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
- Custom taglib

Java Beans for Dummies

- 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

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: ??

```java
public class Foobar {
    private int a, b, c, d;
    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 void setD( String s ) { this.d = Integer.parseInt(s); }
}
```

Common Problems with Bean Property ...

```java
public class Foobar {
    private int a, b, c, d;
    public Foobar() { a = b = c = d = 0; }
    public int getA() { return a; }
    public void setA( int a ) { 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(s); }
}
```
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
<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="*" />
```

Example: RequestCounter2.jsp

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

Need for EL

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

```el
<jsp:getProperty name="bg" property="r" />
```
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

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

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(2+4/2)$</td>
<td>$2$</td>
</tr>
<tr>
<td>$(2+3/2)$</td>
<td>$2.5$</td>
</tr>
<tr>
<td>$(2^2+3)$</td>
<td>$7$</td>
</tr>
<tr>
<td>$(x^2+3\ \text{div} \ 2)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$(x^4+3 \ \text{div} \ 2)$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(\text{null} == \text{test})$</td>
<td>$\text{false}$</td>
</tr>
<tr>
<td>$(\text{null} \eq \text{null})$</td>
<td>$\text{true}$</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.

Implicit Objects in EL

- `pageContext`: servletContext, session, request, response
- `param`, `paramValues`
- `header`, `headerValues`
- `cookie`
- `initParam`

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`

Example: ShoppingCart.jsp

- Store bean and Cart bean
  - Simple properties
    - E.g. number of products in the cart
  - Collection properties
    - E.g. list of products
- Display, add to, and remove from cart

Limitation of EL

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

JSTL