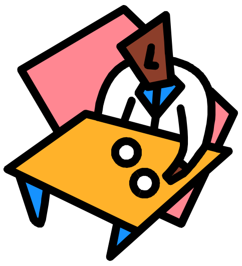
Developing Dynamic Web Sites with JavaServer Pages

John Zukowski

jGuru.com

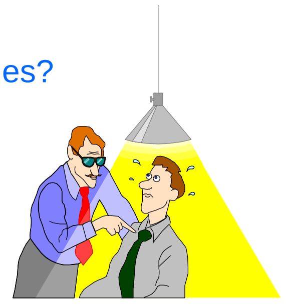
Agenda

- Introduction
- Architecture
- Syntax
- Usage
- Q&A



Who is in Attendance?

- Background Check
  - System/Web Administrator?
  - Page Designers?
  - Programmer? VB/C/C++? Perl? ASP?
  - Java 1.0/1.1/1.2/1.3/2?
  - Read Java Books/Magazine Articles?



JavaServer Pages

Understanding Dynamic Content

- A typical web page consists of static (template) and dynamic (personalized) content
- Static elements include images, navigational elements, descriptive text, etc.
- Dynamic content varies based on user
- Page layout is usually handled via a markup language like HTML or DHTML

Servlets Are Great, But...

- Servlet technology is more suited for developers, not designers
- Servlets cannot be developed using HTML editors
- Minor changes in HTML UI needs recompilation of servlet
- Tight coupling of presentation and content leads to brittle, inflexible applications

Server-side Scripting

- Attempts to separate presentation from content
- Composite page consists of static presentation templates with ‘scripts’ and special tags responsible for extracting and inserting dynamic content
- Composite page is fully processed on the server before a response is sent to the client
- Popular server-side scripting technologies include JavaServer Pages (JSP) and Active Server Pages (ASP)

JSP vs. ASP

- Available for multiple platforms
- JavaBeans component model
- Supports Java for scripting
- Works with the Java security model

- Fully supported only on Win32
- COM-based component model
- Supports VBScript and JScript scripting
- Uses Windows NT security

Understanding JSP

- Integral part of Java 2 Enterprise Edition
- Write Once Run Anywhere
- JSP is a standard extension defined on top of the servlet extension
- Recommended web access layer for N-tier architecture
- Enables a clean partition between static and dynamic content

Understanding JSP

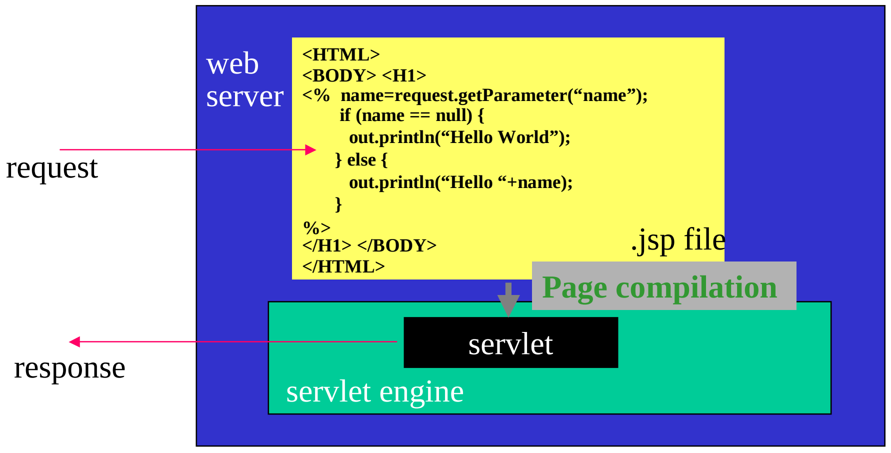
- Emphasizes reusable components like JavaBeans, EJB, and custom tags
- Dynamic content generated by JSP can be HTML, DHTML, XHTML, XML, etc.
- JSP (*.jsp) files include snippets of Java code within static HTML files
- JSP 1.1 and Servlet 2.2 API can be used with JDK 1.1 or Java 2

