Lecture 13
Using Java Beans in JSP

Lecture Agenda

1. Understanding the benefits of beans
2. Creating Beans
3. Installing bean classes on your server
4. Accessing bean properties
5. Explicitly setting bean properties
6. Automatically setting bean properties from request parameters
7. Sharing beans among multiple servlets and JSP pages

JSP and Java Beans

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
- 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 Java Beans?

Introduction

- Java Beans are Java classes that follow certain conventions:
  1. Must have a zero-argument constructor
     - You can satisfy the zero-argument constructor requirement by either declaring one explicitly or omitting the declaration of any constructors altogether.
  2. Should have no public instance variables(fields)
     - You should already be following this practice (access methods etc…)
  3. Persistent values should be accessed through methods name **get**Xxx and **set**Xxx
     - Example: If a class has a method called getTitle() that returns String, the class is said to have (ie: expected) a String property member called title (ex: private String title)
     - Boolean properties may use **is**Xxx instead of **get**Xxx
     - It is the name of the method, not its instance variable, that matters for Java Beans compliance.

Background: What are Java Beans?

Introduction

4. Usual rule to turn method name into property name.
   - Drop the “get” or “set”. Change the next letter to lowercase. Instance variable name is irrelevant.

     Method Name: getUserFirstName

     Property Name: userFirstName
5. Exception 1: Boolean Properties

   Method Name: getPrime() or isPrime()

   Property Name: prime

   Jab
6. Exception 2: Consecutive Uppercase Letters
   - If two uppercase letters in a row in property name
   - Method Name: getURL (not getUrl)
   - Property Name: URL (not Url)

Bean Properties: Example

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

Use Accessors not public fields

Why you should use accessor methods instead of public fields

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

public double speed; ← WRONG: declares public member field

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

CORRECT!!
OOP Design Tip: You should be doing this **already** for all your java coding.

Use Accessors not public fields continued …

Why you should use accessor instead of public fields

1. You can put constraints on values with accessors

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

If users had direct access to fields, each user would be responsible for checking constraints (error prone)

Use Accessors not public fields continued …

Why you should use accessor instead of public fields

2. You can change your internal representation without changing the interface.

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

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

Use Accessors not public fields continued …

Why you should use accessor instead of 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. Naturally, this is too much work to forward to users, not to mention poses a huge risk of displaying inconsistent data.

Bottom Line!!

- It is not a difficult requirement to be a bean.
  - You are probably following most of the conventions already

Summary

1. Zero argument constructor
2. No public instance variables
3. Use getBlah/setBlah or isBlah/setBlah naming conventions

jsp:useBean, jsp:setProperty, jsp:getProperty

Using Beans: Basic Tasks

Basics

- **jsp:useBean**
  - In the simplest case, this element builds a **new** bean.
  - It is normally used as follows:

    <jsp:useBean id="*beanName*" **class=**"*package.Classname*"**>**

- **jsp:setProperty**
  - This element modifies a bean property (calls properties setter method)
  - It is normally used as follows:

<jsp:setProperty name="*beanName*" **property=**"*propName*" **value=**"*propValue*"**/>**

- **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

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 can pass request data to the bean

```
<jsp:setProperty name="myBean" … />
```

  - 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.Classname*” />
- **Purpose**
  - Allow instantiation of Java classes without explicit Java programming (XML like syntax)
- **Notes**
  - Simple interpretation

    <jsp:useBean id=“*book1*” **class**=“*ca.study.Book*” />

    Can be thought of as the following scriplet ….

    <% ca.study.Book book1 = new ca.study.Book(); **%>**
  - Jsp:useBean has two advantages
    1. It is easier to derive object values from request parameters
    2. It is easier to share objects among pages or servlets

Building Beans: setProperty

- **Format**
  - **<jsp: setProperty name**=“*name*” **property**=“*property*” **value**=“*value*”/>
- **Purpose**
  - Allow setting of bean properties (ie. Calls to **set**Xxx methods) without explicit Java programming.

