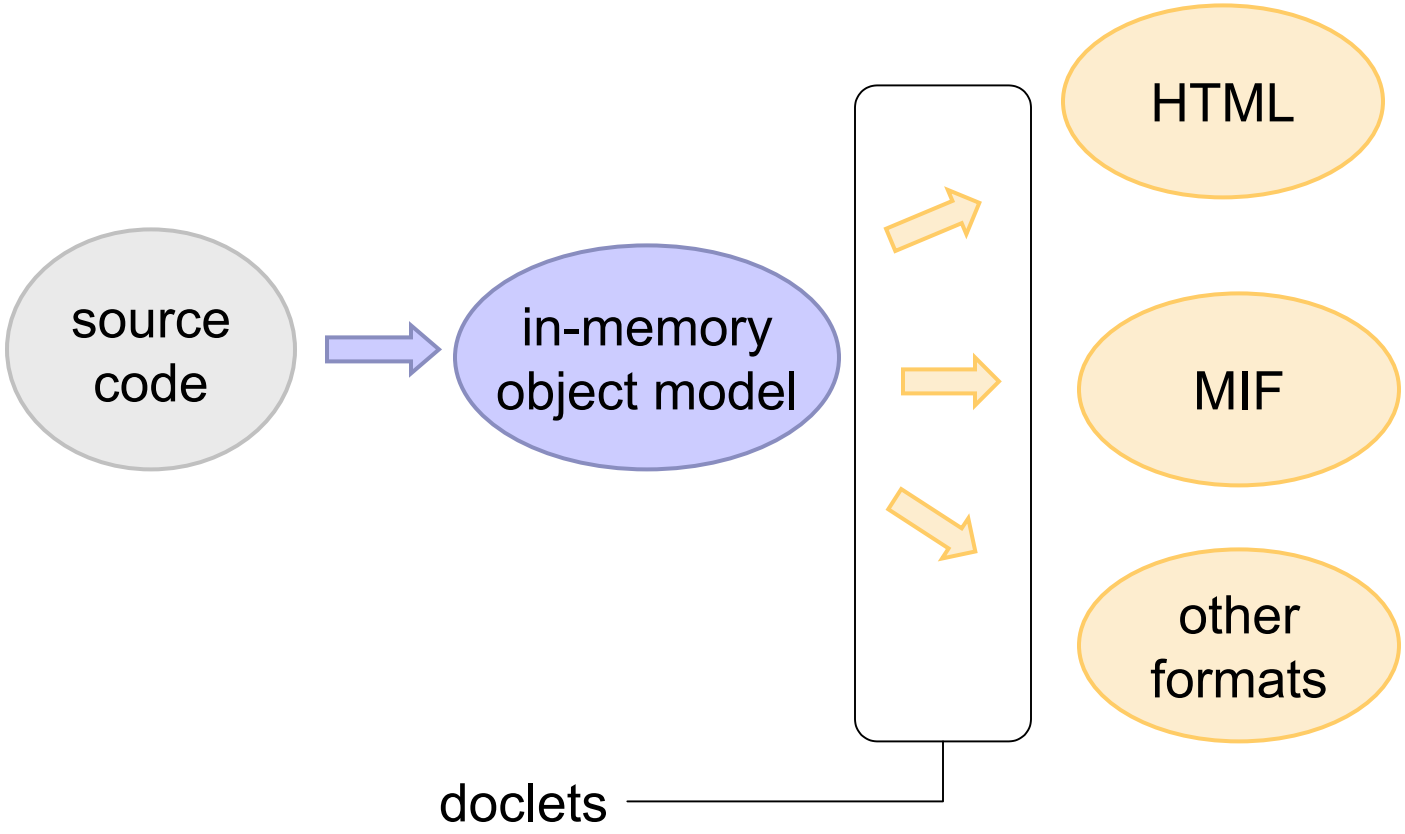
1. All APIs accessible from one location
2. Color-coded programming elements
3. Search capabilities
4. A richer User Interface
5. Navigation enhancement
6. Drill down to source code

