

Using JavaBeans with JSP

Lecture 11 – Chapter 13

Core Servlets & JSP book: www.coreservlets.com
 More Servlets & JSP book: www.moreservlets.com
 Servlet and JSP Training Courses: courses.coreservlets.com

Slides © Marty Hall, <http://www.coreservlets.com>, book © Sun Microsystems Press


Agenda

- Overview of beans in Java
- Basic use of beans in JSP
- Creating and accessing beans
- Setting bean properties explicitly
- Associating individual bean properties with request parameters
- Associating all bean properties with request parameters
- Conditional bean operations
- Sharing beans among multiple JSP pages and servlets

www.coreservlets.com

Uses of JSP Constructs

Simple Application

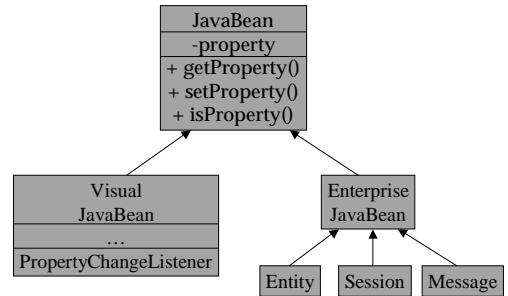


Complex Application

- Scripting elements calling servlet code directly
- Scripting elements calling servlet code indirectly (by means of utility classes)
- Beans
- Custom tags
- Servlet/JSP combo (MVC), with beans and possibly custom tags

www.coreservlets.com

Bean Overview



```

classDiagram
    class JavaBean {
        -property
        +getProperty()
        +setProperty()
        +isProperty()
    }
    class VisualJavaBean {
        -PropertyChangeListener
    }
    class EnterpriseJavaBean {
    }
    class Entity
    class Session
    class Message
    JavaBean <|-- VisualJavaBean
    JavaBean <|-- EnterpriseJavaBean
    EnterpriseJavaBean <|-- Entity
    EnterpriseJavaBean <|-- Session
    EnterpriseJavaBean <|-- Message
  
```

www.coreservlets.com

Background: What Are Beans?

- **Java classes that follow certain conventions**
 - Must have a zero-argument (empty) constructor
 - You can satisfy this requirement either by explicitly defining such a constructor or by omitting all constructors
 - Should have no public instance variables (fields)
 - I hope you already follow this practice and use accessor methods instead of allowing direct access to fields
 - Persistent values should be accessed through methods called getXxx and setXxx
 - If class has method getTitle that returns a String, class is said to have a String *property* named title
 - Boolean properties use isXxx instead of getXxx
 - For more on beans, see <http://java.sun.com/beans/docs/>

www.coreservlets.com

Why You Should Use Accessors, Not Public Fields

- To be a bean, you cannot have public fields
- So, you should replace


```
public double speed;
```
- with


```
private double speed;
public double getSpeed() {
    return(speed);
}
public void setSpeed(double newSpeed) {
    speed = newSpeed;
}
```
- You should do this in *all* your Java code anyhow. Why?

www.coreservlets.com

Why You Should Use Accessors, Not Public Fields

- 1) You can put constraints on values

```
public void setSpeed(double newSpeed) {
    if (newSpeed < 0) {
        sendErrorMessage(...);
        newSpeed = Math.abs(newSpeed);
    }
    speed = newSpeed;
}
```

- If users of your class accessed the fields directly, then they would each be responsible for checking constraints.

Using Beans

www.coreservlets.com

Why You Should Use Accessors, Not Public Fields

- 2) You can change your internal representation without changing interface

```
// Now using metric units (kph, not mph)
```

```
public void setSpeed(double newSpeed) {
    setSpeedInKPH = convert(newSpeed);
}
```

```
public void setSpeedInKPH(double newSpeed) {
    speedInKPH = newSpeed;
}
```

Using Beans

www.coreservlets.com

Why You Should Use Accessors, Not Public Fields

- 3) You can perform arbitrary side effects

```
public double setSpeed(double newSpeed) {
    speed = newSpeed;
    updateSpeedometerDisplay();
}
```

- If users of your class accessed the fields directly, then they would each be responsible for executing side effects. Too much work and runs huge risk of having display inconsistent from actual values.

Using Beans

www.coreservlets.com

Basic Bean Use in JSP

- **Format**

```
<jsp:useBean id="name" class="package.Class" />
```

- **Purpose**

- Allow instantiation of Java classes without explicit Java programming (XML-compatible syntax)

- **Notes**

- Simple interpretation: JSP action
<jsp:useBean id="book1" class="coreservlets.Book" />
can be thought of as equivalent to the scriptlet
<% coreservlets.Book book1 = new coreservlets.Book(); %>
- But useBean has two additional advantages:
 - It is easier to derive object values from request parameters
 - It is easier to share objects among pages or servlets

Using Beans

www.coreservlets.com

Accessing Bean Properties

- **Format**

```
<jsp:getProperty name="name" property="property" />
```

- **Purpose**

- Allow access to bean properties (i.e., calls to getXxx methods) without explicit Java programming