Jab

- **Notes**

<jsp:setProperty name=“book1” **property**=“title” **value**=“My COMP3095 Book Title” **/>**

Is equivalent to the following …

<% book1.setTitle(“My COMP 3095 Book Title”); **%>**

Building Beans: getProperty

- **Format**
  - **<jsp:getProperty name**=“*name*” **property**=“*property*” />
- **Purpose**
  - Allows access to bean properties (ie. calls to **get**Xxx methods) without explicit Java programming.
- **Notes**

<jsp:getProperty name=“book1” **property**=“title” />

Is equivalent to the following …

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

Example: StringBean

```
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
  - **Eclipse**: src/folderMatchingPackageName
  - **Deployment**: WEB-INF/classes/folderMatchingPackageName
- Beans **must** always be in packages!!!

Example: StringBean

Instantiate bean

```
<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="first string bean message" />
           Value after setting property with jsp:setProperty:
       <I><jsp:getProperty name="stringBean"  property="message" /></I>
    <LI>

        <% stringBean.setMessage("second string bean message"); %>
        Value after setting property with scriptlet: <I><%=stringBean.getMessage()%></I>
</OL>
```

jsp:getProperty / jsp:setProperty

jsp scriptlet and jsp expressions

StringBean Result

Practical Example:
Sales Entry Form Case 1-
Explicit conversion and Assignment

SalesEntry Form

Sales form

Sales display

Invoking SaleEntry1.jsp - Microsoft Internet Explorer

File Edit View Favorites Tools Help

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

Invoking SaleEntry1.jsp

Item ID: a1234

Number of Items: 11

Discount Code: 0.95

Show Price

Done

Using jsp:setProperty - Microsoft Internet Explorer

File Edit View Favorites Tools Help

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

Go

Using jsp:setProperty

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

Done

Local intranet

Case 1: Setting Bean Properties

Property **ItemID**: 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") %>' />
```

Instantiate bean

Utilizes incoming request to set Bean property

Casse 1: Setting Bean Properties continued …

Property **numItems**: 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 %>" />
```

JSP scriplet to extract numItems

set numItems property

Case 1: Setting Bean Properties continued …

Property **discountCode**: 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 %>" />
```

JSP scriptlet to extract discountCode

set discount code

Practical Example:
Sales Entry Form Case 2 -
Associating **Individual** Properties with Input Parameters

Case 2: Associating Individual Properties

Associating Individual Properties with Input Parameters

- Use the **param** attribute of jsp:setProperty to indicate that …
  - Value should come from <u>specified request parameter</u>
  - Simple automatic type conversion 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 continued …

Associating Individual Properties with Input Parameters

```
<FORM ACTION="SaleEntry2.jsp">
    Item ID: <INPUT TYPE="TEXT" NAME="itemID"><BR> Number of
    Items: <INPUT TYPE="TEXT" NAME="numItems"><BR> Discount
    Code: <INPUT TYPE="TEXT" NAME="discountCode">
    <P>
        <INPUT TYPE="SUBMIT" VALUE="Show Price">
</FORM>
```

SaleEntry2-Form.jsp
(code fragment only)

```
<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" />
```

SaleEntry2.jsp
(code fragment only)

Sharing Beans

Sharing Beans

Bean Scope

- You can use the **scope** attribute to specify additional places where bean is stored
  - <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”>