Agenda: realizing wish list

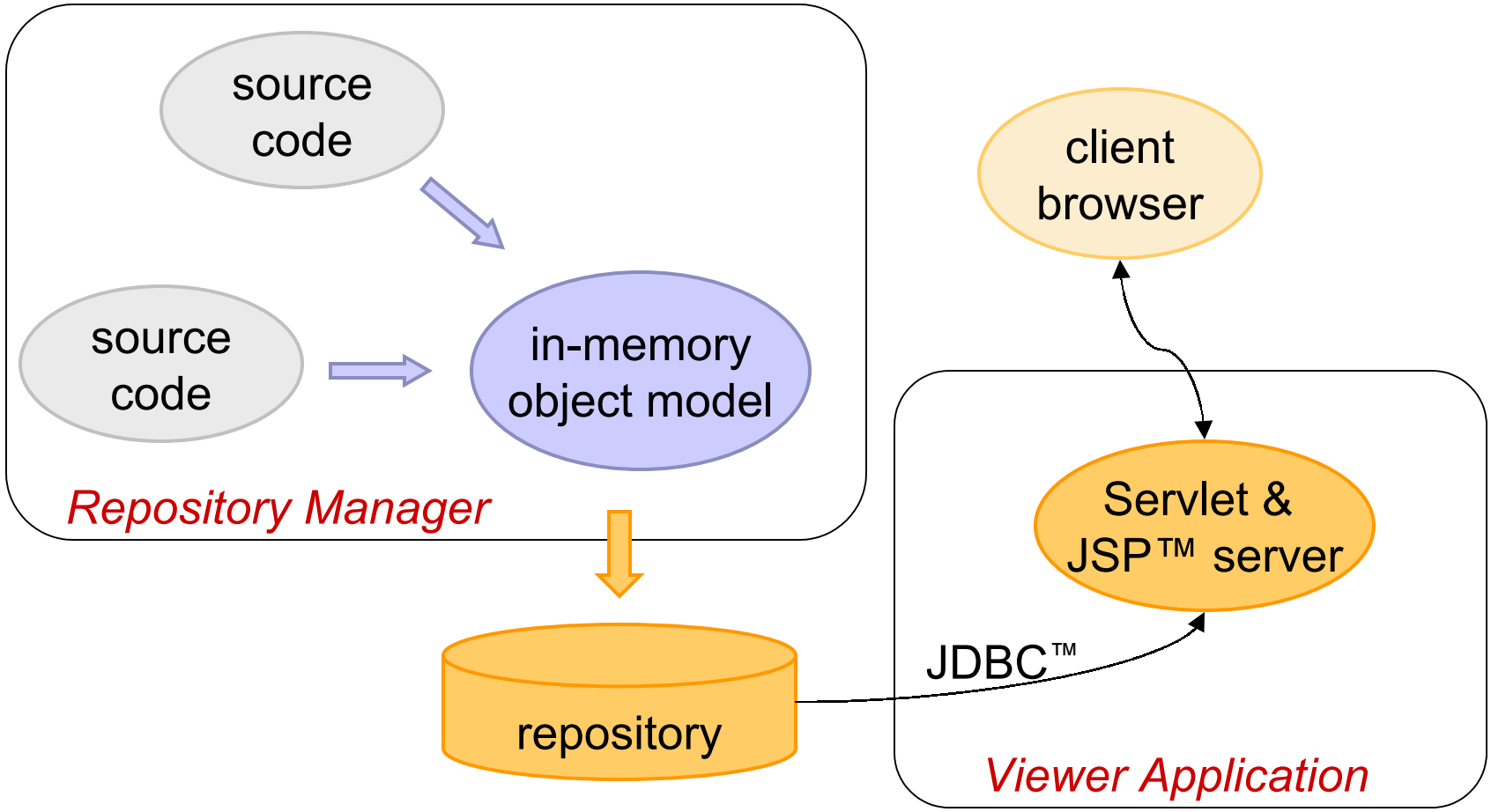
1. Architecture
2. Repository Mgr Design & Implementation
3. Viewer Application
   - Back-End Design & Implementation
   - Front-End Design & Implementation
4. Demo
5. Some Useful Statistics
6. Summary and Q&A