- **Notes**

```
<jsp:getProperty name="book1" property="title" />
```

is equivalent to the following JSP expression

```
<%= book1.getTitle() %>
```

Using Beans

www.coreservlets.com

Setting Bean Properties: Simple Case

- **Format**

```
<jsp:setProperty name="name"
    property="property"
    value="value" />
```

- **Purpose**

- Allow setting of bean properties (i.e., calls to setXxx methods) without explicit Java programming

- **Notes**

```
<jsp:setProperty name="book1"
    property="title"
    value="Core Servlets and JavaServer Pages" />
```

is equivalent to the following scriptlet

```
<% book1.setTitle("Core Servlets and JavaServer Pages"); %>
```

Using Beans

www.coreservlets.com

Example: StringBean

```
package coreservlets;

public class StringBean {
    private String message = "No message specified";

    public String getMessage() {
        return(message);
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

- Installed in normal servlet directory
 - `Tomcat_install_dir\webapps\ROOT\WEB-INF\classes\coreservlets`
 - `JRun_install_dir\servers\default\default-app\WEB-INF\classes\coreservlets`

Using Beans

www.coreservlets.com

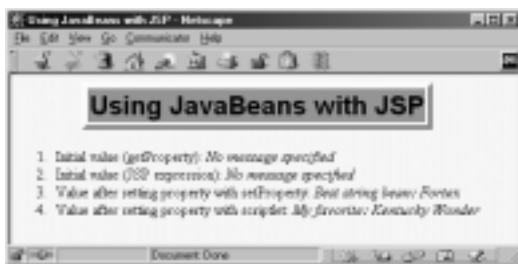
JSP Page That Uses StringBean

```
<jsp:useBean id="stringBean"
             class="coreservlets.StringBean" />
<OL>
<LI>Initial value (getProperty):
    <I><jsp:getProperty name="stringBean"
                      property="message" /></I>
<LI>Initial value (JSP expression):
    <I><%= stringBean.getMessage() %></I>
<LI><jsp:setProperty name="stringBean"
                    property="message"
                    value="Best string bean: Fortex" />
    Value after setting property with setProperty:
    <I><jsp:getProperty name="stringBean"
                      property="message" /></I>
<LI><%=stringBean.setMessage("My favorite: Kentucky Wonder");%>
    Value after setting property with scriptlet:
    <I><%= stringBean.getMessage() %></I>
</OL>
```

Using Beans

www.coreservlets.com

JSP Page That Uses StringBean



Using Beans

www.coreservlets.com

Setting Bean Properties Case 1: Explicit Conversion & Assignment

```
<!DOCTYPE ...>
...
<jsp:useBean id="entry"
             class="coreservlets.SaleEntry" />

<!-- getItemID expects a String -->
<jsp:setProperty
    name="entry"
    property="itemID"
    value='<%= request.getParameter("itemID") %>' />
```

Using Beans

www.coreservlets.com

Setting Bean Properties Case 1: Explicit Conversion & Assignment

```
<%
int numItemsOrdered = 1;
try {
    numItemsOrdered =
        Integer.parseInt(request.getParameter("numItems"));
} catch(NumberFormatException nfe) {}
%>
<!-- getNumItems expects an int -->
<jsp:setProperty
    name="entry"
    property="numItems"
    value="<%= numItemsOrdered %>" />
```

Using Beans

www.coreservlets.com

Setting Bean Properties Case 1: Explicit Conversion & Assignment

```
<%
double discountCode = 1.0;
try {
    String discountString =
        request.getParameter("discountCode");
    discountCode =
        Double.valueOf(discountString).doubleValue();
} catch(NumberFormatException nfe) {}
%>
<!-- getDiscountCode expects a double -->
<jsp:setProperty
    name="entry"
    property="discountCode"
    value="<%= discountCode %>" />
```

Using Beans

www.coreservlets.com

Setting Bean Properties Case 1: Explicit Conversion & Assignment

Item ID	Unit Price	Number Ordered	Total Price
a1234	\$12.34	11	\$135.74

Using Beans

www.coreservlets.com

Case 2: Associating Individual Properties with Input Parameters

- Use the `param` attribute of `jsp:setProperty` to indicate that
 - Value should come from specified request parameter
 - Simple automatic type conversion should be performed for properties that expect values of type boolean, Boolean, byte, Byte, char, Character, double, Double, int, Integer, float, Float, long, or Long.
- **Warning**
 - Two old but formerly popular servers (JSDK and Java Web Server) have bug that causes them to fail for automatic type conversions to double values

Using Beans

www.coreservlets.com

Case 2: Associating Individual Properties with Input Parameters

```

<jsp:useBean id="entry"
             class="coreservlets.SaleEntry" />
<jsp:setProperty
  name="entry"
  property="itemID"
  param="itemID" />
<jsp:setProperty
  name="entry"
  property="numItems"
  param="numItems" />
<jsp:setProperty
  name="entry"
  property="discountCode"
  param="discountCode" />
    
