Web Development using Java, JSP, and Web Services

JavaServer Pages

Lecture #4 2008

Server-Side Web Development

- Data access and logic rather than interfaces
- More technical than HTML/CSS design
- Several alternative technology platforms exists
- Dynamic content vs static content
- Content-driven solutions
- Focus on web applications
- Need for a structured programming model
- Need for a simpler way to do programming

Web Applications

- Applications with a web interface
- Ideal for thin-client solutions
- Suffers from limitations of the web media
- Usually combines techniques (e.g., JSP + AJAX)
- Several development frameworks available
- Clear trend towards generated web interfaces
- Usually session-oriented
- Usually deployed in WAR files

Web Archives (WAR)

- ZIP-file = compressed archive
- JAR-file = Java Archive (ZIP file with a manifest)
- WAR-file = JAR-file with web application information

Sessions

- Used to store data for a series of HTTP requests
- E.g., a shopping cart, user preferences, site history
- A session identifier is sent with each request
- The session identifier is used to locate the session
- Data is stored in the session context

Three Layer Architectures

1. Interface - web pages
2. Logic - software components (JavaBeans, EJB)
3. Data - databases

- Clean separation of concerns
- Scalable
- Support role-based development cycles
- Well suited for large sites and business logic integration

Two Layer Architectures

1. Interface (+ Logic)
2. Data + Logic

- Requires fewer software components
- Shorter development cycles for small development teams
- Better suited for smaller web sites

Web Servers

- Serves resources via HTTP
- Can be anything that serves data via HTTP
- Usually a dedicated machine running web server software
- Can contain modules that processes requests

Request Processing

1. A Connection is established
2. A HTTP request is received
3. The (logical) path in the request is translated
4. The requested resource is identified (via the path)
5. A server module handling that resource is invoked
6. The module processes the request and generates a reply
7. A mime-type is provided and a HTTP response is created
8. The HTTP response is sent (possibly in increments)