Javadoc™ Architecture

Documentation Manufacturing Process



Revised Architecture



Advantages of a Repository

- Can be queried, searched
- Provides single location for multiple APIs
- Simplifies management, control of documentation
- Makes incremental population possible
- Cross-API reference aware

Repository Manager Overview

- Command line interface similar to Javadoc technology
- Not strictly a doclet
- Commands: add, list, remove, reset
- Invoke javadoc programmatically with doclet parameter as self
- Allows incremental population of APIs

Repository Manager Implementation

- Implements *Doclet* interface
- Accesses object model via RootDoc interface
  - Iterate over packages
  - Drill down to classes, interfaces, and members
  - Extract and persist API information and documentation to database using the Java DataBase Connectivity (JDBC™) API

Repository Manager Sample Code

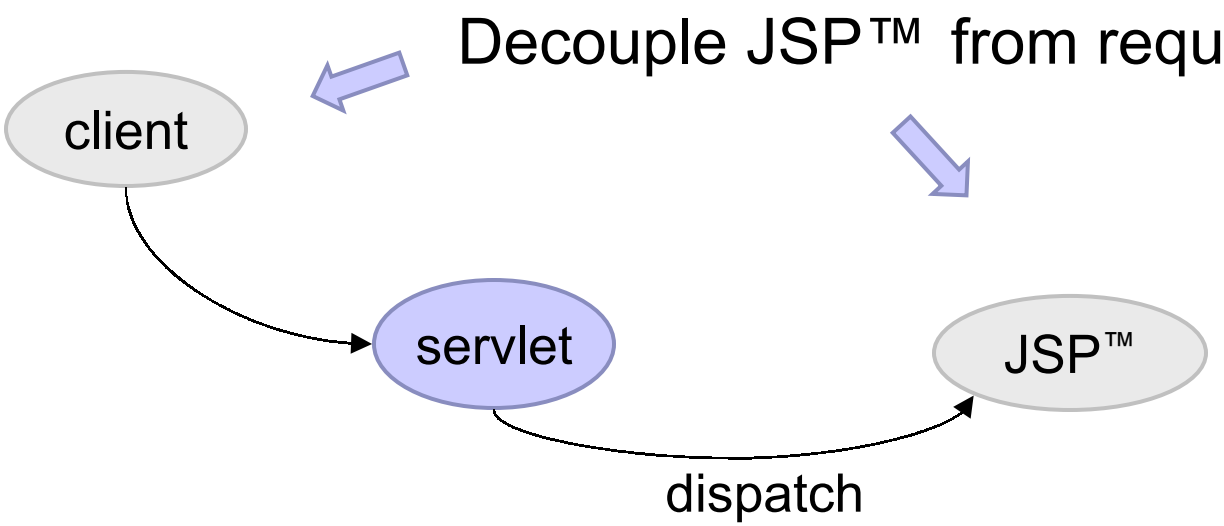
```
public static boolean start(RootDoc root) {
 ..
 PackageDoc[] packages = root.specifiedPackages();
 for (int i=0; i<packages.length; i++)
 {
   try {
     new JPackage(packages[i], true).store(conn);
     conn.commit();
   } catch (SQLException ex)
   { .. }
 }
 ..
}
```

Underlying Object Model Sample Class

```
public class ClassType {
  private String name;
  private boolean isAbstract;
  ..
  public ClassType(ClassDoc classdoc) {
    setName(classdoc.name());
    ..
  }
  ..
  public void store(Connection conn)
    throws SQLException {  .. }
  ..
  public static ClassType makeClassFor(int clsId)
    throws SQLException { ..
    String sql = dbMgr.getStatement("makeclass");
    ..
  } ..
}
```