```

The value for the property will come from the corresponding HTTP parameter from the GET or POST request

Using Beans

www.coreservlets.com

Case 3: Associating All Properties with Input Parameters

- Use "*" for the value of the `property` attribute of `jsp:setProperty` to indicate that
 - Value should come from request parameter whose name matches property name
 - Simple automatic type conversion should be performed

Using Beans

www.coreservlets.com

Case 3: Associating All Properties with Input Parameters

```

<jsp:useBean id="entry"
             class="coreservlets.SaleEntry" />
<jsp:setProperty name="entry" property="*" />
    
```

Using Beans

www.coreservlets.com

Sharing Beans

- You can use `scope` attribute to specify additional places where bean is stored
 - Still also bound to local variable in `_jspService`
 - `<jsp:useBean id="..." class="..." scope="..." />`
- Lets multiple servlets or JSP pages share data
- Also permits conditional bean creation
 - Create new object only if you can't find existing one

Using Beans

www.coreservlets.com

Values of the scope Attribute

- **page**
 - Default value. Bean object should be placed in the PageContext object for the duration of the current request. Lets methods in same servlet access bean
- **application**
 - Bean will be stored in ServletContext (available through the application variable or by call to getServletContext()). ServletContext is shared by all servlets in the same Web application (or all servlets on server if no explicit Web applications are defined).

Using Beans

www.coreservlets.com

Values of the scope Attribute

- **session**
 - Bean will be stored in the HttpSession object associated with the current request, where it can be accessed from regular servlet code with getAttribute and setAttribute, as with normal session objects.
- **request**
 - Bean object should be placed in the ServletRequest object for the duration of the current request, where it is available by means of getAttribute

Using Beans

www.coreservlets.com

Conditional Bean Operations

- **Bean conditionally created**
 - `<jsp:useBean>` results in new bean being instantiated only if no bean with same id and scope can be found.
 - If a bean with same id and scope is found, the preexisting bean is simply bound to variable referenced by id.
- **Bean properties conditionally set**
 - `<jsp:useBean ... />` replaced by `<jsp:useBean ...>statements</jsp:useBean>`
 - The statements (`<jsp:setProperty>` elements) are executed *only* if a new bean is created, not if an existing bean is found.

Using Beans

www.coreservlets.com

Conditional Bean Creation: AccessCountBean

```
public class AccessCountBean {
    private String firstPage;
    private int accessCount = 1;

    public String getFirstPage() {
        return(firstPage);
    }

    public void setFirstPage(String firstPage) {
        this.firstPage = firstPage;
    }

    public int getAccessCount() {
        return(accessCount);
    }

    public void setAccessCountIncrement(int increment) {
        accessCount = accessCount + increment;
    }
}
```

Using Beans

www.coreservlets.com

Conditional Bean Creation: SharedCounts1.jsp

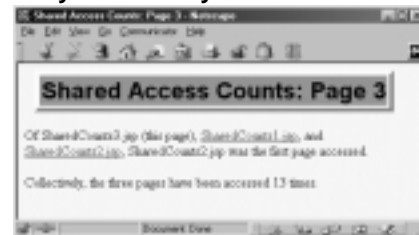
```
<jsp:useBean id="counter"
             class="coreservlets.AccessCountBean"
             scope="application">
    <jsp:setProperty name="counter"
                   property="firstPage"
                   value="SharedCounts1.jsp" />
</jsp:useBean>
Of SharedCounts1.jsp (this page),
<A HREF="SharedCounts2.jsp">SharedCounts2.jsp</A>, and
<A HREF="SharedCounts3.jsp">SharedCounts3.jsp</A>,
<jsp:getProperty name="counter" property="firstPage" />
was the first page accessed.
<P>
Collectively, the three pages have been accessed
<jsp:getProperty name="counter" property="accessCount" />
times.
<jsp:setProperty name="counter"
                 property="accessCountIncrement"
                 value="1" />
```

Using Beans

www.coreservlets.com

Accessing SharedCounts1, SharedCounts2, SharedCounts3

- SharedCounts2.jsp was accessed first.
- Pages have been accessed twelve previous times by an arbitrary number of clients



Using Beans

www.coreservlets.com

Summary

- **Benefits of `jsp:useBean`**
 - Hides the Java syntax
 - Makes it easier to associate request parameters with Java objects (bean properties)
 - Simplifies sharing objects among multiple requests or servlets/JSPs
- **`jsp:useBean`**
 - Creates or accesses a bean
- **`jsp:getProperty`**
 - Puts bean property (i.e. `getXxx` call) into servlet output
- **`jsp:setProperty`**
 - Sets bean property (i.e. passes value to `setXxx`)

Using Beans

www.coreservlets.com



Questions?

Core Servlets & JSP book: www.coreservlets.com
More Servlets & JSP book: www.moreservlets.com
Servlet and JSP Training Courses: courses.coreservlets.com

Slides © Marty Hall, <http://www.coreservlets.com>, book © Sun Microsystems Press