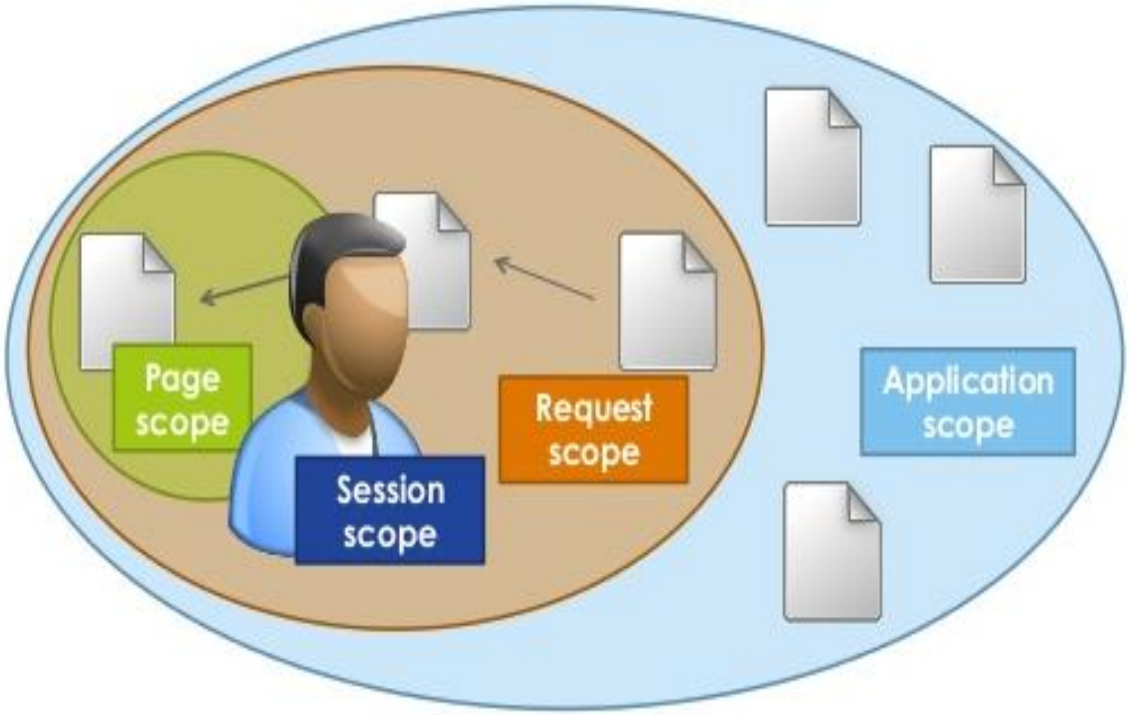
Scope Attribute

page	**<jsp:useBean … scope=**"*page*" />** or simply **<jsp:useBean … >** • **Default** scope value. • Bean object placed in the page context (PageContext) for the duration of the current request only. • Lets methods in the same servlet access a bean.
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.
session	**<jsp:useBean … scope=**"*session*" />** • Bean will be stored in 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.
application	**<jsp:useBean … scope=**"*application*" />** • Bean will be stored in ServletContext (available through the application variable or by making a call to getServletContext()). • ServletContext is shared by **all servlets in the same web application**.

