CS520 Web Programming
Servlet and JSP Review

Chengyu Sun
California State University, Los Angeles

What We Won’t Talk About
(But Expect You to Know)

- Java
  - Use of collection classes like lists and maps
- HTML and CSS
  - Tables and forms
- Database access
  - Use of a DBMS
  - JDBC

URL

http://cs.calstatela.edu:8080/cysun/index.html

Static Web Pages

Browser

Web Application Development

- Server-side
  - CGI
  - C, Perl
  - Java EE
  - ASP.NET
    - VB, C#
  - PHP
  - Ruby
  - Python

- Client-side
  - HTML, CSS
  - JavaScript
  - Applet
  - Flash
### Directory Structure of a Java Web Application

- **Application Root Directory**
  - JSPs and static resources
  - WEB-INF
    - web.xml
    - classes
      - Compiled Java classes
    - lib
      - Compiled Java classes
      - Additional Java libraries

### Directory Structure on CS3

- **Application Root Directory**
  - www
  - WEB-INF
    - JSPs and static resources
    - classes
      - Compiled Java classes
    - lib
      - Compiled Java classes
      - Additional Java libraries

### Directory Structure of an Eclipse Dynamic Web Project

- **Application Root Directory**
  - WebContent
    - JSPs and static resources
    - WEB-INF
      - web.xml
      - classes
        - Compiled Java classes
      - lib
        - Compiled Java classes
        - Additional Java libraries

### Servlet HelloWorld

```java
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```

### Some Simple Observations

- Inherits from HttpServlet
  - [http://java.sun.com/products/servlet/2.5/docs/servlet-2.5-mr2/java/servlet/HttpServlet.html](http://java.sun.com/products/servlet/2.5/docs/servlet-2.5-mr2/java/servlet/HttpServlet.html)
- There's no `main()` method
- `doGet()` and `doPost()`
- `doGet()` and `doPost()`
  - Input: `HttpServletRequest`
  - Output: `HttpServletResponse` → sent back to the client browser

### About web.xml

- Web application deployment descriptor
  - `<welcome-file-list>`
  - `<servlet>` and `<servlet-mapping>`
    - `<load-on-startup>`
- More about web.xml in Java Servlet Specification
TCP/IP Monitor in Eclipse ...

Client request response Server

localhost host:port

TCP/IP Monitor request response Server

localhost host:port

HTTP Request Example

http://ics3.calstatela.edu:8080/whatever

GET /whatever HTTP/1.1
Host: cs3.calstatela.edu:4940
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.7.3) 
Accept: text/xml,application/xml,application/xhtml+xml,application/xenc+xml; 
Accept-Language: en-us,en;q=0.5
Accept-Encoding; gzip, deflate
Accept-Charset: ISO-8859-1UTF-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: net/gateway.dl/uid=4MCFP072; SITESERVER=ID=1675...

HTTP Request

- Request line
  - Method
  - Request URI
  - Protocol
- Header
- [Message body]

Request Methods

- Actions to be performed regarding the resource identified by the Request URI
  - Browser
    - GET
    - POST
  - Editor
    - PUT
    - DELETE
  - Diagnosis
    - HEAD
    - OPTIONS
    - TRACE

HttpServlet Methods

- service()
HttpServletRequest

- "http://java.sun.com/products/servlet/2.5/docs/servlet-2.5-mr2/javax/servlet/ServletRequest.html"

Use Request Parameters as Input

- Query string
  - ?param1=value1&param2=value2&...
- Form data
  - GET vs. POST

Servlet Examples

- Add
- GuestBook

Use Request URI as Input

- ?param1=value1&param2=value2
- /param1/value1/param2/value2

Session Tracking

- The Need
  - shopping cart, personalization, ...
- The Difficulty
  - HTTP is a "stateless" protocol
  - Even persistent connections only last seconds
- The Trick??

General Idea
Servlet Session Tracking API

- HttpServletRequest
  - HttpSession getSession()
- HttpSession
  - setAttribute( String, Object )
  - getAttribute( String )
  - setMaxInactiveInterval( int )
    - Tomcat default: 30 seconds
  - invalidate()

Example: Improved GuestBook

- A user only needs to specify a name when he or she adds the first comment.

HTTP Response Example

HTTP/1.1 200 OK
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 168
Date: Sun, 03 Oct 2004 18:26:57 GMT
Server: Apache-Coyote/1.1

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head><title>Servlet Life Cycle</title></head>
<body>
  n is 299 and m is 440
</body>
</html>

HTTP Response

- Status line
  - Protocol
  - Status code
- Header
  - [Message body]

Status Codes

- 100 – 199: Informational. Client should respond with further action
- 200 – 299: Request is successful
- 300 – 399: Files have moved
- 400 – 499: Error by the client
- 500 – 599: Error by the server

Common Status Codes

- 404 (Not Found)
- 403 (Forbidden)
- 401 (Unauthorized)
- 200 (OK)
Header Fields

- **Request**
  - Accept
  - Accept-Charset
  - Accept-Encoding
  - Accept-Language
  - Connection
  - Content-Length
  - Cookies

- **Response**
  - Content-Type
  - Content-Encoding
  - Content-Language
  - Connection
  - Content-Length
  - Set-Cookie

More Response Header Fields

- **Location**
  - for redirect

- **Refresh**
  - "Push"
  - Incremental display

- **Cache-Control, Expires, Pragma**
  - for cache policies

Example: File Download

- Download file using a servlet
  - Indicate file name
  - Indicate whether file should be displayed or saved

Sharing Data among Servlets

- **HttpServletRequest**
  - getServletContext()

- **HttpServletResponse**
  - setAttribute(String name, Object value)
  - getAttribute(String name)

Example: GuestBook Using Two Servlets

- Separate GuestBook into two servlets
  - GuestBook
  - AddComment

Scopes and Data Sharing

- **Application scope** – data is valid throughout the life cycle of the web application
- **Session scope** – data is valid throughout the session
  - redirect, multiple separate requests
- **Request scope** – data is valid throughout the processing of the request
  - forward
- **Page scope** – data is valid within current page
Access Scoped Variables in Servlet

- **Application scope**
  - `getServletContext()`
- **Session scope**
  - `request.getSession()`
- **Request scope**
  - `request`
- **Page scope (in JSP scriptlet)**
  - `pageContext`

Java Server Page (JSP)

- **Why?**
  - It’s tedious to generate HTML using `println()`
  - Separate presentation from processing
- **How?**
  - *Java code embedded in HTML documents*

HelloJSP.jsp

