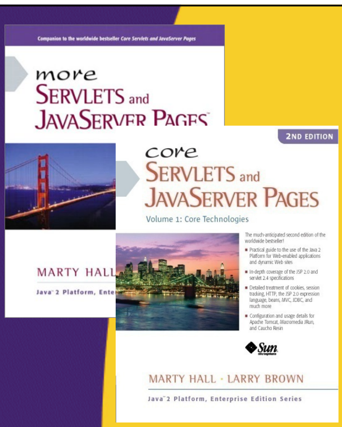
Using JavaBeans in JSP

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/csajsp2.html


more SERVLETS and JAVASERVER PAGES
MARTY HALL
core SERVLETS and JAVASERVER PAGES
2ND EDITION
Volume 1: Core Technologies
MARTY HALL · LARRY BROWN
Java 2 Platform, Enterprise Edition Series

Agenda

- **Understanding the benefits of beans**
  - We will use standalone beans here. Later sections will cover beans with MVC and the JSP expression language.
- **Creating beans**
- **Installing bean classes on your server**
- **Accessing bean properties**
- **Explicitly setting bean properties**
- **Automatically setting bean properties from request parameters**
- **Sharing beans among multiple servlets and JSP pages**

Uses of JSP Constructs

Simple Application

$\big\downarrow$

Complex Application

- **Scripting elements calling servlet code directly**
- **Scripting elements calling servlet code indirectly (by means of utility classes)**
- **Beans**
- **Servlet/JSP combo (MVC)**
- **MVC with JSP expression language**
- **Custom tags**
- **MVC with beans, custom tags, and a framework like Struts or JSF**

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)
    - You should already follow this practice and use accessor methods instead of allowing direct access to fields
  - Persistent values should be accessed through methods called get*Xxx* and set*Xxx*
    - If class has method getTitle that returns a String, class is said to have a String *property* named title
    - Boolean properties may use is*Xxx* instead of get*Xxx*
    - It is the name of the method, not instance var that matters!
  - For more on beans, see http://java.sun.com/beans/docs/

More on Bean Properties

- **Usual rule to turn method name into property name**
  - Drop the word "get" or "set" and change the next letter to lowercase. Again, instance var name is irrelevant.
    - Method name: getUserFirstName
    - Property name: userFirstName
- **Exception 1: boolean properties**
  - If getter returns boolean or Boolean
    - Method name: getPrime <u>or</u> isPrime
    - Property name: prime
- **Exception 2: consecutive uppercase letters**
  - If two uppercase letters in a row after "get" or "set"
    - Method name: getURL
    - Property name: URL (not uRL)

Bean Properties: Examples

Method Names	Property Name	Example JSP Usage
getFirstName setFirstName	firstName	<jsp:getProperty … property="firstName"/> <jsp:setProperty … property="firstName"/> ${customer.firstName}
isExecutive setExecutive (boolean property)	executive	<jsp:getProperty … property="executive"/> <jsp:setProperty … property="executive"/> ${customer.executive}
getExecutive setExecutive (boolean property)	executive	<jsp:getProperty … property="executive"/> <jsp:setProperty … property="executive"/> ${customer.executive}
getZIP setZIP	ZIP	<jsp:getProperty … property="ZIP"/> <jsp:setProperty … property="ZIP"/> ${address.ZIP}

Note 1: property name does not exist anywhere in your code. It is just a shortcut for the method name.
Note 2: property name is derived only from method name. Instance variable name is irrelevant.

Why You Should Use Accessors, Not Public Fields

- **Bean rules**
  - To be a bean, you should not have public fields.
  - Wrong

```
public double speed;
```

  - Right

```
private double speed;  // Var need not match method name

public double getSpeed() {
    return(speed);
}
public void setSpeed(double speed) {
    this.speed = speed;
}
```

Note: in Eclipse, after you create instance variable, if you R-click and choose "Source", it gives you option to generate getters and setters for you.

- **OOP design**
  - You should do this in all your Java code anyhow. Why?

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.

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) {
  speedInKPH = convert(newSpeed);
}

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

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.

Bottom Line

- **It is no onerous requirement to be a “bean”**
  – You are probably following most of the conventions already anyhow
    - Zero arg constructor (not required in MVC!)
    - No public instance variables
    - Use getBlah/setBlah or isBlah/setBlah naming conventions
- **JSP provides many places where you refer to “bean properties”**
  – Which are shortcuts for getter/setter methods
    - getFirstName method: refer to “firstName”
    - isMyThingCool method (boolean): refer to “myThingCool”
    - getHTMLString method: refer to “HTMLString”

Using Beans: Basic Tasks

- **jsp:useBean**
  - In the simplest case, this element builds a new bean. It is normally used as follows:
    - <jsp:useBean id="*beanName*" class="*package.Class*" />
- **jsp:setProperty**
  - This element modifies a bean property (i.e., calls a set*Blah* method). It is normally used as follows:
    - <jsp:setProperty name="*beanName*"
      property="*propertyName*"
      value="*propertyValue*" />
- **jsp:getProperty**
  - This element reads and outputs the value of a bean property. It is used as follows:
    - <jsp:getProperty name="*beanName*"
      property="*propertyName*" />

General Approach with Standalone Pages and jsp:useBean Tags

- **Input form**
  - User submits form that refers to a JSP page
    - <FORM ACTION="SomePage.jsp">
- **JSP Page**
  - JSP page instantiates a bean
    - <jsp:useBean id="myBean" class="…"/>
  - You pass some request data to the bean
    - <jsp:setProperty name="myBean" …/>
      - There are several ways to use jsp:setProperty, but the best is with property="*", as we will see shortly.
  - You output some value(s) derived from the request data
    - <jsp:getProperty name="myBean"
      property="bankAccountBalance"/>

Building Beans: jsp:useBean

- **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:useBean id="book1" class="coreservlets.Book" />
    can be thought of as equivalent to the scriptlet
    <% coreservlets.Book book1 = new coreservlets.Book(); %>
  - But jsp: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