Viewer Application Back-End Architecture

Struts-like architecture



JSP components do not entirely displace servlets

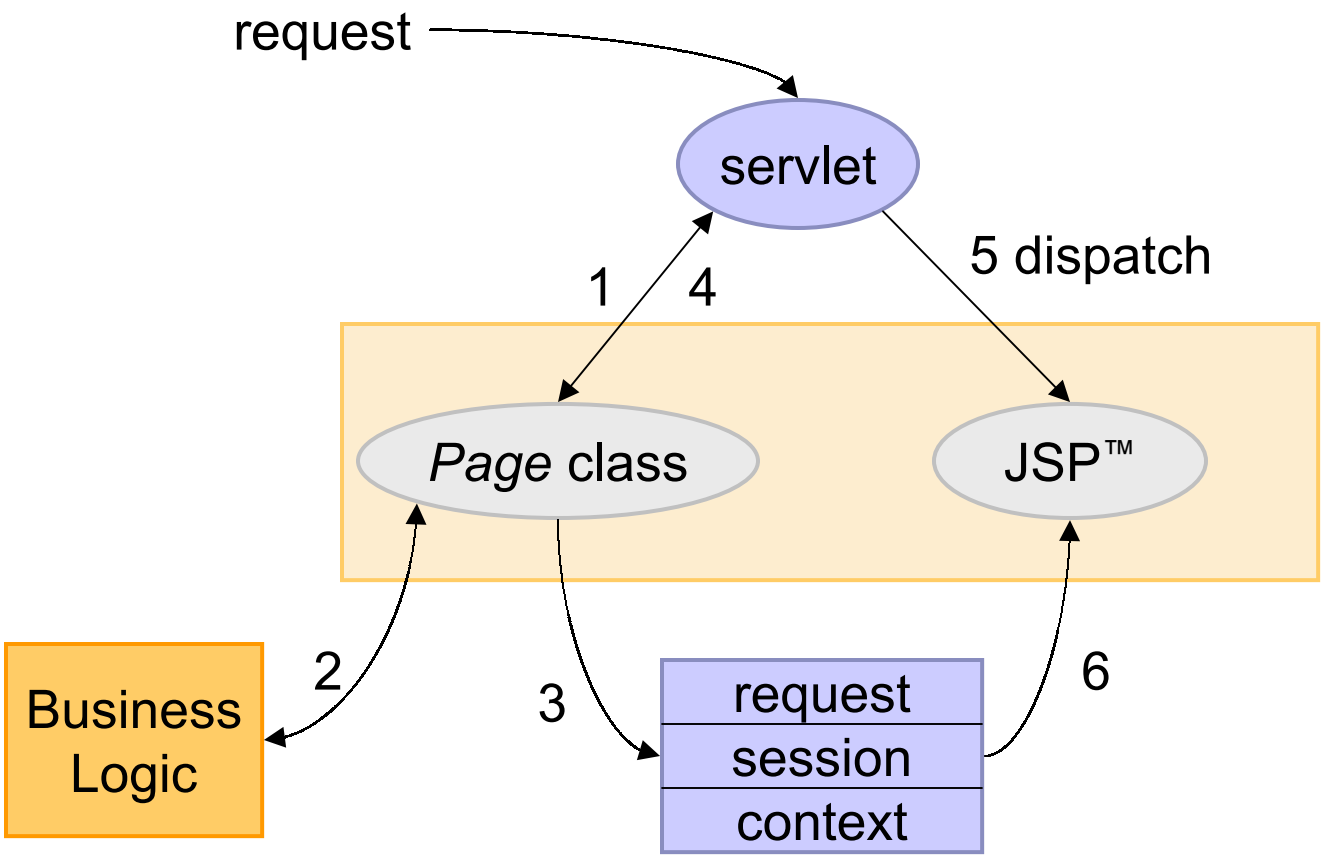
Role of the JavaServer Pages™ (JSP™) Specification

