Objectives

- SE Life Skills
- Loose Ends
- Expression Language (EL)
- JSP Standard Tag Library (JSTL)

Software Engineering Life Skills

- Version Control
- Debugging
  - Look at “log” files
- Demos reveal bugs
- Meeting deadlines

Loose Ends

- Markup languages: HTML and XML
- JSP Implicit Objects
  - Have used request, session, response
  - application is ServletContext

JSTL: JSP Standard Tag Library

- Implement basic, common functionality for typical presentation-layer tasks
  - Data formatting
  - Iterative or conditional content
- JSP authors can focus on application-specific development rather than generic operations

JSTL: JSP Standard Tag Library

- Not by default part of JSP Specification
- Need to include jstl.jar and standard.jar into the lib directory of your web application

Scriptlets

- Scriplets break up the HTML
  - Harder to read, debug, maintain

```jsp
<% if (user.getRole() == "member") { %>
<p>Welcome, member!</p>
<% } else { %>
<p>Welcome, guest!</p>
<% } %>
```
Custom Tag Libraries

- **Core**
  - Custom actions to manage data through scoped variables
  - Perform iteration & conditionalization of page content
  - Generate, operate on URLs
- **Format**
  - Format numbers and dates
  - Internationalization
- **XML**
  - For data represented in XML
- **SQL**
  - Query relational databases

Expression Language (EL)

- Provides identifiers, accessors, and operators for retrieving and manipulating data
- EL is loosely based on EcmaScript (a dialect of JavaScript) and the XML Path Language (XPath)
- Geared toward
  - Looking up objects and their properties
  - Performing simple operations on objects
- Not a programming or scripting language
  - When combined with the JSTL tags, enables complex behavior to be represented using a simple and convenient notation

Expressions in EL

- Delimited using a leading $, leading and trailing 

```
<%: user.firstName %>
```

- Combine multiple expressions with static text

```
<%: user.firstName %> ${user.lastName} /
```

Implicit Objects in EL: Scoped Variables

- Can retrieve objects (attributes) from scopes
  - Scope Names: pageScope, requestScope, sessionScope, applicationScope
- Name of variable is the attribute’s name
  ```
  ${sessionScope.user}
  ```
- If don’t specify the scope, it looks for attribute, starting at page up through application
  ```
  ${user}
  ```

Implicit Objects in EL

- Request parameters
  - param - mapping to parameter values as Strings
  - paramValues - mapping to parameter values as String arrays
- Request headers
  - header - mapping to header values as Strings
  - headerValues - mapping to header values as String arrays
- Cookie
  - cookie - mapping to cookies as Strings
- Initialization parameters
  - initParam - Web app’s context parameters

Accessors: Dot operator

- Access object’s properties using . operator
  ```
  ${user.firstName} /
  ```
  - Implies that there is a method `getFirstName()`
- Can be used recursively
  ```
  ${user.address.city}
  ```
Accessors: Bracket operator

- Access array or collection's elements using [ ] operator

$$array\_name[3]$$  $$map\_name["key"]$$

Could contain expressions

Null values in EL

- In the following expression, if user or address is null, the whole expression evaluates to null

$$user\_address.city$$

> No NullPointerExceptions

Operators

- Arithmetic operators

$$item.price * (1 + tax\_Rate[address.zipcode])$$

- Logical and relational operators

$$x >= min \&\& x <= max$$

> "eq", "ne", "lt", "gt", "le", and "ge" could also be used as relational operators

taglib Directives

- Include directive in JSP to use library's tags

Prefix for library's tags

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

Which library to use (Core)

Core Library: Variable Tags

- `c:set`: Create, set scoped variables

```xml
<%=
set var="name" scope="scope" value="expression"%>
```

> scope attribute is optional

* default is page

```xml
<%=
set var="square" value="param["x"] * param["x"]%>
```

```xml
<%=
set var="timezone" scope="session" value="EST"%>
```