Setting Simple Bean Properties: jsp:setProperty

- **Format**

  -
    ```
    <jsp:setProperty name="name"
                     property="property"
                     value="value" />
    ```
- **Purpose**
  - Allow setting of bean properties (i.e., calls to set*Xxx* 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"); %>

Accessing Bean Properties: jsp:getProperty

- **Format**
  - <jsp:getProperty name="*name*" property="*property*" />
- **Purpose**
  - Allow access to bean properties (i.e., calls to get*Xxx* methods) without explicit Java programming
- **Notes**
  - <jsp:getProperty name="book1" property="title" />

    is equivalent to the following JSP expression

    <%= book1.getTitle() %>

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;
  }
}
```

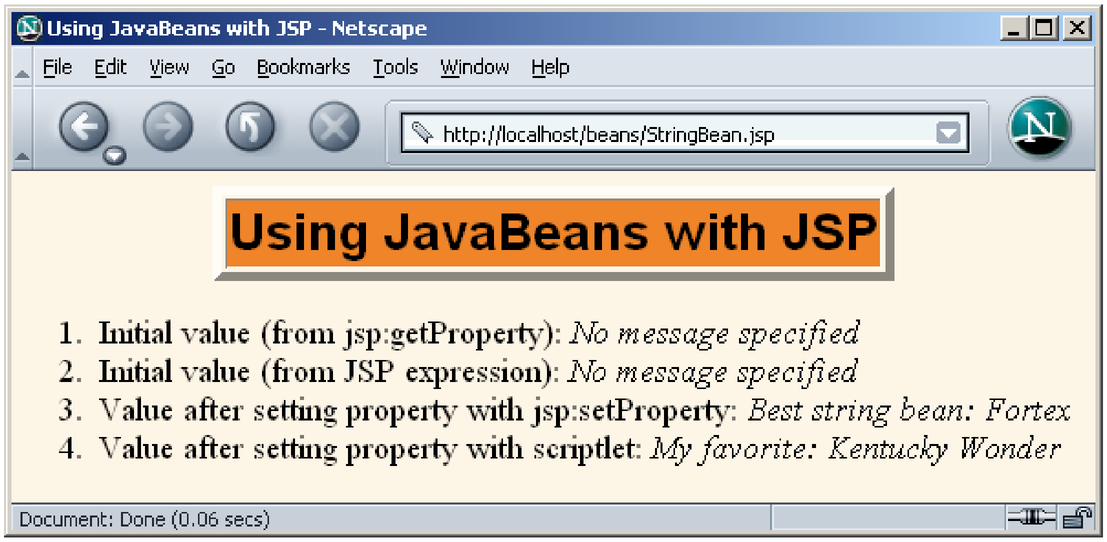
- **Beans installed in normal Java directory**
  - MyEclipse: src/*folderMatchingPackage*
  - Deployment: WEB-INF/classes/*folderMatchingPackage*
- **Beans must *always* be in packages!**

JSP Page That Uses StringBean (Code)

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

Don't really mix scripting and jsp:useBean in same page! I am just illustrating that behind the scenes, the jsp: tags are just calling Java code.

JSP Page That Uses StringBean (Result)


Using JavaBeans with JSP - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/beans/StringBean.jsp

Using JavaBeans with JSP

1. Initial value (from jsp:getProperty): *No message specified*
2. Initial value (from JSP expression): *No message specified*
3. Value after setting property with jsp:setProperty: *Best string bean: Fortex*
4. Value after setting property with scriptlet: *My favorite: Kentucky Wonder*

Document: Done (0.06 secs)

Setting Bean Properties Case 1: Explicit Conversion & Assignment

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

<%-- setItemID expects a String --%>

<jsp:setProperty
    name="entry"
    property="itemID"
    value='<%= request.getParameter("itemID") %>' />
```

Bad: violates rule of not having scripting in pages that use jsp:useBean, jsp:getProperty, and jsp:setProperty.

Setting Bean Properties Case 1: Explicit Conversion & Assignment

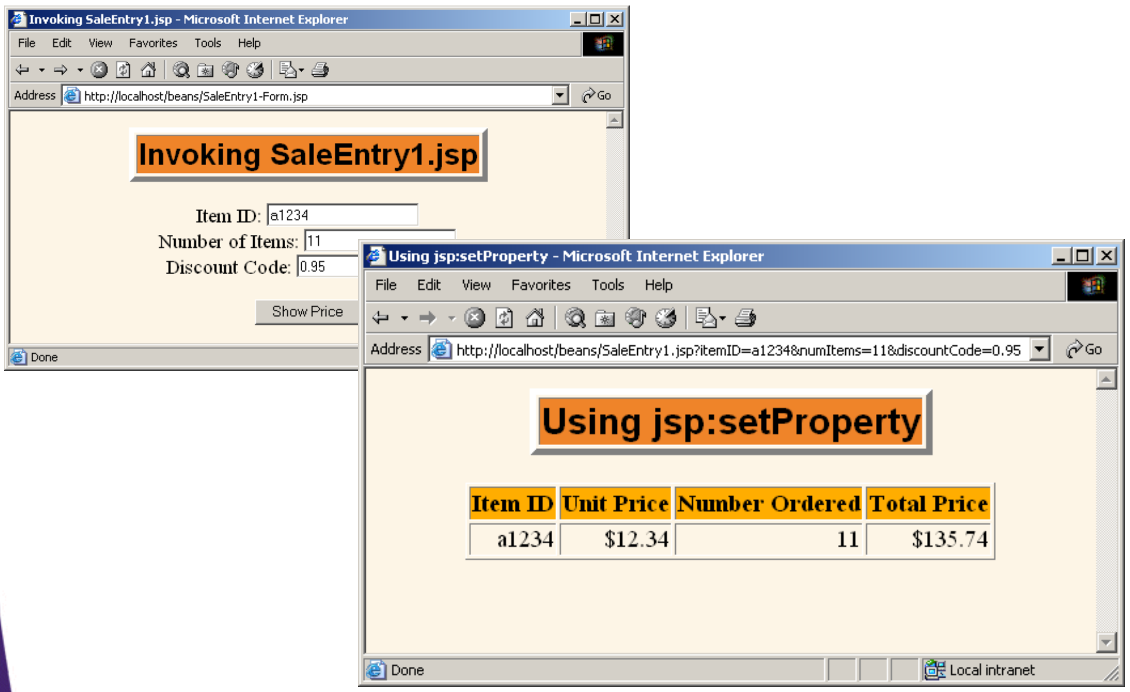
```
<%
int numItemsOrdered = 1;
try {
  numItemsOrdered =
    Integer.parseInt(request.getParameter("numItems"));
} catch(NumberFormatException nfe) {}
%>