Anatomy of a JSP Page

```
<html>
<head> <title>Order Information </title></head>
<body>
<%@ page import="com.foo.*" buffer="16k" %>
<jsp:useBean id=cust class=Customer scope=session>
 <jsp:setProperty name=cust property=itemNumber param="item">
</jsp:useBean>
<% if (cust.powerShopper()) {
       out.println("Shipping is free! Thanks for your order.");
     }  else {
       out.println("Thank you. We appreciate your business.");
    }
%>
</body>
</html>
```

▮	template
▮	directive
▮	action
▮	scripting element

JavaServer Pages Visual

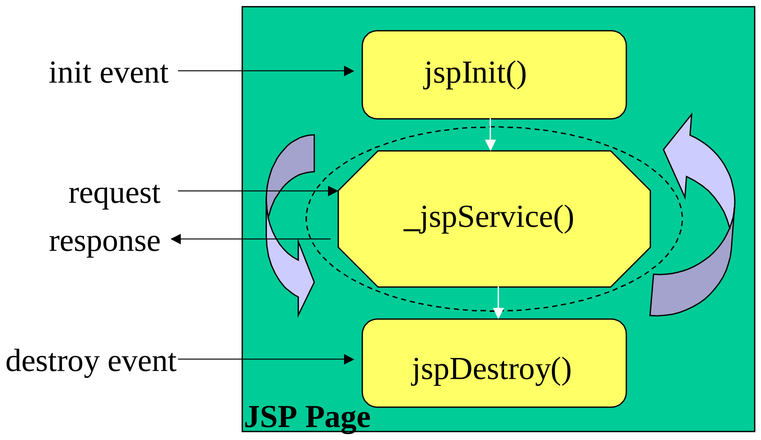


```
<HTML>
<BODY> <H1>
<%  name=request.getParameter("name");
      if (name == null) {
        out.println("Hello World");
     } else {
        out.println("Hello "+name);
     }
%>
</H1> </BODY>
</HTML>
```

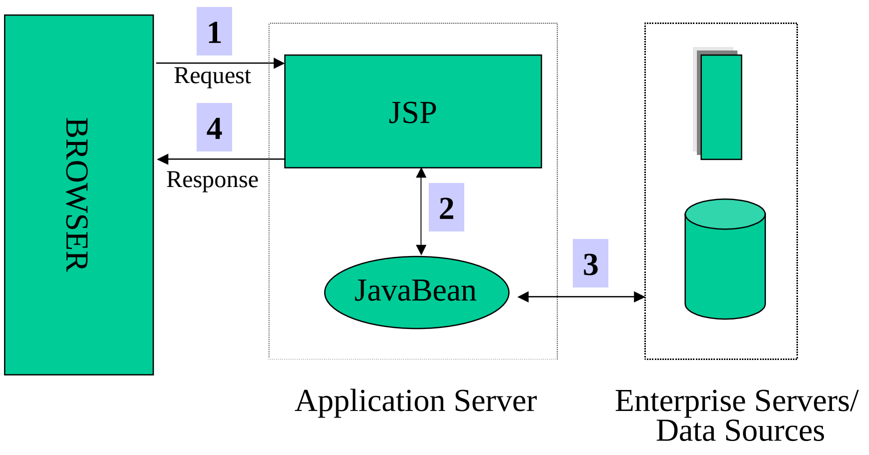
Page Compilation

- JSP engine creates an implementation class file for each page
- Page implementation class implements the HttpJspPage interface if protocol is HTTP
- Many details of the translation phase are implementation dependent
- Fatal translation error results in client receiving “Status code 500”