Use the JSP™ spec strictly as a template engine

- Separate request handling from response rendering:
  1. Specialized Java class handles page requests
  2. JSP spec handles rendering of response
- Communicate via servlet request, session, and context objects

Request Handling

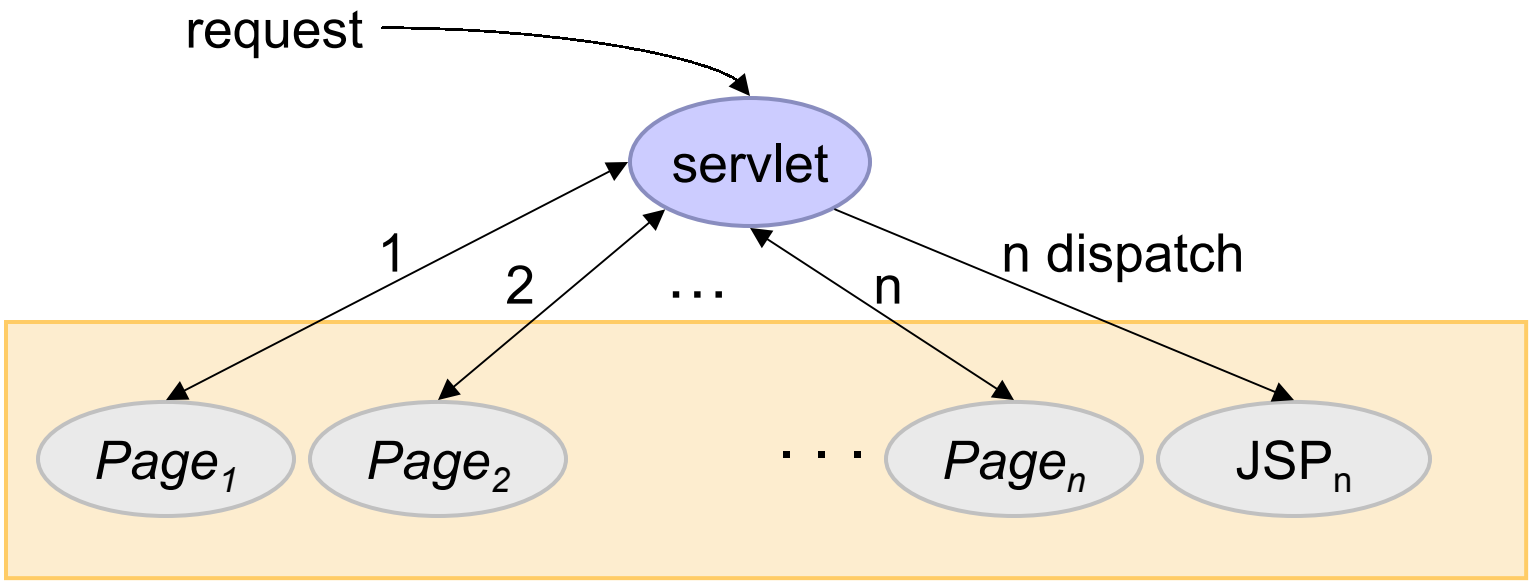
Combine the two..



