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

HTTP request

HTTP response

Welcome.html

Welcome.html

index.html

Deliver Dynamic Content

Application Server

Browser

HTTP request

HTTP response

Program

input output

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
  - JSPs and static resources
    - WEB-INF
      - web.xml
      - classes
        - Compiled Java classes
      - lib
        - Compiled Java classes
        - Additional Java libraries
  - www

Directory Structure of an Eclipse Dynamic Web Project

- Application Root Directory
  - JSPs and static resources
    - WEB-INF
      - web.xml
      - classes
        - Compiled Java classes
      - lib
        - Compiled Java classes
        - Additional Java libraries
      - Servlet HelloWorld
        - import java.io.*;
        - import javax.servlet.*;
        - import javax.servlet.http.*;
        - 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
- Web application deployment descriptor
- There's no main() method
- doGet() and doPost()
  - Input: HttpServletRequest
  - Output: HttpServletResponse \(\rightarrow\) sent back to the client browser
- <welcome-file-list>
- <servlet> and <servlet-mapping>
- <load-on-startup>
- More about web.xml in Java Servlet Specification

About web.xml

- Web application deployment descriptor
- <welcome-file-list>
- <servlet> and <servlet-mapping>
- <load-on-startup>
**Wildcard in Servlet Mapping**

- A string beginning with a `/` and ending with a `/*`
  - E.g. `/`, `/content/*`
- A string beginning with a `*`
  - E.g. `*.html`, `*.do`

See Servlet Specification 2.4, Section SRV.11.2

---

**TCP/IP Monitor in Eclipse**

1. Client → request → TCP/IP Monitor → response → Server
2. Client → request → TCP/IP Monitor → response → Server

---

**... TCP/IP Monitor in Eclipse**

- Window → Preferences → Run/Debug → TCP/IP Monitor
- Example: monitor the access of [http://sun.calstatela.edu/~cysun/public/form.html](http://sun.calstatela.edu/~cysun/public/form.html)
  - Local Monitoring Port?? Host?? Port??
  - Browser URL??

---

**HTTP Request Example**

```
GET /whatever HTTP/1.1
Host: cs3.calstatela.edu:8080
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,text/html;q=0.9,...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: nct/gateway.dll/uid=4B4CF072; SITESERVER=ID=f1675...
```

---

**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**

<table>
<thead>
<tr>
<th>Method</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>doGet()</td>
</tr>
<tr>
<td>POST</td>
<td>doPost()</td>
</tr>
<tr>
<td>PUT</td>
<td>doPut()</td>
</tr>
<tr>
<td>DELETE</td>
<td>doDelete()</td>
</tr>
<tr>
<td>HEAD</td>
<td>doHead()</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>doOptions()</td>
</tr>
<tr>
<td>TRACE</td>
<td>doTrace()</td>
</tr>
</tbody>
</table>

**HttpServletResponse**


**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

- Client
- Request
- Response + Session ID (sid)
- Request + sid
- Request + sid
- Request + sid
- Request + sid

Servlet Session Tracking API

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

Example: Improved GuestBook

- A use 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

Example:

```
<!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

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

Sharing Data among Servlets
- HttpServlet
  - getServletContext()
- HttpServletRequest
  - 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
<DOCTYPE HTML 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

```jsp
<%@ 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 Declaration: `<%! ... %>`

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        Data Models
  ?
```

Model-View-Controller (MVC) Architecture

- A.K.A. Model 2 Architecture

```
Java Web Application

View  Controller  Model

JSP  Servlet  Bean

GuestBookEntry class
```
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
  - `+, -`, `/`, `%`
- `div`, `mod`
- 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><code>$(2+4/2)</code></td>
<td>$1`</td>
</tr>
<tr>
<td><code>$(2+3/2)</code></td>
<td>$2`</td>
</tr>
<tr>
<td><code>$(2^2+3</code> div 2)</td>
<td>$3`</td>
</tr>
<tr>
<td><code>$(&quot;a&quot; + 3</code> div 2)</td>
<td>$4`</td>
</tr>
<tr>
<td><code>$(null == 'test')</code></td>
<td>$5`</td>
</tr>
<tr>
<td><code>$(null </code>eq null<code>)</code></td>
<td>$6`</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.

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

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>
  &lt;c:out value="Hello World in JSTL." />
</body>
</html>
```

JSTL Core

- URL
  - &lt;c:param>
  - &lt;c:redirect>
  - &lt;c:import>
  - &lt;c:url>
- Output
  - &lt;c:out>
- Exception handling
  - &lt;c:catch>

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>


taglib Directive

- URI
  - A unique identifier for the tag library
- Prefix
  - NOT a real URL
  - A short name for the tag library
  - Could be an arbitrary name
Branch Tags

<c:if test="${cart.notEmpty}"> The cart is empty.</c:if>

<c:choose>
  <c:when test="${cart.notEmpty}">
    The cart is empty.
  </c:when>
  <c:otherwise>
    %-- do something --%
  </c:otherwise>
</c:choose>

Set and Remove Scope Variables

In Login.jsp

<c:set var="authorized" value="true" scope="session"/>

In CheckLogin.jsp

<c:if test="${empty sessionScope.authorized}"
    <c:redirect url="Login.jsp"/>
</c:if>

URL Tags

<c:import url="/books.xml" var="something"/>

<x:parse doc="${something}"
  var="booklist"
  scope="application"/>

<c:url var="url" value="/catalog"

  <c:param name="Add" value="${bookId}"/>
</c:url>

<a href="${url}">Get book</a>

Output

<c:out value="100"/>
<c:out value="${price}"/>

When escapeXML=true

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;</code></td>
<td>&lt;</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>&gt;</td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>&amp;</td>
</tr>
<tr>
<td><code>'</code></td>
<td>'</td>
</tr>
<tr>
<td><code>&quot;</code></td>
<td>&quot;</td>
</tr>
</tbody>
</table>
Exception Handling

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

Format Date and Time

```xml
<fmt:formatDate value="${date}" type="date" />
<fmt:formatDate value="${date}" type="time" />
<fmt:formatDate value="${date}" type="both" />
<fmt:formatDate value="${date}" pattern="yyyy-MM-dd hh:mm:ss a" />
```

See [http://java.sun.com/javase/6/docs/api/java/text/SimpleDateFormat.html](http://java.sun.com/javase/6/docs/api/java/text/SimpleDateFormat.html) for the date formatting patterns.

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

JSP

Scripting Elements

Bean

EL

Tag Library

(Property Access)

(Display Logic)

Filter

web.xml

Java Web Application

Static content

Other libraries

Web App Development – Where Do We Start?

- Control flow driven approach

Web App Development – Where Do We Start?

- Data driven approach

1. Models

2. Database Schema

3. Application