Request Processing Phase

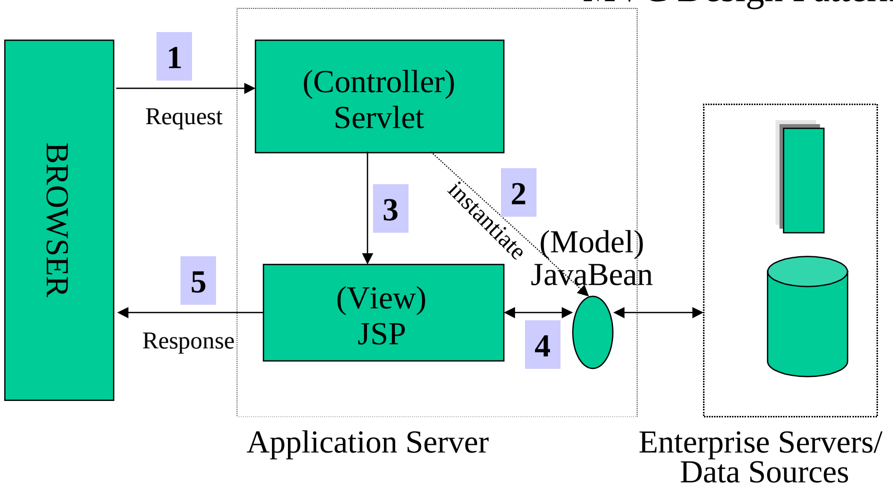


Basic JSP Architecture - Model 1

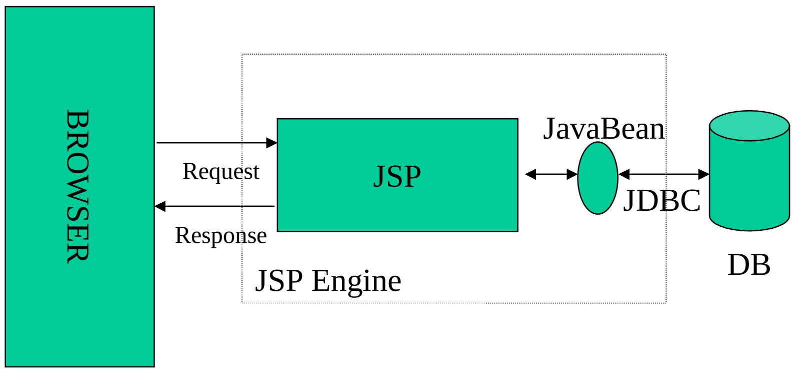


Basic JSP Architecture - Model 2

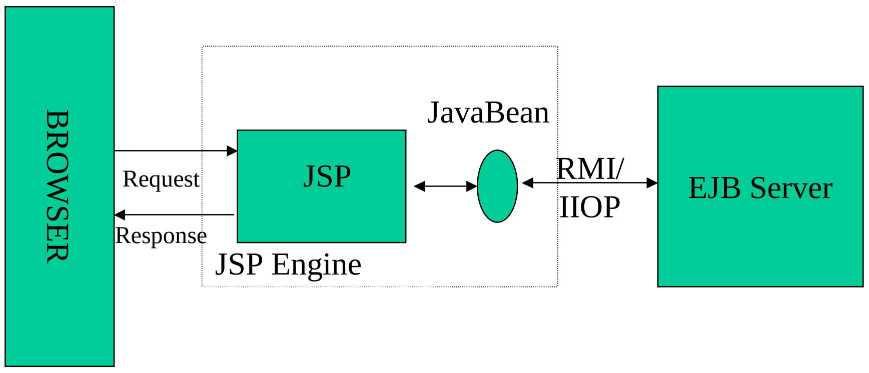
MVC Design Pattern



2-tier JSP Architecture



N-tier JSP Architecture



Implicit JSP Objects

- **`request`** - represents HttpServletRequest triggering service invocation
  - Request scope
- **`response`** - represents HttpServletResponse to request
  - Not used often by page authors
  - Page scope
- **`pagecontext`** - encapsulates implementation-dependent features as PageContext
  - Page scope

Implicit JSP Objects/2

- **`application`** - represents the ServletContext obtained from servlet configuration object
  - Application scope
- **`out`** - represents a JspWriter object that writes into the output stream
  - Page scope.
- **`config`** - represents the ServletConfig for the JSP
  - Page scope

Implicit JSP Objects/3

- **`page`** - synonym for the “this” operator, HttpJspPage
  - Not used often by page authors
  - Page scope
- **`exception`** - the uncaught Throwable object that resulted in the error page being invoked
  - Page scope

JSP Object Scopes

- **`page`** - reference is discarded upon completion of the current request by the page body
- **`request`** - reference is released upon completing the client request. Named object can be obtained from the ServletRequest using getAttribute(name).