<%-- setNumItems expects an int --%>

<jsp:setProperty
    name="entry"
    property="numItems"
    value="<%= numItemsOrdered %>" />
```

Even worse than previous slide: not only has scripting, but has long ugly sections of Java code.

Setting Bean Properties Case 1: Explicit Conversion & Assignment

```
<%
double discountCode = 1.0;
try {
  String discountString =
    request.getParameter("discountCode");
  discountCode =
    Double.parseDouble(discountString);
} catch(NumberFormatException nfe) {}
%>

<%-- setDiscountCode expects a double --%>

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

Setting Bean Properties Case 1: Explicit Conversion & Assignment


Invoking SaleEntry1.jsp - Microsoft Internet Explorer

Address http://localhost/beans/SaleEntry1-Form.jsp

Invoking SaleEntry1.jsp

Item ID: a1234
Number of Items: 11
Discount Code: 0.95

Using jsp:setProperty - Microsoft Internet Explorer

Address http://localhost/beans/SaleEntry1.jsp?itemID=a1234&numItems=11&discountCode=0.95

Using jsp:setProperty

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

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 standard types
    - boolean, Boolean, byte, Byte, char, Character, double, Double, int, Integer, float, Float, long, or Long.

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" />
```

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

Case 3: Associating All Properties with Input Parameters

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

- **This is extremely convenient for making “form beans” -- objects whose properties are filled in from a form submission.**
  - You can even divide the process up across multiple forms, where each submission fills in part of the object.

Sharing Beans

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

Sharing Beans: Example

- **page1.jsp**

```
<jsp:useBean id="foo" class="…" scope="application"/>
<jsp:setProperty name="foo" property="message"
                 value="Hello"/>
<jsp:getProperty name="foo" property="message"/>
```

- **page2.jsp**

```
<jsp:useBean id="foo" class="…" scope="application"/>
<jsp:getProperty name="foo" property="message"/>
```

- **Possible scenario 1**
  - Joe goes to page 2 (output is "Default Message")
  - Jane goes to page 1 (output is "Hello")
- **Possible scenario 2**
  - Joe goes to page 1 (output is "Hello")
  - Jane goes to page 2 (output is "Hello")

Values of the scope Attribute

- **page (<jsp:useBean … scope="page"/> or <jsp:useBean…>)**
  - 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 (<jsp:useBean … scope="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).

Values of the scope Attribute

- **session (<jsp:useBean … scope="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 (<jsp:useBean … scope="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

Sharing Beans in Four Different Ways

- **Using unshared (page-scoped) beans.**
- **Sharing request-scoped beans.**
- **Sharing session-scoped beans.**
- **Sharing application-scoped (i.e., ServletContext-scoped) beans.**

- **Important:**
  - Use different names (i.e., id in jsp:useBean) for different beans
    - Don't store beans in different places with same id

Sharing Beans Four Ways: Bean Code

```
package coreservlets;
…
public class BakedBean implements Serializable {
  private String level = "half-baked";
  private String goesWith = "hot dogs";

  public String getLevel() {
    return(level);
  }
  public void setLevel(String newLevel) {
    level = newLevel;
  }
  public String getGoesWith() {
    return(goesWith);
  }
  public void setGoesWith(String dish) {
    goesWith = dish;
  }
}
```

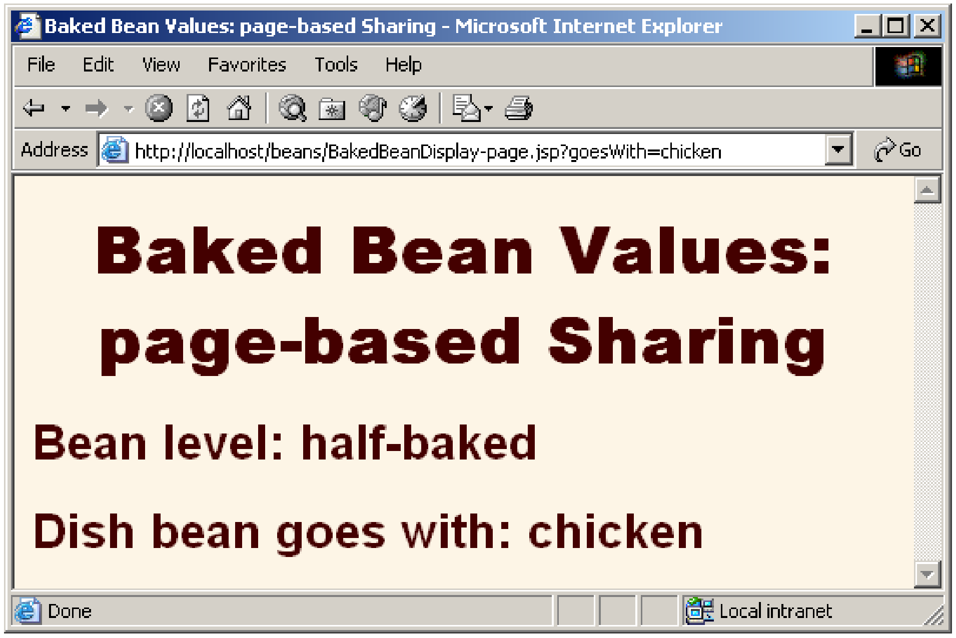
Sharing Beans Example 1: Page-Scoped (Unshared)

- **Create the bean**
  - Use jsp:useBean with scope="page" (or no scope at all, since page is the default).
- **Modify the bean**
  - Use jsp:setProperty with property="*".
  - Then, supply request parameters that match the bean property names.
- **Access the bean**
  - Use jsp:getProperty.

Sharing Beans Example 1: Page-Scoped (Unshared)