```html
<IOException PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>JSF Hello World</TITLE></HEAD>
<BODY>Hello World on <%= new java.util.Date() %>. </BODY>
</HTML>
```

How Does JSP Work?

- **Look under**
  - `$CATALINA_HOME/work/Catalina/localhost/context_name`

JSP Components

- **HTML template text**
- **Code elements of Java**
  - Directives
  - Scripting elements
  - Beans
  - Expression language
  - Custom tag libraries

Directives

- **Affect the overall structure of the JSP/servlet**
- `<%@ type attr="value" ... %>`
- **Three type of directives**
  - `page`
  - `include`
  - `taglib`
Directive Examples

```html
<%@ page import="java.util.*, java.util.zip.*" %>
<%@ page contentType="text/html" %>
<%@ page pageEncoding="Shift_JIS" %>
<%@ page session="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ include file="path_to_file" %>
```

Scripting Elements

- JSP Expression
- JSP Scriptlet
- JSP Declarations

Example: Add.jsp

- Convert Add servlet to JSP using scripting elements

Problems with Scripting Elements

- Mixing presentation and processing
  - hard to debug
  - hard to maintain
- No clean and easy way to reuse code

Solution – separate out Java code

Separate Data and Presentation

- Presentation
  - Create UI
  - Input and output
  - Web, JFC/Swing ...
- Data Models
  - Independent of UI
  - POJO (Beans)
  - E.g. the GuestBookEntry class

Model-View-Controller (MVC) Architecture

- A.K.A. Model 2 Architecture
About MVC

- Originate from the work on Smalltalk
- Widely used in GUI applications

MVC in a Web Application ...

... MVC in a Web Application

1. Process request
2. Create/update beans
3. Store beans in request, session, or application scope
4. Forward request to JSP page
5. Extract data from beans and display

Example: GuestBook Using MVC

- Model
  - GuestBookEntry.java
- View
  - AddComment.jsp, GuestBook.jsp
- Controller
  - AddComment.java, GuestBook.java

Java Beans

- A zero-argument constructor
- No public class variables
- Properties
  - Properties are defined by getter and/or setters, e.g. getFoo() and setFoo()
  - Properties != Class variables

About Bean Properties

- Property naming conventions
  - 1st letter is always in lower case
  - 1st letter must be capitalized in getter (accessor) and/or setter (mutator)
- Property types
  - read-only property: only getter
  - write-only property: only setter
  - read/write property: both
Expression Language

- Expression Language (EL)
  - A JSP 2.0 standard feature
  - A more concise way to write JSP expressions
    - vs. `<%= expression %>`
  - Java’s answer to scripting languages
    - e.g. associative array
- EL Syntax
  ```expression```

EL Literals

- `true, false`
- `23, 0x10, ...`
- `7.5, 1.1e13, ...`
- "double-quoted", 'single-quoted'
- `null`
- No char type

EL Operators

- Arithmetic
  - `+, -, *, /, %`
- Logical
  - `&&, ||, !`
- and, or, not
- Relational
  - `=, !=, <, >, <=, >=`
- `eq, ne, lt, gt, le, ge`
- `Conditional`
  - `?:`
- `empty`
  - check whether a value is null or empty
- `Other`
  - `[[], {}]`