JSP Object Scopes/2

- `session` - references are stored in the session object and are released when session is invalidated
- `application` - references are released when runtime reclaims SessionContext
  - Pages need not be session aware

JSP Standard Directives

- Directives are messages to the JSP engine
- Directive have scope of the entire JSP file
- They do not produce any output into the current out stream
- Directive Syntax

```
<%@ directive {attr="value"}* %>
```

`include` Directive

- Useful for including static resources
- Inclusion performed during translation phase
- Syntax

```
<%@ include file="relativeURL" %>
```

- Example

```
<%@ include file="header.html" %>
```

page Directive

- A translation unit can contain any number of page directives
- The attribute/value pair must be unique for each translation unit
- Unrecognized attributes or values result in a translation error

page Directive Syntax

```
<%@ page page_directive_attr_list %>
page_directive_attr_list ::=
                { language    = "scriptingLanguage"}
                { extends     = "className"}
                { import      = "importList"}
                { session     = "true|false"}
                { buffer      = "none| sizekb"}
                { autoFlush   = "true| false"}
                { isThreadSafe= "true|false"}
                { info        = "info_text"}
                { errorPage   = "error_url"}
                { isErrorPage = "true|false"}
                { contentType = "ctinfo"}
```

- Examples

```
<%@ page info="hello world jsp example" %>
<%@ page import="com.foo.*" buffer="16k" %>
```

JSP Exception Handling

- For translation errors, the browser is returned status code 500 indicating server error
- Uncaught exceptions during request processing may be automatically forwarded to an errorPage URL
- Throwable object describing the exception may be accessed within the error page via the exception implicit object
- Example

```
<%@ page isErrorPage="false" errorPage="/errors.jsp" %>
```

Synchronization Issues

- JSP authors must ensure synchronized access to shared page state
- To have page implement SingleThreadModel set directive:

```
<%@ page isThreadSafe="false" %>
```

- Access to shared objects within HttpSession or ServletContext must always be synchronized

Scripting Elements: Comments

- JSP-style comments document what the page is doing

```
<%-- this does not appear at the client --%>
```

- You can also use comment mechanism of the scripting language

```
<% /** this is a comment **/ %>
```

- Comments can also be made to appear within generated content sent to client

```
<!-- this comment is visible at the client -->
```

Scripting Elements: Declarations

- Used for defining variables and methods
- Are initialized during the translation phase
- Syntax

```
<%! declaration(s) %>
```

- Examples

```
<%! int foo=10, bar=20; %>
<%! public void jspInit() {
    . . .
  }
%>
<%! public void jspDestroy() {
    . . .
  }
%>
```

Scripting Elements: Scriptlets

- Scriptlets contain valid code fragments that are executed during the request phase
  - Can modify any visible object
  - May send output to the `out` stream
- Syntax

```
<% scriptlet %>
```

Scriptlet Example

```
<%
  Date d = new Date();
  DateFormat df = DateFormat.getDateInstance();
  out.println("Today is " +df.format(d));
%>
```

Scripting Elements: Expressions

- An expression is evaluated, the result converted to a String and then sent to the response stream
  - Expressions are evaluated during the translation phase
- Syntax

```
 <%= this is an expression %>
```

- Example

```
<%= java.text.DateFormat.getDateInstance().format(
                           new java.util.Date())
%>
```

Standard Actions

- Actions usually depend on the details of the specific request object received by JSP page
- May affect the current out stream
- May read, create, or modify visible objects
- Action syntax is based on XML
- Most attributes for JSP 1.1 actions have translation time semantics

JSP and JavaBeans

- A JSP page can access a JavaBean component using the `<jsp:useBean>` action
- Syntax

```
<jsp:useBean id="name"
  scope="page|request|session|application" typeSpec
  />
```

```
– typeSpec ::= class="className" | class="className"
  type="typeName" | beanName="beanName"
  type="typeName" | type="typeName"
  beanName="beanName" | type="typeName"
```