```
…
<BODY>
<H1>Baked Bean Values: page-based Sharing</H1>
<jsp:useBean id="pageBean"
             class="coreservlets.BakedBean" />
<jsp:setProperty name="pageBean" property="*" />
<H2>Bean level:
<jsp:getProperty name="pageBean"
                 property="level" />
</H2>
<H2>Dish bean goes with:
<jsp:getProperty name="pageBean"
                 property="goesWith" />
</H2>
</BODY></HTML>
```

Sharing Beans Example 1: Result (Initial Request)

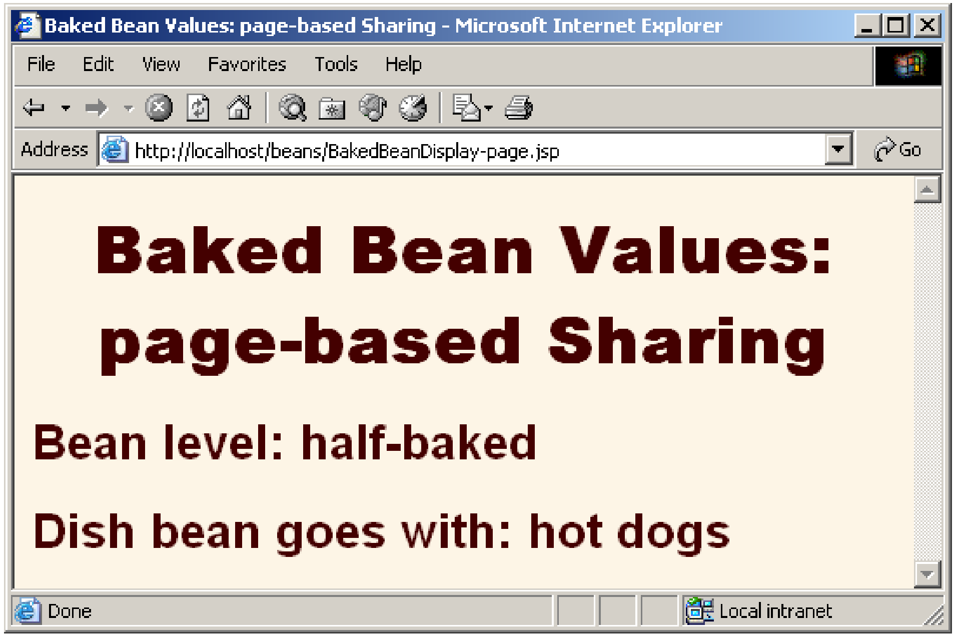

Baked Bean Values: page-based Sharing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
http://localhost/beans/BakedBeanDisplay-page.jsp?goesWith=chicken
Go
Done
Local intranet

Baked Bean Values: page-based Sharing

Bean level: half-baked

Dish bean goes with: chicken

Sharing Beans Example 1: Result (Later Request)


Baked Bean Values: page-based Sharing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
http://localhost/beans/BakedBeanDisplay-page.jsp
Go
Done
Local intranet

Baked Bean Values: page-based Sharing

Bean level: half-baked

Dish bean goes with: hot dogs

Sharing Beans Example 2: Request-Based Sharing

- **Create the bean**
  - Use jsp:useBean with scope="request".
- **Modify the bean**
  - Use jsp:setProperty with property="*".
  - Then, supply request parameters that match the bean property names.
- **Access the bean in the 1st (main) page**
  - Use jsp:getProperty.
  - Then, use jsp:include to invoke the second page.
- **Access the bean in the 2nd (included) page**
  - Use jsp:useBean with the same id as on the first page, again with scope="request".
  - Then, use jsp:getProperty.

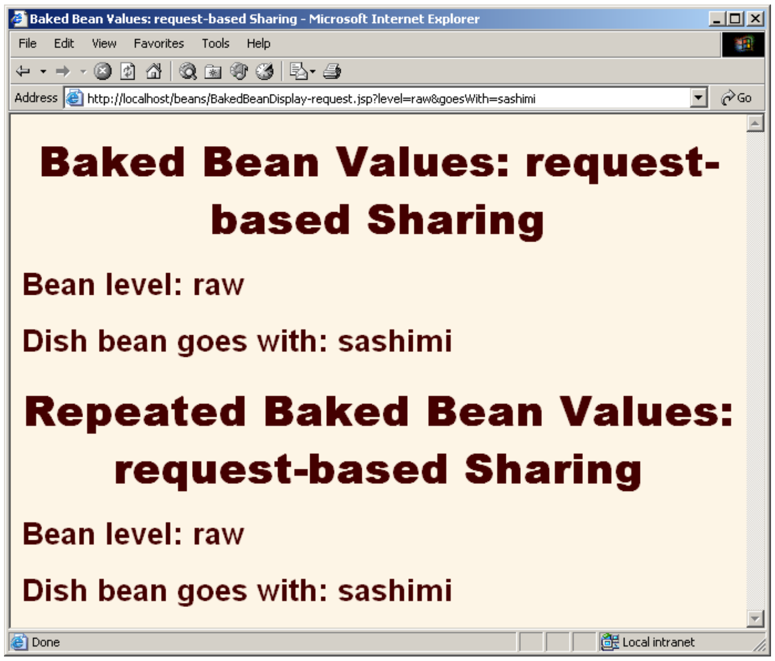
Request-Based Sharing: Code for Main Page