Sharing Beans

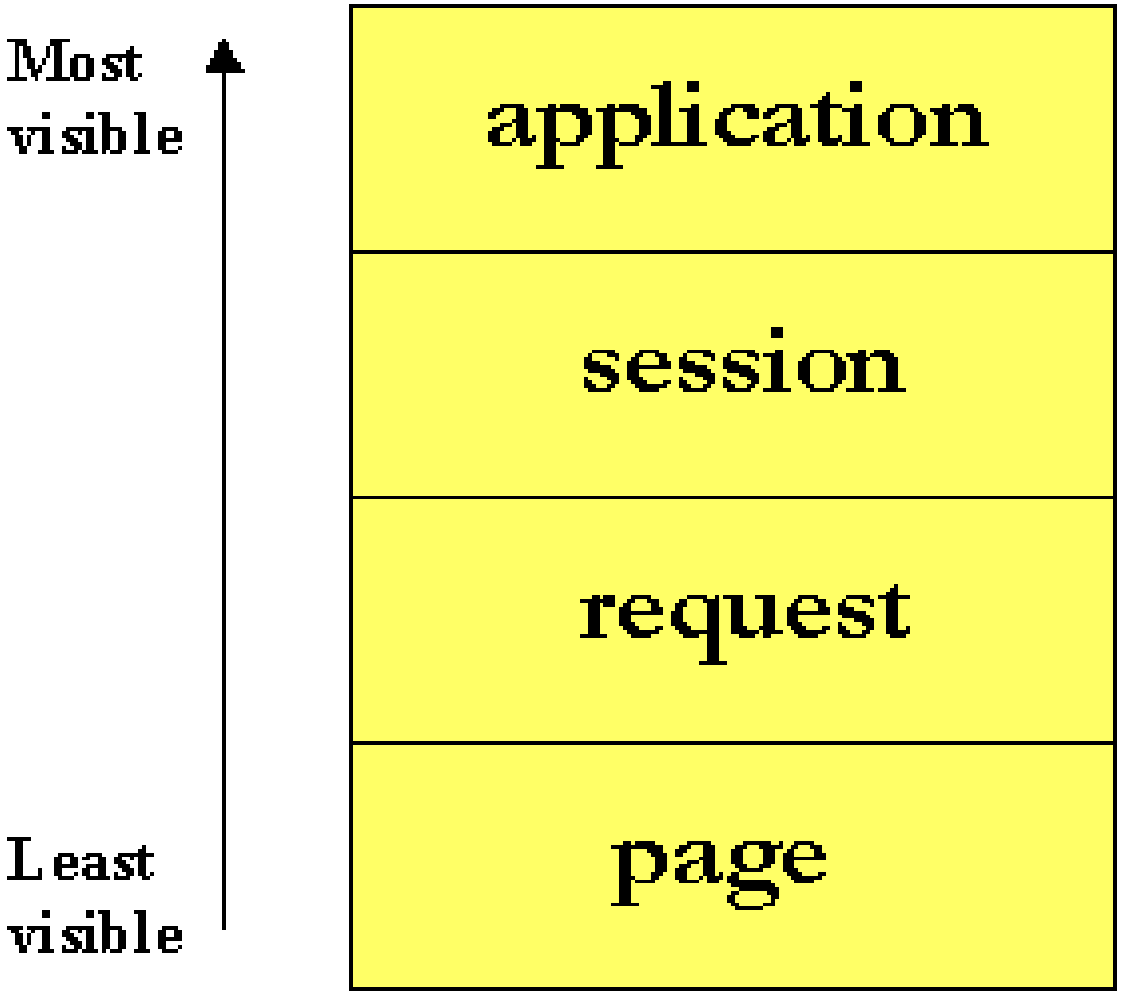
Bean Sharing Scenarios

1. Using unshared (page-scoped) beans.
2. Sharing request-scope beans.
3. Sharing session-scoped beans.
4. Sharing application-scoped beans



Sharing Beans

Bean Sharing Scenarios



Objects accessible from pages that belong to the same application

Objects accessible from pages belonging to the same session as the one in which they were created

Objects accessible from pages processing the request where they were created

Objects accessible only within pages where they were created

Sharing Beans: Page Scoped - Practical Example

Sharing Beans

Sample Bean

```
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;
  }
}
```

Private members only accessed through accessor methods

Note: no constructor means implicit no argument constructor added.