JSP and JavaBeans Body

- If the action has a body, it is of the form:

```
<jsp:useBean id="name"
  scope="page|request|session|application" typeSpec>
body
</jsp:useBean>
```

- The body is invoked after bean created
- Body usually contains scriptlets or `<jsp:setProperty>` tags to initialize newly created bean

JSP and JavaBeans Examples

- A reference named “connection” to a bean of type “com.foo.db.Connection” is obtained. If the bean was not previously created, it is newly instantiated

```
<jsp:useBean id="connection"
  class="com.foo.db.Connection" />
```

- If bean instantiated, Timeout property is set to 1800

```
<jsp:useBean id="connection"
  class="com.foo.db.Connection" >
<% connection.setTimeout(1800); %>
</jsp:useBean>
```

Setting Properties

- The <jsp:setProperty> action sets the values of properties in a bean.
- Syntax

```
<jsp:setProperty name=" beanName" prop_expr />
– prop_expr ::= property="*" |
  property="propertyName" |
  property="propertyName" param="parameterName" |
  property="propertyName" value="propertyValue"
– propertyValue ::= string
– propertyValue ::= expr_scriptlet
```

Setting Properties/2

- Examples

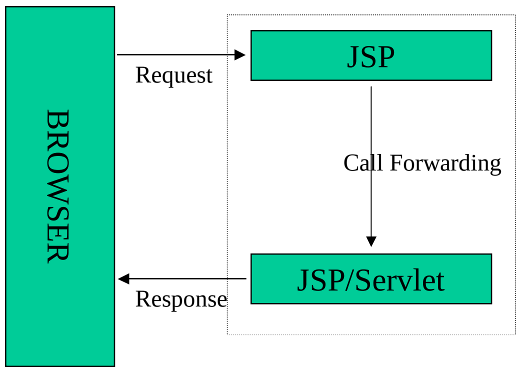
```
<jsp:setProperty name="order" property="*" />
<jsp:setProperty name="user" property="user"
  param="username" />
<jsp:setProperty name="res" property="row"
  value="<%= i+1 %>" />
```

Getting Properties

- The `<jsp:getProperty>` action gets the values of properties in a bean
- Places the value of the bean instance property into the implicit out object
- Bean must been previously defined
- Syntax

```
<jsp:getProperty name="name"
  property="propertyName" />
```

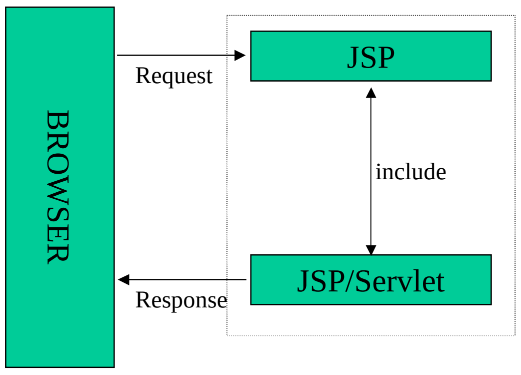
Forwarding Requests



Forwarding Requests

- Requests can be redirected based on client properties or user profile info to a static page, JSP, or servlet
- If page is buffered, then buffer is cleared prior to forwarding
- Syntax
```
<jsp:forward page="relativeURL" />
```
- Example
```
<% String someURL="/jsp/resource.html" %>
<jsp:forward page='<%=someURL %>' />
```

Including Requests



Including Requests

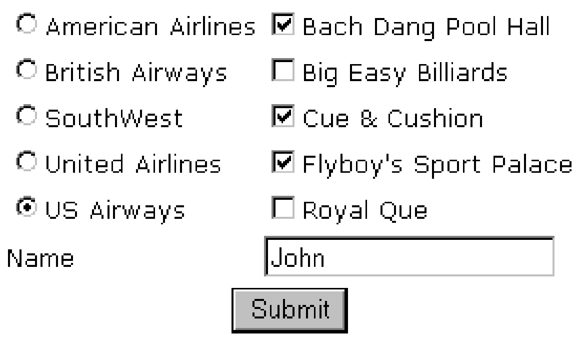
- Included contents may be static or be dynamically generated by servlet or JSP
- Processed during the request handling phase
- Included resource cannot set headers
- Syntax

```
<jsp:include page="relativeURL" flush="true"/>
```