```
…
<BODY>
<H1>Baked Bean Values: request-based Sharing</H1>
<jsp:useBean id="requestBean"
             class="coreservlets.BakedBean"
             scope="request" />
<jsp:setProperty name="requestBean"
                 property="*" />
<H2>Bean level:
<jsp:getProperty name="requestBean"
                 property="level" /></H2>
<H2>Dish bean goes with:
<jsp:getProperty name="requestBean"
                 property="goesWith" /></H2>
<jsp:include page=
  "/WEB-INF/includes/BakedBeanDisplay-snippet.jsp"/>
</BODY></HTML>
```

Request-Based Sharing: Code for Included Page

```
<H1>Repeated Baked Bean Values:
request-based Sharing</H1>
<jsp:useBean id="requestBean"
             class="coreservlets.BakedBean"
             scope="request" />
<H2>Bean level:
<jsp:getProperty name="requestBean"
                 property="level" />
</H2>
<H2>Dish bean goes with:
<jsp:getProperty name="requestBean"
                 property="goesWith" />
</H2>
```

Request-Based Sharing: Result (Initial Request)


Baked Bean Values: request-based Sharing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/beans/BakedBeanDisplay-request.jsp?level=raw&goesWith=sashimi
Go
Done
Local intranet

Baked Bean Values: request-based Sharing

Bean level: raw

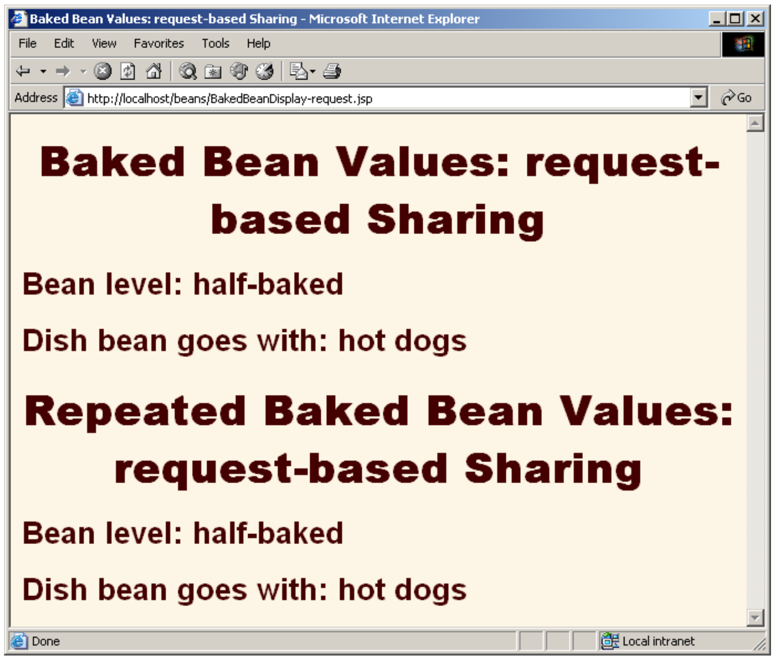
Dish bean goes with: sashimi

Repeated Baked Bean Values: request-based Sharing

Bean level: raw

Dish bean goes with: sashimi

Request-Based Sharing: Result (Later Request)


Baked Bean Values: request-based Sharing - Microsoft Internet Explorer

Address http://localhost/beans/BakedBeanDisplay-request.jsp

Baked Bean Values: request-based Sharing

Bean level: half-baked

Dish bean goes with: hot dogs

Repeated Baked Bean Values: request-based Sharing

Bean level: half-baked

Dish bean goes with: hot dogs

Done

Local intranet

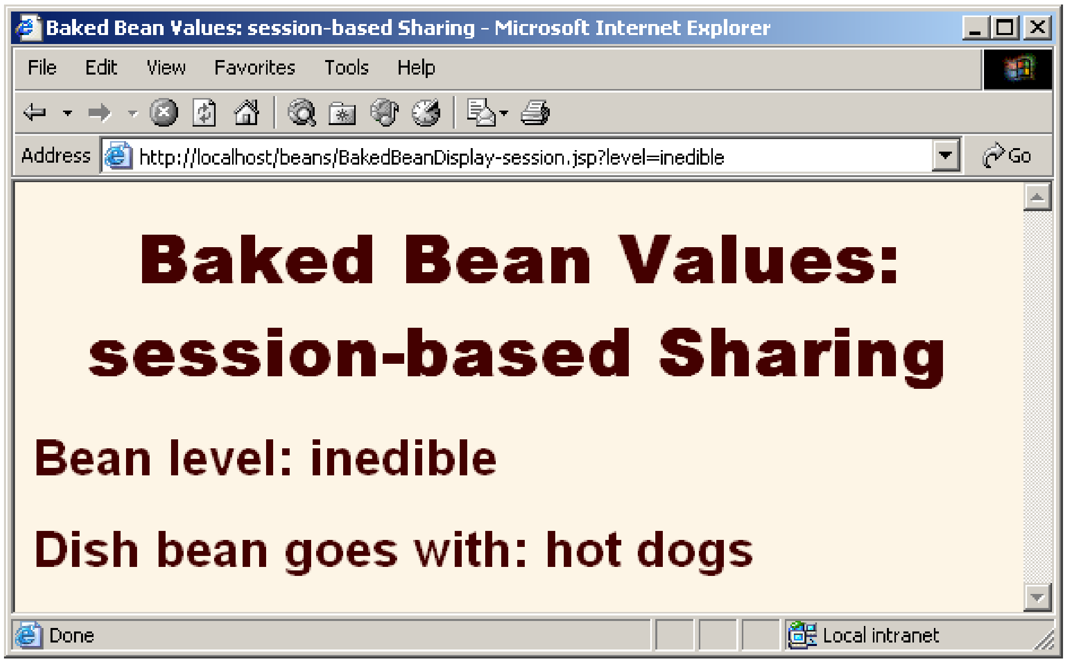
Sharing Beans Example 3: Session-Based Sharing

- **Create the bean**
  - Use jsp:useBean with scope="session".
- **Modify the bean**
  - Use jsp:setProperty with property="*".
  - Then, supply request parameters that match the bean property names.
- **Access the bean in the initial request**
  - Use jsp:getProperty in the request in which jsp:setProperty is invoked.
- **Access the bean later**
  - Use jsp:getProperty in a request that does not include request parameters and thus does not invoke jsp:setProperty. If this request is from the same client (within the session timeout), the previously modified value is seen. If this request is from a different client (or after the session timeout), a newly created bean is seen.

Session-Based Sharing: Code

