

Objectives

- Expression Language (EL)
- JSP Standard Tag Library (JSTL)

May 13, 2016

Sprinkle - CSCI335

1

Review Ajax

- What is Ajax?
- What is it used for?
- How does it work?

May 13, 2016

Sprinkle - CSCI335

2

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

May 13, 2016

Sprinkle - CSCI335

3

JSTL: JSP Standard Tag Library



- Not by default part of JSP Specification
- Need to include `jstl.jar` and `jstl-api.jar` in the `lib` directory of your web application (`WEB-INF`)
 - Maven handles in AGP

May 13, 2016

Sprinkle - CSCI335

4

Issues with JSP Scriptlets

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

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

May 13, 2016

Sprinkle - CSCI335

5

Custom Tag Libraries

Core

- Custom actions to manage data through *scoped* variables
 - Perform iteration & conditionalization of page content
 - Generate, operate on URLs
- Functions
 - Common, helpful mostly String-related functions
 - XML
 - For data represented in XML
 - SQL
 - Query relational databases

May 13, 2016

Sprinkle - CSCI335

6

Expression Language (EL)

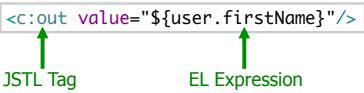
- Provides identifiers, accessors, and operators for retrieving and manipulating data
- EL is loosely based on EcmaScript (a dialect of JavaScript) and the XML Path Language (XPath)
- Geared toward
 - Looking up objects and their properties
 - Performing simple operations on objects
- Not a programming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

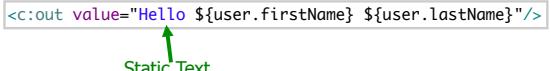
May 13, 2016

Sprinkle - CSCI335

7

Expressions in EL

- Delimited using a leading \$, leading and trailing {}


JSTL Tag EL Expression
- Combine multiple expressions with static text


Static Text

May 13, 2016

Sprinkle - CSCI335

8

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

- If don't specify the scope, it looks for attribute, starting at **page** up through **application**


Found in session scope

May 13, 2016

Sprinkle - CSCI335

9

Implicit Objects in EL

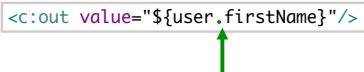
- Request parameters
 - param** - mapping to parameter values as Strings
 - paramValues** - mapping to parameter values as String arrays
- Request headers
 - header** - mapping to header values as Strings
 - headerValues** - mapping to header values as String arrays
- Cookie
 - cookie** - mapping to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

May 13, 2016

Sprinkle - CSCI335

10

Accessors: Dot operator

- Access object's properties using . operator


Implies that there is a method `getFirstName()`
- Can be used recursively


May 13, 2016

Sprinkle - CSCI335

11

Accessors: Bracket operator

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


Could contain expressions

May 13, 2016

Sprinkle - CSCI335

12

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

May 13, 2016

Sprinkle - CSCI335

13

Operators

- Arithmetic operators

```
 ${item.price * (1 + taxRate[user.address.zipcode])}
```

- Logical and relational operators

```
 ${x >= min} && (x <= max)}
```

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

May 13, 2016

Sprinkle - CSCI335

14

taglib Directives

- Include directive in JSP to use library's tags

Prefix for library's tags

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

Which library to use (Core)

May 13, 2016

Sprinkle - CSCI335

15

Core Library: Variable Tags

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

```
<c:set var="name" scope="scope" value="expression"/>
```

- scope attribute is optional

• default is page

```
<c:set var="square" value="${param['x']} * param['x']"/>
```

```
<c:set var="timezone" scope="session">EST</c:set>
```

- Delete a variable using

```
<c:remove var="timezone" scope="session"/>
```

optional

May 13, 2016

Sprinkle - CSCI335

16

Core Library Output: **<c:out>**

- Prints the result of evaluated **value**

```
<c:out value="expression"
       default="expression" escapeXml="boolean" />
```

- Optional **default** attribute

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

```
<c:out value='${user}' default='guest'/>
```

- Optional **escapeXml** attribute

- Displays XML (<, >, &, ‘, “) appropriately

May 13, 2016

Sprinkle - CSCI335

17

cout.jsp

- Examples

May 13, 2016

Sprinkle - CSCI335

18

Setting Variables with Default Values

- Use `c:set` and `c:out`

```
<c:set var="language" scope="session">
<c:out value="${cookie['lang-pref'].value}"
default="English"/>
</c:set>
```

May 13, 2016

Sprenkle - CSCI335

19

Loops `c:forEach`

- Simplify iterating through arrays

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

Loop variable
Array, Collection, Map, or comma-delimited String

- Can nest `c:forEach` tags

May 13, 2016

Sprenkle - CSCI335

20

Example

- Available Information in Scopes, etc

May 13, 2016

Sprenkle - CSCI335

21

Alternative Version of `c:forEach`

- Traditional for loop:

```
<c:forEach var="i" begin="${3}" end="${upperLimit}">
</c:forEach>
```

May 13, 2016

Sprenkle - CSCI335

22

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

May 13, 2016

Sprenkle - CSCI335

23

Conditionals: `c:choose`

- Only one branch will execute

```
<c:choose>
    <c:when test="#{isPrime}">
        <c:out value="#{i} is a prime number."/>
        <br />
    </c:when>
    <c:otherwise>
        <c:out value="#{i} is a not prime number."/>
        <br />
    </c:otherwise>
</c:choose>
```

Like an else

May 13, 2016

Sprenkle - CSCI335

24

Example

- Prime Numbers

May 13, 2016

Sprenkle - CSCI335

25

JSTL: Just the Beginning!

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

May 13, 2016

Sprenkle - CSCI335

26

PROJECT

May 13, 2016

Sprenkle - CSCI335

27

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" on Tues

May 13, 2016

Sprenkle - CSCI335

28

Documentation Deliverable Goals

- Description of final use cases -- how the application should be used.
 - Often, students unknowingly hide features. They know the application well, but the client doesn't.
- You should provide documentation about how to *install, configure, and run the application*.
 - For example, how to get the data into the database, any additional libraries/jar files needed, any configuration parameters-- what they represent, what are valid values, and where they need to be set.
- Where: README file <http://www.cs.wlu.edu/~sprenkle/cs335/project.php>
 - Could point to a page on the wiki

May 13, 2016

Sprenkle - CSCI335

29

Project Analysis

- Analysis of project, design
 - What went right
 - What went wrong
 - How you could improve in future
- Remind yourself what you did
- Future work: Suggestions for features
- See <http://www.cs.wlu.edu/~sprenkle/cs335/analysis.php>

May 13, 2016

Sprenkle - CSCI335

30

Fault Analysis

- Using Jira to document bugs
- Fault analysis

http://www.cs.wlu.edu/~sprenkle/cs335/assignments/bug_analysis.php

May 13, 2016

Sprenkle - CSCI335

31

TODO

- For Monday midnight: Project Implementation
 - Tuesday a.m. checkin
 - 10:30??
 - 11 ??
- Project
 - Documentation
 - Analysis – see course web site
- Bug Analysis – due Friday

May 13, 2016

Sprenkle - CSCI335

32