Sharing Bean: Practical Example 1

Page-Scoped Unshared

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

Sharing Bean: Practical Example 1 continued …

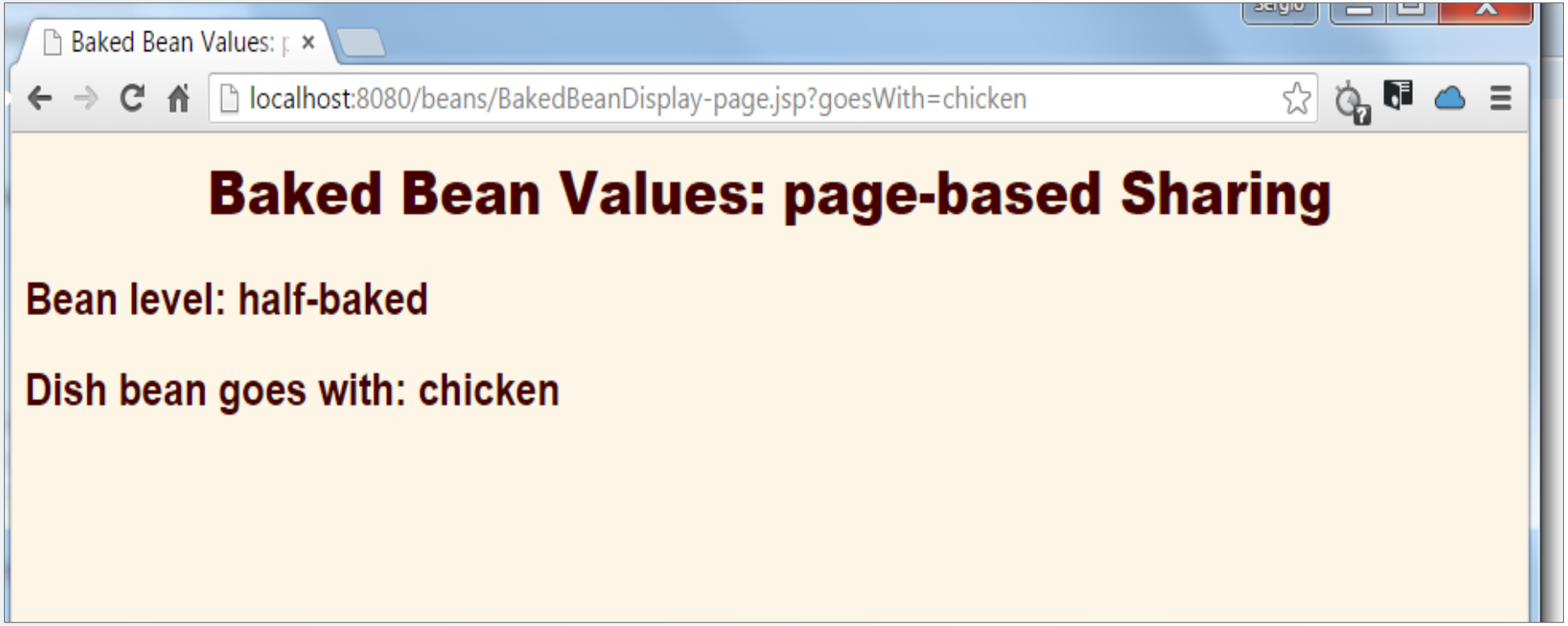
Page-Scoped Unshared

```
<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>
```

Utilizing "*" indicates property values should come from incoming **request** parameters whose name **match** the Bean **property** name

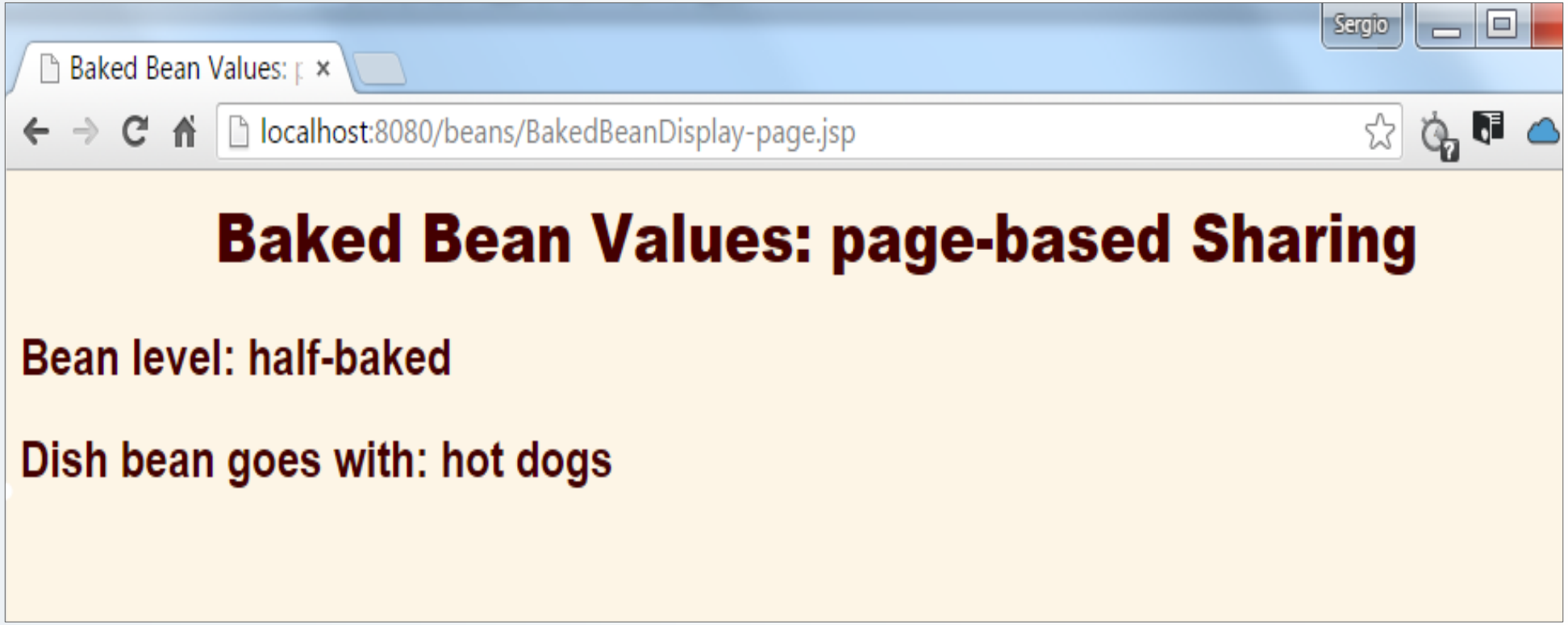
Sharing Bean: Practical Example 1 Result