```
…
<BODY>
<H1>Baked Bean Values: session-based Sharing</H1>
<jsp:useBean id="sessionBean"
             class="coreservlets.BakedBean"
             scope="session" />
<jsp:setProperty name="sessionBean"
                 property="*" />
<H2>Bean level:
<jsp:getProperty name="sessionBean"
                 property="level" />
</H2>
<H2>Dish bean goes with:
<jsp:getProperty name="sessionBean"
                 property="goesWith" />
</H2></BODY></HTML>
```

Session-Based Sharing: Result (Initial Request)

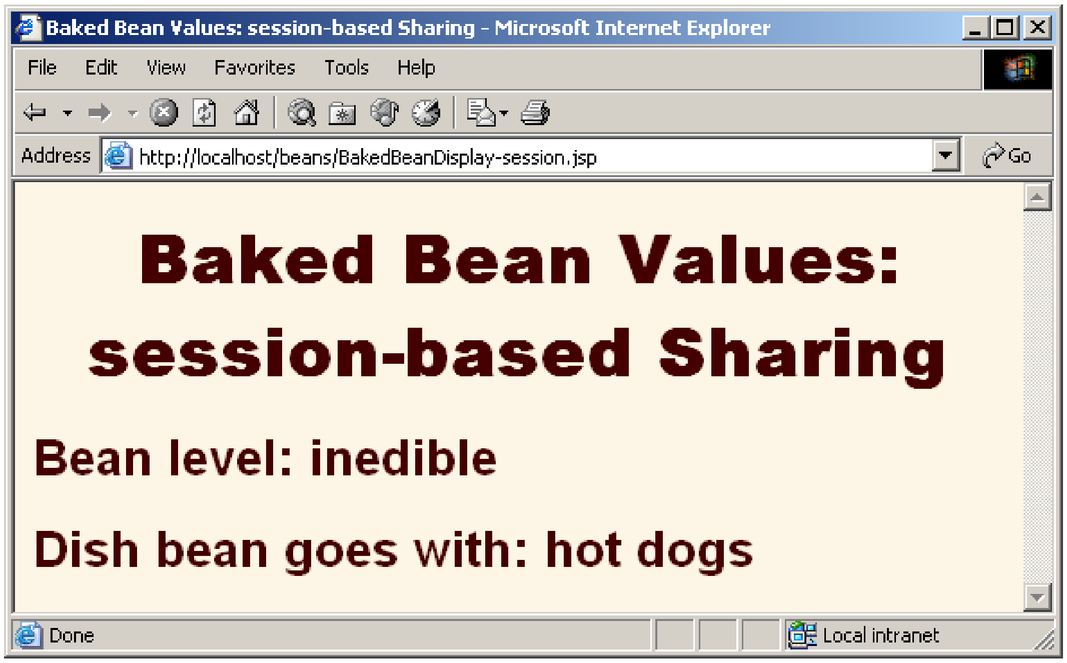

Baked Bean Values: session-based Sharing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/beans/BakedBeanDisplay-session.jsp?level=inedible
Go
Done
Local intranet

Baked Bean Values: session-based Sharing

Bean level: inedible

Dish bean goes with: hot dogs

Session-Based Sharing: Result (Later Request -- Same Client)

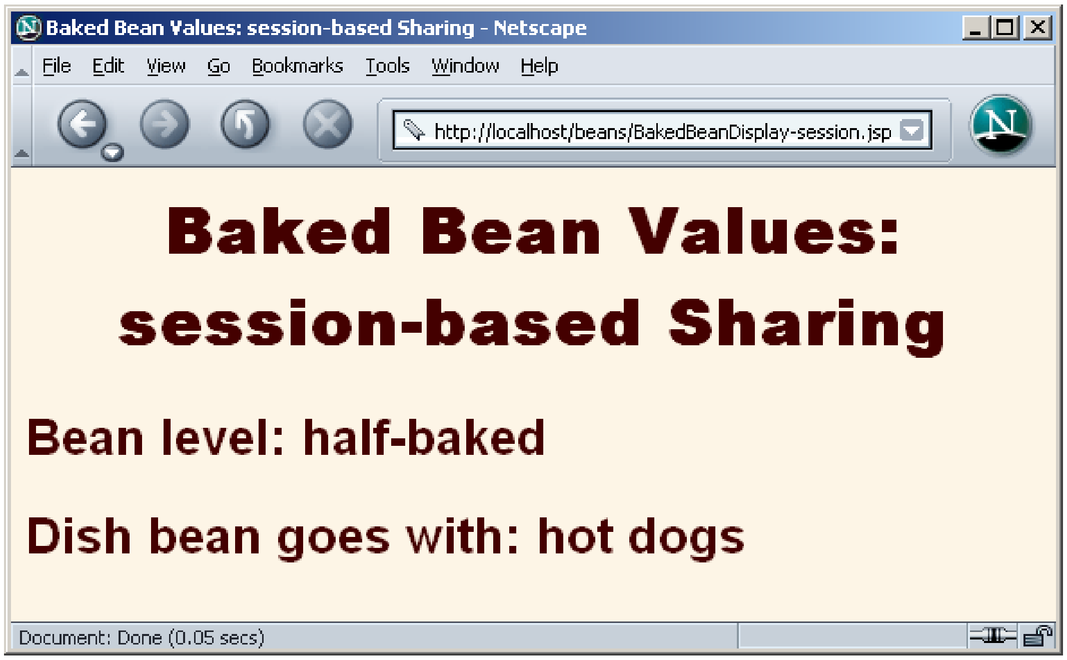

Baked Bean Values: session-based Sharing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/beans/BakedBeanDisplay-session.jsp
Go
Done
Local intranet

Baked Bean Values: session-based Sharing