- Example

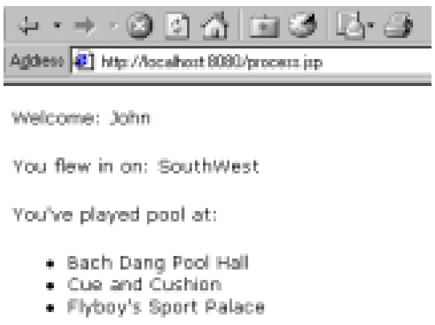
```
<jsp:include page="/examples/jsp/copyright.jsp"
  flush="true"/>
```

JSP Complete Example



American Airlines	[✓ Bach Dang Pool Hall]
British Airways	Big Easy Billiards
SouthWest	[✓ Cue & Cushion]
United Airlines	[✓ Flyboy's Sport Palace]
[✓ US Airways]	Royal Que
Name	John

JSP Example Results



Welcome: John

You flew in on: SouthWest

You've played pool at:

- Bach Dang Pool Hall
- Cue and Cushion
- Flyboy's Sport Palace

JSP Example Source

```
<html>
<head><title>JSP Example</title></head>
<body>
<p>Welcome:
<%= request.getParameter("name") %>
</p>
<p>You flew in on:
<%= request.getParameter("airline") %>
</p>
<p>You've played pool at:</p>
<%! String poolhall[]; %>
<%
 poolhall=request.getParameterValues("poolhall");
    if (poolhall != null) {
%>
```

JSP Example Source/2

```
<ul>
<%
      for (int i=0, n=poolhall.length; i<n; i++) {
%>
<li><%= poolhall[i] %>
<%
      }
%>
</ul>
<%
    } else {
%>
<p><em>Nowhere</em></p>
<%
    }
%>
</p></body></html>
```

Installing Tomcat with Apache

- Key things:
  - Add to <ApacheInstallDir>/Apache Group/Apache/conf/httpd.conf
    - Include "<TomcatInstallDir>/conf/tomcat.conf"
  - Uncomment lines in <TomcatInstallDir>/conf/tomcat.conf
    - LoadModule jserv_module modules/ApacheModuleJServ.dll
    - ApJServDefaultHost localhost
  - Be sure you have ApacheModuleJServ.dll
- See http://www.jguru.com/jguru/faq/view.jsp?EID=56994

JSP Summary

- JSP technology is an excellent cross-platform method of generating dynamic content
- JSPs effectively separate presentation from content by emphasizing reusable components
- Expect to see more JSP-aware WYSIWYG editors in the near future (HomeSite now)

Resources

- Servlets FAQ
  - http://www.jguru.com/faq/Servlets
- Servlets Home
  - http://java.sun.com/products/servlet/
- JSP FAQ
  - http://www.jguru.com/faq/JSP
- JSP Home
  - http://java.sun.com/products/jsp
- Tomcat (Reference Implementation)
  - http://jakarta.apache.org

More Resources

- JDC JSP Tutorial
  - http://developer.java.sun.com/developer/onlineTraining/
- JSP-Interest Mailing List
  - http://archives.java.sun.com/archives/jsp-interest.html
- JRun
  - http://www.allaire.com/products/jrun/

JSP Books

- October JavaWorld Issue has comprehensive review of JSP books
  - Best: **Web Development with JavaServer Pages** (Manning)
  - Good but short: **Pure JSP** (Sams)
  - Servlets and JSP mix: **Core Servlets and JavaServer Pages** (Prentice Hall)
  - Honorable Mention: **Professional JSP** (Wrox)
    - I'm one of 21 authors of book.
      [Too many voices/styles in book]

Questions

Contact Information

- John - jaz@jguru.com
  - http://www.jguru.com
  - http://java.about.com