Page-Scoped Unshared



Sharing Bean: Practical Example 1 Result (Later Request)

Page-Scoped Unshared



Sharing Beans: Request Scoped - Practical Example

Sharing Bean: Practical Example 2

Request-Scoped 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 <u>same id as on the first page</u>, again with the **scope**=“*request*”
  - Then, use **jsp:getProperty**

Sharing Bean: Practical Example 2

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>
```

Main JSP Page

Sharing Bean: Practical Example 2

BakedBeanDisplay-snippet.jsp (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>
```

Included JSP Page

Sharing Bean: Practical Example 2 Result

Request-Scoped

Baked Bean Values: r

localhost:8080/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

Initial Page Request

Passing Parameters

Baked Bean Values: r

localhost:8080/beans/BakedBeanDisplay-request.jsp?level=raw&goesWith=sashimi

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

Sharing Beans: Session Scoped - Practical Example

Sharing Bean: Practical Example 3

Session-Scoped 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 session timeout), the previously modified value is seen.
  - If this request is from a different client (or after session timeout), a newly created bean is seen.

Sharing Bean: Practical Example 3

Session Based Sharing

```
<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>
```

declaring bean in session scope

get Bean properties

Sharing Bean: Practical Example 3 Result

Session-Scoped

Initial request

Baked Bean Values: s ×

localhost:8080/beans/BakedBeanDisplay-session.jsp?level=inedible

Baked Bean Values: session-based Sharing

Bean level: inedible

Dish bean goes with: hot dogs

Later request, same client

Baked Bean Values: s ×

localhost:8080/beans/BakedBeanDisplay-session.jsp

Baked Bean Values: session-based Sharing

Bean level: inedible

Dish bean goes with: hot dogs

Sharing Bean: Practical Example 3 Result

Session-Scoped

Sharing Beans: Application Scoped - Practical Example

Sharing Bean: Practical 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.

Sharing Bean: Practical Example 4

Application Based Sharing

```
<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>
```