Request Handling Enhanced

Take it one step further

- Allow Page Classes to request a server-side “redirect”



Request Handling Sample Code

Communication mechanism between flow controller servlet and Page classes

```
public abstract class Page
{
   ...
   /**
    * @return the name of a valid command
    *  which the flowcontroller will use
    *  to forward the request to instead of the
    *  default associated page name, or null.
    */
   public abstract String init()
     throws PageException;
}
```

Roles of Servlet

- Principal role: flow controller
- Handle logic common to all requests
- UI Abstraction Layer
  - serve different sets of JSP components to different clients

FlowController Servlet Sample Code

```
public class FlowController extends HttpServlet {
  ..
  public void doPost(.. request, .. response) ..
  {
    String cmd = request.getParameter("cmd");
    cmdInfo = (CommandInfo)
       configInfo.getCommandMap().get(cmd);
    pageName = cmdInfo.getPageName();
    className = cmdInfo.getClassName();
    Page page = (Page)
      Class.forName(className).newInstance();
    cmd = page.init();
    RequestDispatcher rd =
       context.getRequestDispatcher(pageName);
    rd.forward(request, response);
  } ..
}
```

Database Abstraction Layer

- Vendors have proprietary extensions to SQL
- Syntax varies for joins and other operations
- Database layer abstracted by:
  - Database type as configuration parameter
  - Statements for supported databases in resource file
  - Analogous to use of resource files for internationalization

DB Abstraction Layer Sample Resource File

```
..
database.type={mysql|oraclei|..}
#
oraclei.getclass=select * from classtype where
id=? and rownum<=20
#
mysql.getclass=select * from classtype where id=?
limit 20
#
..
```

Viewer Application Front-End Why HTML?

- Notion of hyperlinks is *essential*
- Not just “another format”
- A standard for modern GUIs
- Proven to work for documentation systems

The Next Step: DHTML

- Continue in direction of what works
- Can construct richer graphic interface
- Rapid development
- Provide increased control over user interface:
  - Ability to hide, display information on a page
  - Collapsible trees, tabbed panes, tool tips..