- Delete a variable using

```xml
<%=
remove var="timezone" scope="session"%>
```

Core Library Output: `<c:out>`

- Prints the result of evaluated value

```xml
<%=
out value="expression"
default="expression" escapeXml="boolean"%>
```

- Optional `default` attribute

> Print default if evaluated expression is null or an empty string

```xml
<%=
out value="$\{user\}" default="guest"%>
```

- Optional `escapeXml` attribute

> Displays XML (`<`, `>`, `&`, `'`, `"`) appropriately
cout.jsp

- Examples

Setting Variables with Default Values

- Use c:set and c:out

```html
<c:set var="timezone" scope="session">
<c:out value="${cookie['lang_pref'].value}" default="English"/>
</c:set>
```

Loops c:forEach

- Simplify iterating through arrays

```
<ul>
<c:forEach var="movie" items="${movieList}">
<li>${movie}</li>
</c:forEach>
</ul>
```

- Can nest c:forEach tags

Example

- Available Information in Scopes, etc

Alternative Version of c:forEach

- Traditional for loop:

```
<c:forEach var="i" begin="3" end="9">
</c:forEach>
```

Conditionals: c:if

- To do something based on a condition

```
<c:if test="user.role == 'student'">
  <h2>Student Options</h2>
  ...
</c:if>
```

- No else statement
Conditionals: `c:choose`

- Only one branch will execute

```xml
<cff:choose>
  <cff:when test="${isPrime}">
    <cff:out value="${i} is a prime number. "/>
  </cff:when>
  <cff:otherwise>
    <cff:out value="${i} is a not prime number. "/>
  </cff:otherwise>
</cff:choose>
```

Example

- Prime Numbers

JSTL: Just the Beginning!

- More tags in standard library available
  - Import, XML processing, formatting, SQL, ...
- JSTL tags: implemented by Java code
- Custom tag libraries available too
- You can even write custom tags!
  - Map tags to Java code

Project Status

- What we’ve done
  - Requirements, Static Mock-up, Preliminary Implementation
- What we need to do
  - Final implementation, Documentation, Demo
  - Decide on what’s “final”

What Did We Say We’d Do?

- Student
  - Practice, Real quizzes
  - View performance
- Professor
  - Create quizzes
  - Review student performance
- Admin
  - Create accounts
  - Delete accounts

- Everyone
  - Login/logout
  - Send password if forgotten
    - Actually reset password
  - Change password
  - Update anything else?
What Can We Complete?

• Goal
  ➢ Sell product to others (users, employers, etc.)
  ➢ Remind yourself what you did
• Outcome: Web pages
  ➢ Wiki or with Logic web application

Documentation Deliverable

• Application overview
  ➢ motivation, what, why, who, how
  ➢ Statistics? - how large an application
• Demonstration - e.g., live example, static mock-up, screenshots
• Design, implementation
  ➢ Components, interaction, interfaces (FIGURE!)
  ➢ Technologies used
• Installation, configuration information
• User manuals - precise, screenshots
• Future work: Suggestions for features

Understanding Design Decisions

• What were some of the design decisions made for the project?
  ➢ Why were they made?

  • ContextListener
    ➢ DB Connection Pool - shared resource, better performance, code in one place
  • AuthorizationFilter
    ➢ Security, code in one place
  • Code organization
    ➢ Servlets - “business logic”
    ➢ Act on Java classes - JDBC
    ➢ JSP - view oriented
    ➢ JavaScript

Project Code Base

• Metrics plugin
• 21 Classes
• 1241 Lines of Code
  ➢ Servlets, Model classes only
  ➢ Why not JSPs?
• Metrics
  ➢ Lack of Cohesion of Methods: 0.23
  ➢ McCabe Cyclomatic Complexity: 1.7
  ➢ Efferent Coupling: 3.2
TODO

- For Monday: JSTL in Project
- Project
  - Implementation
  - Documentation