

CS320 Web and Internet Programming Java Beans and Expression Language (EL)

Chengyu Sun
California State University, Los Angeles

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
 - Servlets
 - Java Beans
 - Custom tag libraries

Java Beans

- ◆ A regular Java object, typically used for modeling data, e.g. `GuestBookEntry`
- ◆ A.K.A. POJO (Plain Old Java Object)

Bean Properties

- ◆ The properties of a bean are defined by getters and setters
- ◆ Properties != Class variables

```
public class User {  
    String firstname;  
    String lastname;  
  
    public getFirstname() { return firstname; }  
    public getLastname() { return lastname; }  
    public getName() { return firstname + " " + lastname; }  
}
```

Properties

- firstname
- lastname
- name

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 Property Example

- ◆ What properties does `FooBar` have?
 - Read-only: ??
 - Write-only: ??
 - Read/write: ??

```
public class FooBar {  
    private int a, b, c, d;  
    private boolean e;  
  
    public FooBar() { a = b = c = d = 0; }  
  
    public int getA() { return a; }  
    public void setA( int a ) { this.a = a; }  
  
    public int getB() { return b; }  
  
    public void setC( int c ) { this.c = c; }  
  
    public int getAB() { return a+b; }  
  
    public boolean isE() { return e; }  
    public void setE( boolean e ) { this.e = e; }  
}
```

Common Problems with Bean Property ...

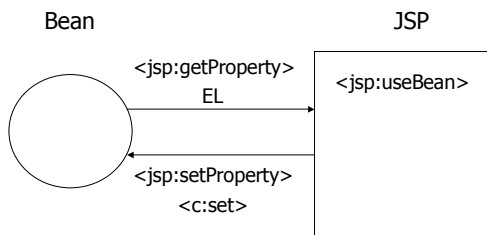
```
public class FooBar {  
    private int a, b, c, d;  
    public FooBar() { a = b = c = d = 0; }  
    public int getA() { return a; }  
    public void setA( String s ) { this.a = Integer.parseInt(s); }  
    public int getB( int x ) { return b+x; }  
    public void setC( int c, int x ) { this.c = c+x; }  
    public void setD( String s ) { this.d = Integer.parseInt(d); }  
}
```

How many properties does FooBar have??

... Common Problems with Bean Property

- ◆ A getter must have no argument
- ◆ A setter must have exactly one argument
- ◆ The *type* of a property must be consistent in both the getter and the setter

Bean and JSP



Bean Tags and Attributes

- ◆ `jsp:useBean`
 - class
 - id
 - scope
 - page (default)
 - request
 - session
 - application
- ◆ `jsp:getProperty`
 - name
 - property
- ◆ `jsp:setProperty`
 - name
 - property
 - value
 - param

Example: BGColor.jsp

- ◆ Use a bean `BGBean` to control the background color of a JSP page

```
<jsp:useBean id="bg" class="cs320.bean.BGBean" />  
<jsp:getProperty name="bg" property="r" />  
<jsp:setProperty name="bg" property="r" value="255" />  
<jsp:setProperty name="bg" property="r" param="r" />  
<jsp:setProperty name="bg" property="r" />  
<jsp:setProperty name="bg" property="*" />
```

Example: Counters.jsp

- ◆ Use `Counter` bean
- ◆ How do we increment the counter??
- ◆ Understand scopes
 - Application
 - Session
 - Request
 - Page

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
 - `ServletContext`
- ◆ Session scope
 - `HttpSession`
- ◆ Request scope
 - `HttpServletRequest`
- ◆ Page scope (in JSP scriptlet)
 - `pageContext`

Need for EL

- ◆ Using `<jsp:getProperty>` to access bean properties is tedious



EL

What is EL?

- ◆ Expression Language (EL)
 - Part of the JSP 2.0 Specification
 - A more concise way to access bean properties and write JSP expressions
 - ◆ vs. `<jsp:getProperty>`
 - ◆ vs. `<%= expression %>`
 - Java's answer to scripting languages
- ◆ Syntax: `${ expression }`

Example: BGColor.jsp Revisited

- ◆ Use EL to access the bean properties

```
${ bean_name.property_name }
```

Example: ShowCookie.jsp

- ◆ Display the value of the cookie `JSESSIONID`
 - PHP: `echo $HTTP_COOKIE_VARS[JSESSIONID]`
 - JSP Scriptlet??
 - EL

Expression

- ◆ Literals
- ◆ Operators
- ◆ Variables

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

<code>\${2+4/2}</code>		<code>\${empty ""}</code>	
<code>\${2+3/2}</code>		<code>\${empty param.a}</code>	
<code>\${"2"+3/2}</code>		<code>\${empty null}</code>	
<code>\${"2"+3 div 2}</code>		<code>\${empty "null"}</code>	
<code>\${"a" + 3 div 2}</code>		<code>\${"abc" lt "b"}</code>	
<code>\${null == "test"}</code>		<code>\${"cs320" > "cs203"}</code>	
<code>\${null eq "null"}</code>			

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 in EL

- ◆ pageContext
 - servletContext
 - session
 - request
 - response
- ◆ param, paramValues
- ◆ header, headerValues
- ◆ cookie
- ◆ initParam
- ◆ pageScope
- ◆ requestScope
- ◆ sessionScope
- ◆ applicationScope

Example: RequestInfo.jsp

- ◆ Display some information about the request
 - Client address ...
 - Cookies and parameters
- ◆ Use of implicit objects
 - Find the Java class type for the object
 - Look for *getters* in the API
 - E.g. `${request.remoteAddr}`
 - Access elements in a collection
 - `cookie` and `param`

Limitation of EL

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



JSTL