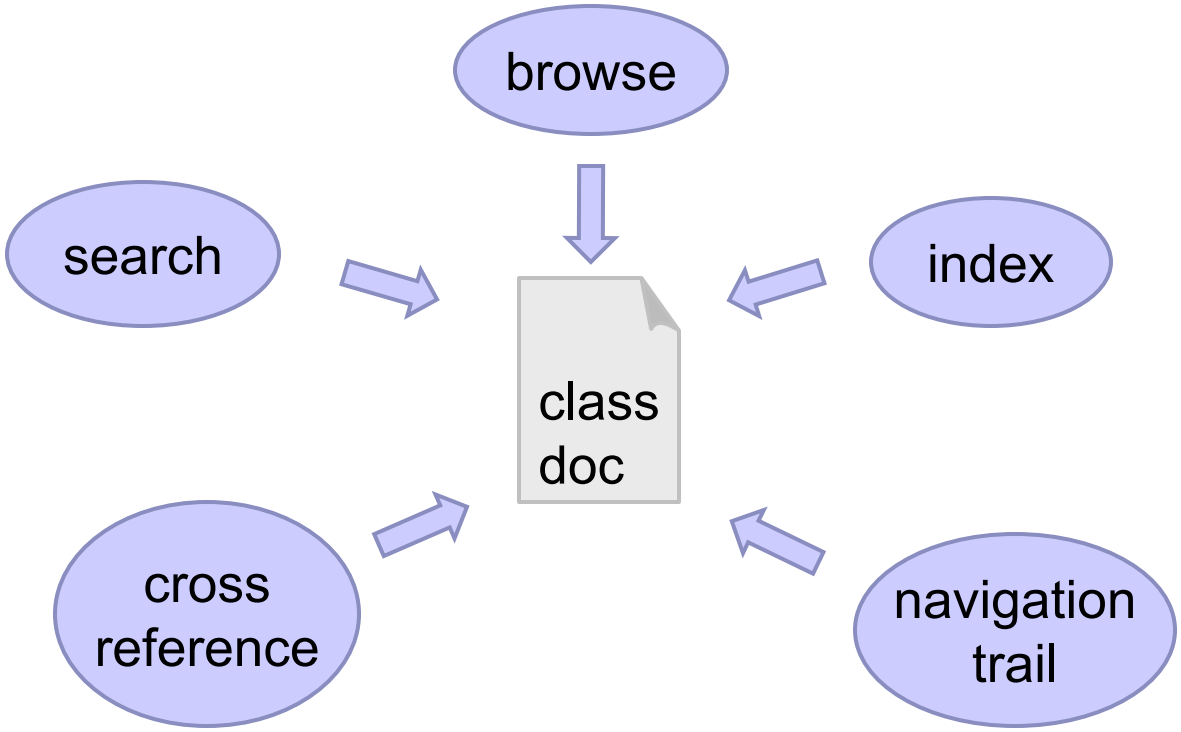
Challenge: Managing Information

By using effective GUI Techniques

- Color-coded programming elements
- Try to minimize scrolling requirements
- GUI Components serve as aids
  - collapsible trees, filtering information on page, tab sets, tool tips
- Effective use of hyperlinks

Managing Information (continued)

Provide multiple access points



Front-End Design Overview

- Can handle multiple alternate clients
- Two clients: DHTML and Classic
  - Only JSP components differ between clients
- Classic client constructed in three days

Front-End Implementation

- DHTML Client:
  - Uses CSS to separate style from content
  - Single style sheet controls programming element styles for entire application
- Classic Client:
  - HTML 3.2, no javascript client
  - Look and feel similar to Javadoc technology

Sample Style Sheet

```
..
.ordinaryClass {
 color: #0000ff;
}
.exception {
 color: #ff8080;
}
.interface {
 color: #008000;
 font-style: italic;
}
.deprecated {
 text-decoration: line-through;
}
..
```

```
<SPAN CLASS="<%=attr%>">
  <%=cls_name%>
</SPAN>

may resolve to:

<SPAN CLASS="deprecated">
  StringBufferInputStream
</SPAN>
```

"JSP Components" Revisited (JSP™ specification-based components)

- <jsp:include> tag promotes componentization, can be parameterized
- Achieves high degree of re-factoring:
  - 72 client objects (HTML, JS, JSP, CSS..)
  - 60 are JSP components
  - Only 26 are top-level pages
- Template mechanism makes for a manageable GUI (remember the days before JSP technology?)

Sample JSP™ Technology-based Template ("JSP Template")

```
<%@ page info="class view" import=".." %>
<%
  ClassType cls =
   (ClassType) request.getAttribute("cls");
  ..
%>
<HTML> .. <BODY>
 <jsp:include page="header.jsp" flush="true"/>

 <DIV CLASS="PAGEBODY">
   <jsp:include page="clsinfo.jsp" flush="true">
    <jsp:param name="clstype" value="<%=type%>"/>
   </jsp:include>
 </DIV>
..
</BODY> </HTML>
```

Use Cases

- Locate a class directly by name
- Customize or narrow search criteria
- Utilize cross references as a discovery tool
- Illustrate role of navigation trail
- Access to Source
- Provide multiple clients

Demo

Trivia anyone?

Question

How many methods comprise the Java 2 Platform, Standard Edition, version 1.4?