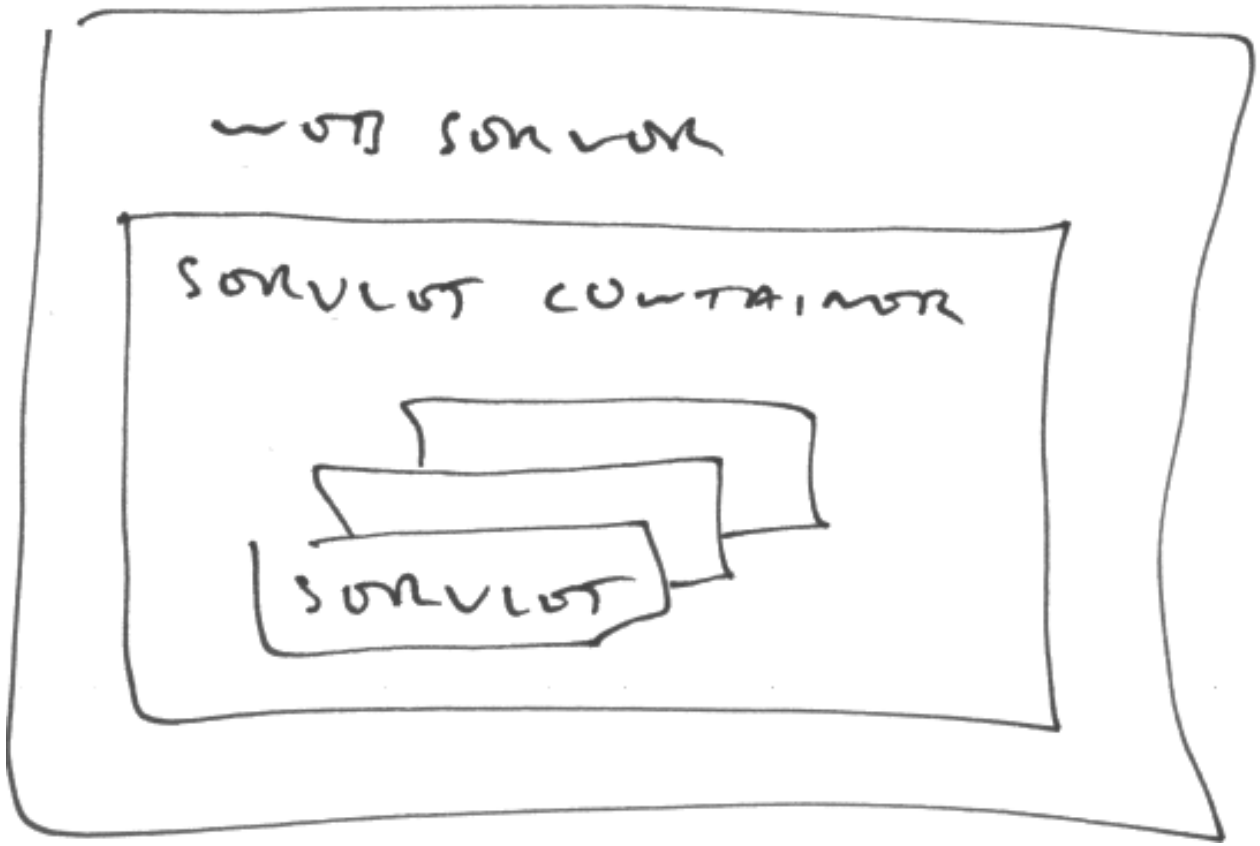
JSP Processing

1. A JSP page is requested
2. Server checks if a Java Servlet for the page exists
3. If no Servlet is found (or newer JSP is detected), the JSP is translated to Java (a Servlet class is created)
4. The Java Servlet is compiled
5. The Java Servlet is invoked and processes the request

Java Servlets

- Java classes
- Implements the Java Servlet API interfaces (predates JSP)
- Receives a request and generates a response
- Can be written manually
- Must be thread-safe
- Usually generated automatically from JSP
- Hosted in a Servlet container

Java Servlets



The Servlet Lifecycle

1. init() - called on Servlet instantiation
2. service() - called for each request
3. destroy() - called on container shutdown

The Servlet service() method

- Part of a service-pattern
- Should not be implemented directly
- Inherit base class and implement handler methods
- Distinct handlers for each HTTP method (e.g., doGet())

1. service() parses request and determines HTTP method
2. service() calls appropriate handler method
3. Handler method processes request

JavaServer Pages (JSP)

- HTML with extra XML-tags
- (Scripted server-side) Java for the web
- A way to provide dynamic content in web pages
- XML-tags act as front-ends for Java classes
- May include other pages dynamically
- JSPs are compiled into Java Servlets
- JSP code is never visible to clients
- Generates response (HTML) dynamically
- Model-View-Controller pattern recommended

JSP Syntax

JSP may contain

- Static HTML
- Directives
- Scripting Elements
- Actions
- Comments
- Tag Libraries
- Implicit Objects

JSP Script Tags

- Directives

  `<%@ ... %>`

- Declarations

  `<%! ... %>`

- Scriptlets

  `<% ... %>`

- Expressions

  `<%= ... %>`

- Comments

  `<%-- ... --%>`

Static HTML

- All HTML is treated by JSP as static text
- HTML may be mixed with JSP in any way
- All non-JSP tags are treated as HTML

Directives

```
<%@ ... %>
```

Types of JSP directives

- Page
- Include
- Tag Libraries (aka Custom Tags)

Page Directives

```
<%@ page attribute="..." %>
```

- Instructs the JSP engine how to process the JSP
- Attributes determine directive content