Sharing Bean: Practical Example 4 Result

Application-Scoped

Initial request

Baked Bean Values: a

localhost:8080/beans/BakedBeanDisplay-application.jsp?goesWith=steak

Baked Bean Values: application-based Sharing

Bean level: half-baked

Dish bean goes with: steak

Later request, same client

Baked Bean Values: a

localhost:8080/beans/BakedBeanDisplay-application.jsp

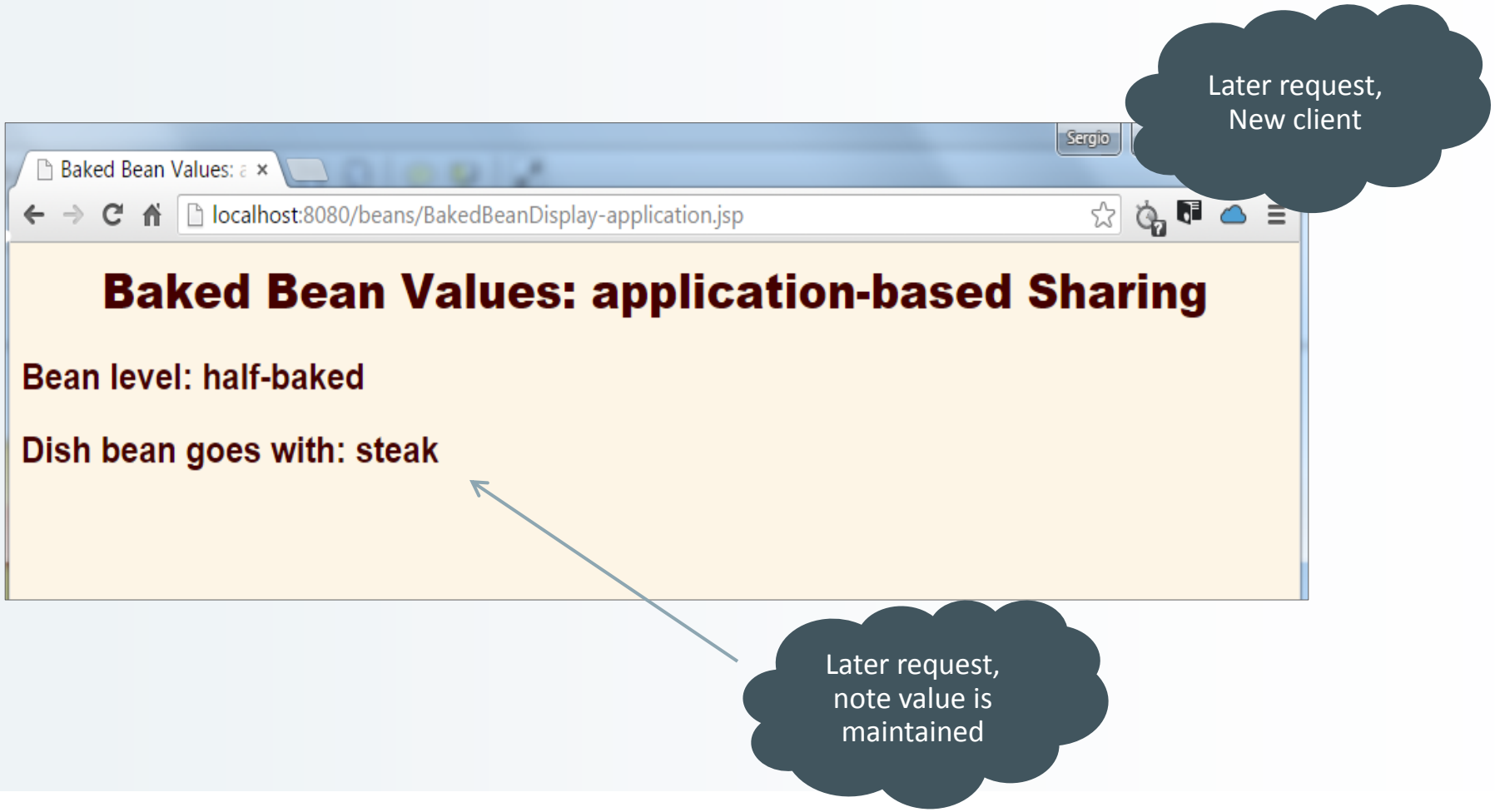
Baked Bean Values: application-based Sharing

Bean level: half-baked

Dish bean goes with: steak

Sharing Bean: Practical Example 4 Result

Application-Scoped



Summary

Lecture 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:setProperty
  - Sets bean property (ie. passes value to **set**Xxx)
  - You usually use **property**=**"*"** to pass in request parameters
- jsp:getProperty
  - Puts bean property (ie: **get**Xxx call) into servlet output

Questions?