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

Web Application Development

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

- **Client-side**
  - HTML, CSS
  - JavaScript
  - Applet
  - Flash
Dynamic Web Project in Eclipse

- **build**: generated files
  - **classes**: compiled Java classes
- **src**: source code
- **WebContent**: other resources (HTML pages, images, JavaScript ...); root directory of the web application
  - **WEB-INF**: cannot be accessed remotely
    - **lib**: Library jar files
    - **web.xml**: web application deployment descriptor

More About web.xml

- **web.xml** in CSNS
- **Java Servlet 2.4 Specification**
  - SRV.13.4

Servlet Hello World

```java
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" );
    }
}
```

Program I/O

- **Input**: HTTP Request
- **Output**: HTTP Response

TCP/IP Monitor in Eclipse

HTTP Request Example

```
http://cs3.calstatela.edu:4040/whatever
```

**GET** /whatever HTTP/1.1
Host: cs.calstatela.edu:4040
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: nxt/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

HttpException Methods

- service()
  - GET ➔ doGet()
  - POST ➔ doPost()
  - PUT ➔ doPut()
  - DELETE ➔ doDelete()
  - HEAD ➔ doHead()
  - OPTIONS ➔ doOptions()
  - TRACE ➔ doTrace()

HttpServletRequest

- getThis(), getThat(), ...
- [Link to documentation]

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

Three Ways to Implement Session Tracking

◆ URL Re-writing
◆ Hidden form field
◆ Cookies

Cookies

◆ Issued by the server
  ■ HTTP Response: Set-Cookie
◆ Part of the next client request
  ■ HTTP Request: Cookie

Cookie Attributes

◆ Name, Value
◆ Host/Domain, Path
◆ Require secure connection
◆ Max age
◆ Comment (Version 1)
### Servlet Cookie API

**Cookie**
- `getThis()`, `setThat()` ...
- `setMaxAge( int )`
  - `1000??`, `-1??`, `0??`

**HttpServletResponse**
- `addCookie( Cookie )`

**HttpServletRequest**
- `Cookie[] getCookies()`

### Servlet Session Tracking API

**HttpServletRequest**
- `getSession()`

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

### Example: Improved GuestBook

**A use only need to specify a name when he or she leaves the first message**

### Scopes and Data Sharing

- **page scope** – data is valid within current page
  - include
- **request scope** – data is valid throughout the processing of the request
  - forward
- **session scope** – data is valid throughout the session
  - redirect, multiple separate requests
- **application scope** – data is valid throughout the life cycle of the web application

### Access Scoped Variables in Servlet

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

### Scoped Variable Example

**A separate AddComment servlet for GuestBook**
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

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>JSP 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

Some Simple Observations about the JSP/Servlets

- In package org.apache.jsp
- _jspService() handles everything
- HTML text ➔ out.write(…)
- A number of pre-defined variables
  - request, response, out
  - config, pageContext
  - page, session, application

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 types of directives
  - page
  - include
  - taglib

Directive Examples

- `<%@ 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" %>`

Comments

- `<%-- Hidden Comments --%>`
- `<!-- Output (HTML) Comments -->`

Scripting Elements

- JSP Expression
- JSP Scriptlet
- JSP Declarations

Scripting Elements Example

- RequestCounter servlet
- RequestCounter in JSP with scripting elements

JSP Expression

- `<%= Java expression %>`
  - What’s an expression??
- `Converted to out.print(...) in _jspService()`
JSP Scriptlet

- `<% Java code %>`
- All code goes into `_jspService()`

JSP Declaration

- `<%! class variables or methods %>`
- All code goes outside `_jspService()`

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

Java Beans

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

(*) Only needed if the bean is created in a JSP using `<jsp:useBean>`
(****) You can have public class variables, but they can’t be accessed directly in JSP.

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

Bean Tags and Attributes

- `<jsp:useBean>`
  - `class`
  - `id`
  - `scope`
    - `page` (default)
    - `request`
    - `session`
    - `application`
- `<jsp:getProperty>`
  - `name`
  - `property`
- `<jsp:setProperty>`
  - `name`
  - `property`
  - `value`
  - `param`
Simple Bean Example
- RequestCounter using a Counter bean

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 }`

Expression
- Literals
- Operators
- Variables
- Functions
  - see Custom Tag Libraries

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
- `${2+4/2}`
- `${2+3/2}`
- `${(2+2)*2}`
- `${empty param.a}`
- `${empty null}`
- `${empty "null"}`
- `${"abc" lt "bc"}`
- `${null eq 'test'}`
- `${null eq "null"}`
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
- sessionContext
- session
- request
- response
- param, paramValues
- header, headerValues
- cookie
- initParam
- pageScope
- requestScope
- sessionScope
- applicationScope

Simple EL Example

RequestCounter that shows visitor’s IP address

Limitations of EL

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

JSTL Example

```xml
<%@ 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>
```
taglib Directive

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

```text
<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>
```

Loop Tags

```text
<%-- iterator style --%>
<c:forEach items="${cart.items}" var="i">
  ${i} <br>
</c:forEach>

<%-- for loop style --%>
<c:forEach begin="0" end="${cart.size}" step="1" var="i">
  ${cart.items[i]}
</c:forEach>
```

Set and Remove Scope Variables

In Login.jsp

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

In CheckLogin.jsp

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

URL Tags

```text
<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" />  \rightarrow  \$100
<c:out value="$\{price\}" />  \rightarrow  \$\{price\}
```

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

Character Conversion

When `escapeXML=true`

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></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>
```

Bean + EL + JSTL Example

```
GuestBook.jsp
```

Filter

```
Intercept, examine, and/or modify request and response
```

Filter Example

```
CheckParamFilter
- Check whether a request comes with certain parameters
```
Putting It All Together - Java Web Application

- Components
  - Servlets
  - Filters
  - JSPs
  - Classes
  - Static documents (HTML, images, sounds etc.)
  - Meta information
- Everything in the same context is considered part of one application

Model 1 Architecture

- JSPs + Java beans
  - JSPs for presentation
  - beans for business logic

Model 2 Architecture

- Also know as Model-View-Controller (MVC) architecture
  - JSPs + beans + servlet
  - Beans for business logic – Model
  - JSPs for presentations – View
  - servlet for web logic – Controller
    - HTTP related processing, e.g. request, response, sessions etc.
    - Request dispatching

MVC Control Flow ...

... MVC Control Flow

1. Process request
2. Populate beans
3. Store results in request, session, or servlet context
4. Forward request to JSP page
5. Extract bean data from beans and display

GuestBook (MVC Version)

- Model
  - GuestBook
- Controller
  - Display
  - Add comment
- View
  - Display
  - Add comment
**Need for Web Application Frameworks**
- Simplifying creation of controllers
- Front controller
- Input validation
- Error and exception handling
- Transaction support
- Integration of common libraries
- ...

**Some Java Web Application Frameworks**
- Struts
- Spring
  - [http://www.springframework.org/](http://www.springframework.org/)
  - More than a MVC framework
- WebWork, Tapestry, JSF, GWT, ...

**Web App Development – Where Do We Start?**
- Control flow driven approach

**Web App Development – Where Do We Start?**
- Data driven approach

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