Page Directive Attributes

- language - selects scripting language (Java)
- extends - base class for generated Servlet
- **import** - Java class / package import
- **session** - enable session tracking (default: true)
- buffer - set output buffer size
- autoFlush - enable auto flushing of output buffer
- isThreadSafe - thread safe marker
- info - page information (author, version, copyright etc)
- **errorPage** - set default error page
- **isErrorPage** - enable exception tracking on page
- **contentType** - set response mime type

Include Directive

```
<%@ include file="page.jsp" %>
```

- Includes another page at translation time
- Included page becomes part of Servlet
- Generates error if page not found

Tag Libraries

```
<%@ taglib uri="taglib.tld" prefix="prefix" %>
```

- Loads a tag library
- Tags are usable via the specified tag prefix
- The JSP version of language extension

Directive Examples

```
<%@ page import="examples.*, examples.tags.*" %>
<%@ taglib uri="/WEB-INF/examples-taglib.tld"
           prefix="examples" %>

<%@ include file="/includes/head.jsp" %>

...

<examples:ValidateParameters
           parameters="name,age"/>

...

<%@ include file="/includes/foot.jsp" %>
```

Scripting Elements

- Declarations

  <%! ... %>

- Scriptlets

  <% ... %>

- Expressions

  <%= ... %>

Declarations

- Content placed in Servlet body (members)
- Used to declare members and methods
- Does (usually) not produce HTML output
- Declared content is later used by Scriptlets or expressions
- Lines must be terminated with a semicolon

Declaration examples

```
<%!
  public int getSum (int x, int y)
  {
    return x + y;
  }
%>
```

```
<jsp:declaration>
  public int getSum (int x, int y)
  {
    return x + y;
  }
</jsp:declaration>
```

Scriptlets

- Content placed in Servlet _jspService() method (local variables and in-line code)
- Used to embed Java code directly in the page
- May produce HTML output (via out.println())
- Lines must be terminated with a semicolon

Scriptlet examples

```
<%
  int x = 1;
  int y = 2;
  int sum = x + y;
%>
```

```
<jsp:scriptlet>
  int x = 1;
  int y = 2;
  int sum = x + y;
</jsp:scriptlet>
```

Expressions

- Content output directly to HTML
- Used as an alias for out.println()
- Code must evaluate to an expression
- Lines must **not** be terminated with a semicolon

Expression examples

```
1 + 2 = <%= 1 + 2 %>
```

```
1 + 2 =
<jsp:expression>
  1 + 2
</jsp:expression>
```

Actions

- **include** - include another page
- **forward** - forward request to another resource
- **param** - specify parameters when calling or forwarding

- plug-in - generates browser-specific code for applets
- fallback - content if browser does not support applets

- getProperty - get a property from a JavaBean
- setProperty - set a property on a JavaBean
- useBean - use a JavaBean

Include Action

```
<jsp:include page="page.jsp"/>
```

- Includes another page at request time
- Generates a request to included page Servlet
- Ignored if page not found
- Usually used to call declared methods
- Control is returned

Forward Action

```
<jsp:forward page="page.jsp"/>
```

- Forwards request to another resource
- Control is not returned

Param Action

```
<jsp:forward page="page.jsp">
  <jsp:param name="name" value="value"/>
</jsp:forward>
```

- Used to specify parameters when including / forwarding

HTML Comments

```
<!-- HTML comment -->
```

- Evaluated by server (as HTML)
- Part of the server response / web page
- Visible in page source from web browser

JSP Comments

```
<%-- JSP comment --%>
```

- Not evaluated by server
- Not part of the server response / web page
- Not visible in page source from web browser

Tag Libraries

- Java classes
- Implements the JSP Tag Extension API interfaces
- Usable as JSP tags in JSPs
- Java class coupled to a tag using a XML-descriptor
- Tags can control JSP processing

Implicit Objects

- **request** - HTTP request
- **response** - HTTP response
- **out** - response stream
- **session** - web application session
- pageContext - page context data
- application - Servlet context data
- config - Servlet configuration data
- page - Servlet object
- exception - exception data

Next Time

- JSP Web Development