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

```java
public class User {
    String firstname;
    String lastname;

    public void setFirstName(String name) { firstname = name; }
    public String getFirstName() { return firstname; }

    public void setLastName(String lastname) { this.lastname = lastname; }
    public String getLastName() { return lastname; }
}
```

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;
    private boolean e;

    public Foobar() {
        a = b = c = d = 0;
        e = false;
    }

    public int getA() { return a; }
    public void setA(int a) { this.a = a; }

    public int getB() { return b; }
    public void setB(int b) { this.b = b; }

    public int getC() { return c; }
    public void setC(int c) { this.c = c; }

    public int getD() { return d; }
    public void setD(int d) { this.d = d; }

    public boolean isE() { return e; }
    public void setE(boolean e) { this.e = e; }
}
```
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( String s ) { this.a = Integer.parseInt(s); }
    public void setB( int x ) { this.b = 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?

Bean and JSP

```
<jsp:getProperty>
    <c:set>
    </c:set>
</jsp:getProperty>
```

Bean Tags and Attributes

```
<jsp:useBean
    <jsp:getProperty
    <jsp:setProperty
```

Example: BGColor.jsp

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

Understand <jsp:useBean> and <jsp:getProperty>

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

The bean class used in <jsp:useBean> must have a Constructor that takes no argument.
Set Bean Properties

```jsp
<jsp:setProperty name="bg" property="r" value="255" />
<jsp:setProperty name="bg" property="r" param="r" />
<jsp:setProperty name="bg" property="r" value="255" />
```

Understand Scopes

- **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

Scopes 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

```
<%= expression %>
```

What is EL?

- **Expression Language (EL)**
  - Since 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 }
```
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

```
$2+4/2)
$2+3/2)
$2*3/2)
$2*3 div 2)
$2*3 div 2)
$null == 'test')
$null eq 'null')

$empty "")
$empty param.a)
$empty null)
$"abc" lt 'b')
$"ca320" > "ca203")
```

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. ${pageContext.request.remoteAddr}
  - Access elements in a collection
    - cookie and param

Limitation of EL

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

JSTL