EL Evaluation and Auto Type Conversion

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(2+4)/2</td>
<td>3</td>
</tr>
<tr>
<td>$(2+3)/2</td>
<td>3</td>
</tr>
<tr>
<td>$(2*3)/2</td>
<td>3</td>
</tr>
<tr>
<td>$(2*3) div 2</td>
<td>3</td>
</tr>
<tr>
<td>$(&quot;a&quot; + 3 div 2)</td>
<td>3</td>
</tr>
<tr>
<td>$(null eq 'test')</td>
<td>3</td>
</tr>
<tr>
<td>$(null eq 'null')</td>
<td>3</td>
</tr>
</tbody>
</table>

EL Variables

- You cannot declare new variables using EL (after all, it’s called “expression” language).
- However, you can access beans, implicit objects, and previously defined scoped variables.
Implicit Objects

- pageContext
- servletContext
- session
- request
- response
- param, paramValues
- header, headerValues
- cookie
- initParam

pageScope
requestScope
sessionScope
applicationScope

Limitations of EL

- Only expressions, no statements, especially no control-flow statements

JSTL Example

```html
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html><head><title>JSTL Hello</title></head>
<body>
<c:out value="Hello World in JSTL." />
</body>
</html>
```

JSTL Core

- URL
  - <c:param>
  - <c:redirect>
  - <c:import>
  - <c:useBean>
  - <c:catch>

- Output
  - <c:out>

- Exception handling
  - <c:catch>

- Flow control
  - <c:if>
  - <c:choose>
    - <c:when>
    - <c:otherwise>
  - <c:forEach>
  - <c:forToken>

JSTL Standard Tag Library (JSTL)

<table>
<thead>
<tr>
<th>Library</th>
<th>URI</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td><a href="http://java.sun.com/jsp/jstl/core">http://java.sun.com/jsp/jstl/core</a></td>
<td>c</td>
</tr>
<tr>
<td>XML Processing</td>
<td><a href="http://java.sun.com/jsp/jstl/xml">http://java.sun.com/jsp/jstl/xml</a></td>
<td>x</td>
</tr>
<tr>
<td>I18N Formatting</td>
<td><a href="http://java.sun.com/jsp/jstl/fmt">http://java.sun.com/jsp/jstl/fmt</a></td>
<td>fmt</td>
</tr>
<tr>
<td>Database Access</td>
<td><a href="http://java.sun.com/jsp/jstl/sql">http://java.sun.com/jsp/jstl/sql</a></td>
<td>sql</td>
</tr>
<tr>
<td>Functions</td>
<td><a href="http://java.sun.com/jsp/jstl/functions">http://java.sun.com/jsp/jstl/functions</a></td>
<td>fn</td>
</tr>
</tbody>
</table>

Branch Tags

```xml
<if test="$！{cart.notEmpty}" > The cart is empty.</if>

<choose>
  <when test="$！{cart.notEmpty}" >
    The cart is empty.
  </when>
  <otherwise>
    %&gt; do something --%>
  </otherwise>
</choose>
```

Loop Tags

```xml
<forEach style="&gt; iterator style --%>
  &gt; $！{cart.items} var="i"&gt;
  $（i） &lt;br&gt;
</forEach>

<forEach style="&gt; for loop style --%>
  &gt; $！{cart.size} end="$！{cart.size}" step="1" var="i"&gt;
  &gt; $！{cart.items（i）}
</forEach>
```

Set and Remove Scope Variables

In Login.jsp

```xml
<set var="authorized" value="true" scope="session"/>
```

In CheckLogin.jsp

```xml
<if test="$！{empty sessionScope.authorized}" >
  <redirect url="Login.jsp" />
</if>
```

URL Tags

```xml
<import url="/books.xml" var="something" />  
<x:parse doc="$！{something}" var="booklist" scope="application" />

<url var="url" value="/catalog" >
  &lt;param name="Add" value="$！{bookId}" />
</url>
<a href="$！{url}" >Get book</a>
```

Output

```xml
<out value="100" />
<out value="$！{price}" />
```

- You want to use `<out>` if
  - `escapeXML=true`
  - `value` is a Java.io.Reader object

Character Conversion

- When `escapeXML=true`

<table>
<thead>
<tr>
<th>Character</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>'</td>
<td>'</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
Exception Handling

```<c:catch>```nn

About MVC

- Servlets do NOT generate HTML directly
  - No `out.println()`
  - Redirect and Forward
- JSPs are only used for display
- Use of scopes
  - Application and session scopes are shared by all servlets and JSPs
  - Request scope is used for passing data from a servlet to a JSP

Summary

- Server-side Programming
  - ASP, PHP
  - Servlet
  - ...nn
  - JSP with Scripting Elements
    - Bean (Business logic)
    - EL (Property Access)
    - Tag Library (Display Logic)
- Filter
  - `web.xml`
- Java Web Application
- Static content
- Other libraries

Web App Development – Where Do We Start?

- Control flow driven approach

Data driven approach

1. Models
2. Database Schema
3. Application