Bean level: inedible

Dish bean goes with: hot dogs

Session-Based Sharing: Result (Later Request -- New Client)


Baked Bean Values: session-based Sharing - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/beans/BakedBeanDisplay-session.jsp
Document: Done (0.05 secs)

Baked Bean Values: session-based Sharing

Bean level: half-baked

Dish bean goes with: hot dogs

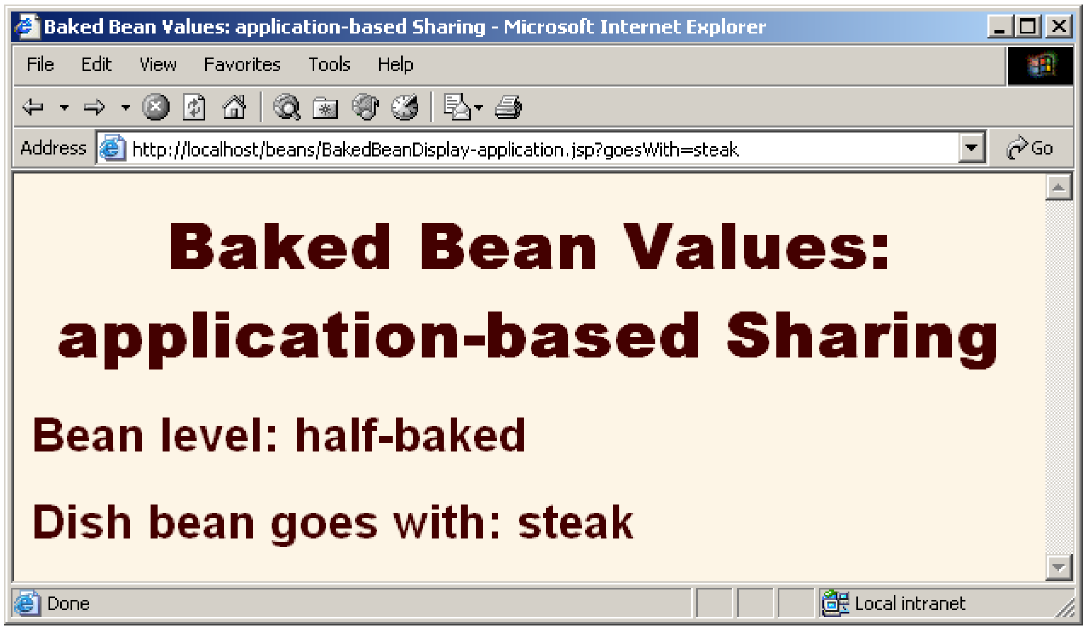
Sharing Beans Example 4: Application-Based Sharing

- **Create the bean**
  - Use jsp:useBean with scope="application".
- **Modify the bean**
  - Use jsp:setProperty with property="*".
  - Then, supply request parameters that match the bean property names.
- **Access the bean in the initial request**
  - Use jsp:getProperty in the request in which jsp:setProperty is invoked.
- **Access the bean later**
  - Use jsp:getProperty in a request that does not include request parameters and thus does not invoke jsp:setProperty. Whether this request is from the same client or a different client (regardless of the session timeout), the previously modified value is seen.

Application-Based Sharing: Code