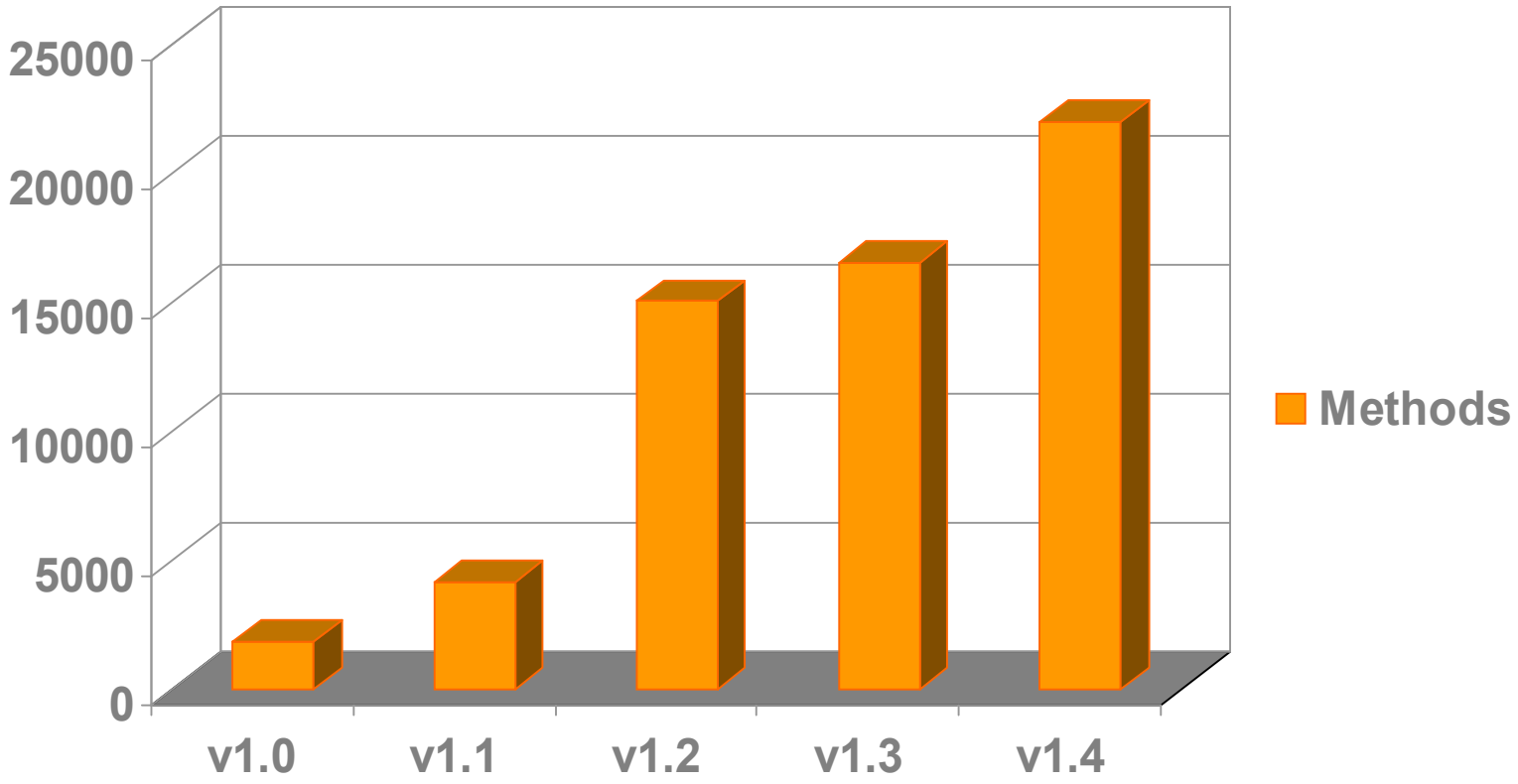
Java™ Technology has grown

Answer

There are over 21,981* methods in J2SE™ v1.4 release

*v1.4 information compiled using beta 3; count does not include javax.crypto, javax.net, and a few other packages (13 in all)

The J2SE™ Platform: Statistics



Summary

- Wishes do come true ☺
- The Java™ platform offers powerful tools for 3-Tier web application design
- Manageability of solution of paramount importance
- Good design decisions help simplify future challenges

If You Only Remember One Thing…

Tools and technologies
are only as good as
their systems of documentation

Q&A

My Questions to You

- Did I overlook items on your wish list?
- Comments, feedback for improving solution?
- What tools do you use for Java™ API documentation?
- Do you find a tool such as described in this presentation useful?
  - You may answer me at eitan@uptodata.com

JavaOne℠

Sun's 2002 Worldwide Java Developer Conference™

BEYOND BOUNDARIES

Session # 1145