```
<BODY>
<H1>Baked Bean Values:
application-based Sharing</H1>
<jsp:useBean id="applicationBean"
             class="coreservlets.BakedBean"
             scope="application" />
<jsp:setProperty name="applicationBean"
                 property="*" />
<H2>Bean level:
<jsp:getProperty name="applicationBean"
                 property="level" />
</H2>
<H2>Dish bean goes with:
<jsp:getProperty name="applicationBean"
                 property="goesWith"/>
</H2></BODY></HTML>
```

Application-Based Sharing: Result (Initial Request)


Baked Bean Values: application-based Sharing - Microsoft Internet Explorer

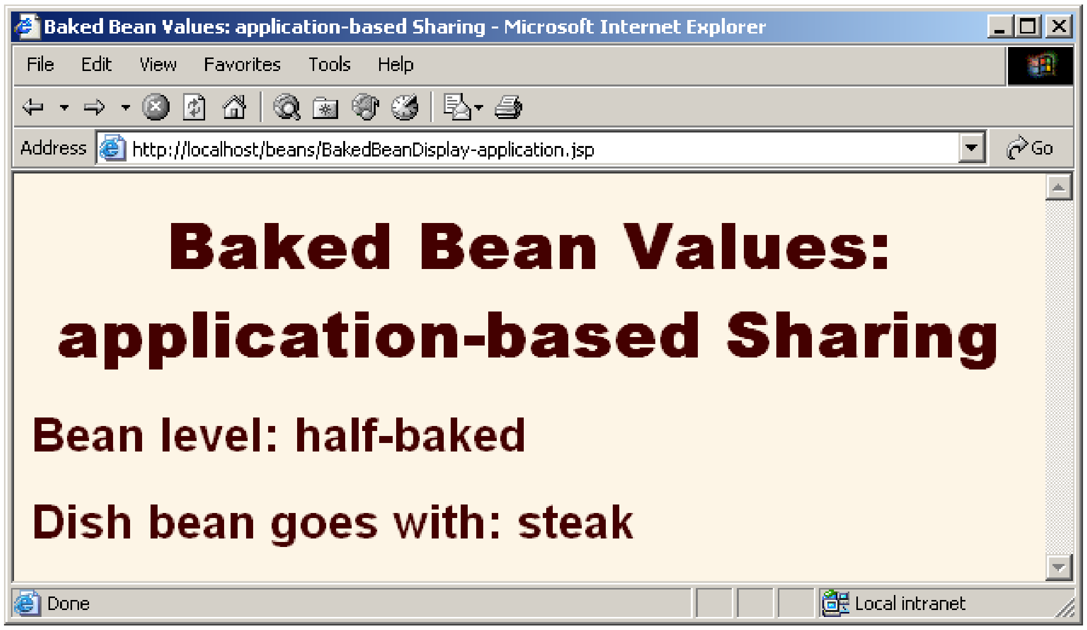
http://localhost/beans/BakedBeanDisplay-application.jsp?goesWith=steak

Baked Bean Values: application-based Sharing

Bean level: half-baked

Dish bean goes with: steak

Application-Based Sharing: Result (Later Request -- Same Client)


Baked Bean Values: application-based Sharing - Microsoft Internet Explorer

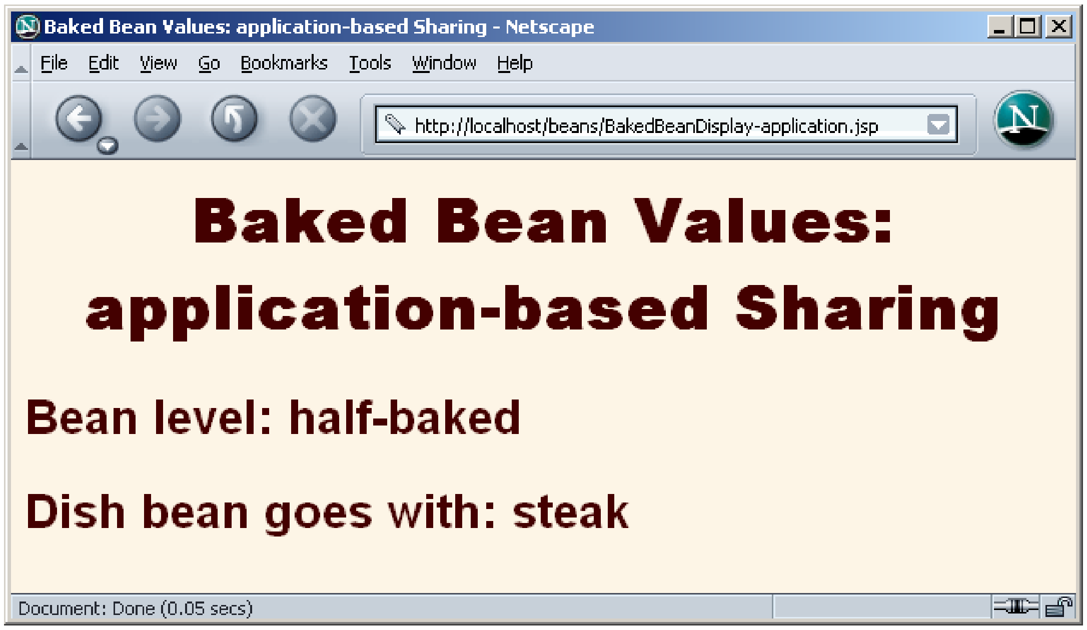
http://localhost/beans/BakedBeanDisplay-application.jsp

Baked Bean Values: application-based Sharing

Bean level: half-baked

Dish bean goes with: steak

Application-Based Sharing: Result (Later Request -- New Client)


Baked Bean Values: application-based Sharing - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/beans/BakedBeanDisplay-application.jsp
Document: Done (0.05 secs)

Baked Bean Values: application-based Sharing

Bean level: half-baked

Dish bean goes with: steak

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.

Conditional Bean Creation: AccessCountBean

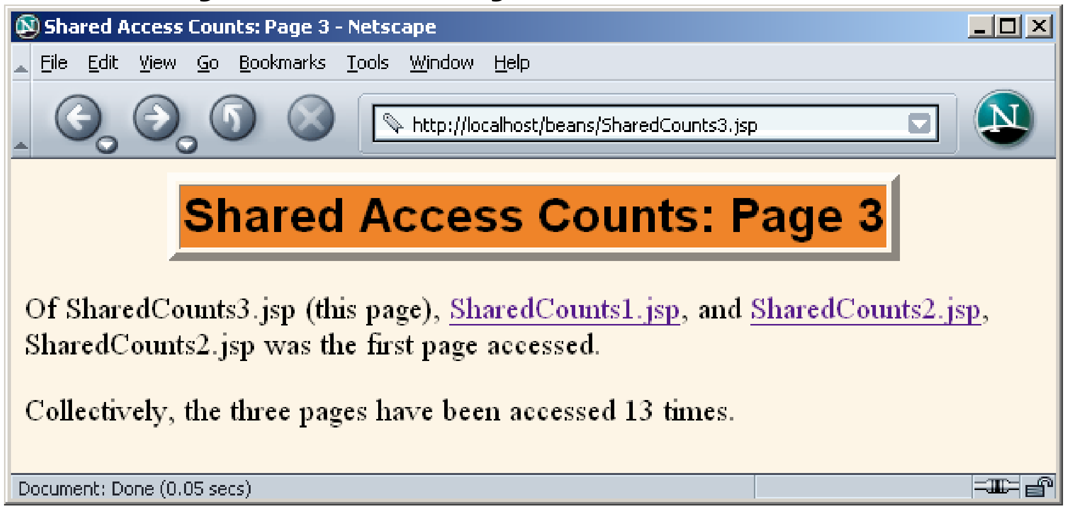
```
public class AccessCountBean {
  private String firstPage;
  private int accessCount = 1;

  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;
  }
}
```

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" />
```

Accessing SharedCounts1, SharedCounts2, SharedCounts3

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


Shared Access Counts: Page 3 - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/beans/SharedCounts3.jsp
Document: Done (0.05 secs)

Shared Access Counts: Page 3

Of SharedCounts3.jsp (this page), SharedCounts1.jsp, and SharedCounts2.jsp, SharedCounts2.jsp was the first page accessed.

Collectively, the three pages have been accessed 13 times.

Summary

- Benefits of jsp:useBean
  – Hides the Java syntax. *No* scripting in these pages!
  – 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:setProperty
  – Sets bean property (i.e. passes value to set*Xxx*)
    - You usually use property="*" to pass in request params
- jsp:getProperty
  – Puts bean property (i.e. get*Xxx* call) into servlet output

Questions?