JavaBean组件在JSP文档中的应用

议程

- bean的优点
- bean的创建
- bean类在服务器上的安装
- 访问bean属性
- 显式地设置 bean的属性
- 根据请求参数自动设置bean的属性
- 多个servlet和JSP页面间共享bean

不同JSP构造的应用

简单应用

复杂应用

- 脚本元素直接调用servlet代码
- 脚本元素间接调用servlet代码（通过实用工具类）
- bean
- servlet/JSP的组合(MVC)
- MVC连同JSP表达式语言
- 定制标签

背景知识：bean是什么？

- 遵循某些特定约定的Java类
  - 必须具备一个零参数（空）的构造函数
    - 显式地定义这样一个构造函数或者省略所有的构造函数都能满足这项要求
  - 不应该有公开的实例变量（字段）
    - 希望您早就遵循这项准则，使用存取方法而非允许对字段直接访问
  - 持续性的值应该能够通过getXxx和setXxx方法来访问
    - 如果类有getTitle方法，返回String，我们就说类有一个名为title的String属性
    - 布尔型的属性使用is*Xxx*，*而非*get*Xxx*
  - 有关bean的更多内容，参见 http://java.sun.com/beans/docs/

为什么我们应该使用存取方法，不应使用公开字段呢？

- bean不能有公开字段
- 因此，应该将下面的代码

```
public double speed;
```

- 替换为

```
private double speed;

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

- 总之，您的所有Java代码都应该遵循这项准则。为什么呢？

为什么我们应该使用存取方法，不应使用公开字段呢？

- 1) 可以对值加以约束

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

  - 如果类的用户直接访问这些字段，那么每个用户都得负责检查这些约束条件。

为什么我们应该使用存取方法，不应使用公开字段呢？

- 2) 我们可以在不改变接口的情况下修改内部的表达

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

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

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


为什么我们应该使用存取方法，不应使用公开字段呢？

- 3) 我们可以执行任意的边界效应

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

  - 如果类的用户直接访问这些字段，那么每个用户都得独立负责执行边界效应。需要大量的工作，并且极有可能会显示和实际值不一致的结果。

使用bean：基本任务

- **jsp:useBean**
  - 在最简单的情况下，这个元素构建一个新的bean。
    常见的用法如下：

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

- **jsp:getProperty**
  - 这个元素读取并输出bean属性的值。
    用法如下：

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

- **jsp:setProperty**
  - 这个元素修改bean的属性（即调用形如setXxx的方法）。常见的用法如下：

```
<jsp:setProperty name="beanName"
```


构建bean：jsp:useBean

- **格式**
  - <jsp:useBean id="*name*" class="*package.Class*" />
- **目的**
  - 不需用到显式的Java编程就能够创建Java类的实例（XML兼容的语法）
- **注意**
  - 简单来说：
    <jsp:useBean id="book1" class="coreservlets.Book" />
    可以认为等同于下面的scriptlet
    <% coreservlets.Book book1 = new coreservlets.Book(); %>
  - 但jsp:useBean拥有下面两项额外的优势：
    - 从请求参数中导出对象的值更容易
    - 在页面和servlet间共享对象更容易

访问bean的属性：jsp:getProperty

- 格式
  - <jsp:getProperty name="*name*" property="*property*" />
- 目的
  - 不需用到显式的Java编程就可以访问bean的属性（即调用get*Xxx*方法）
- 注意
  - <jsp:getProperty name="book1" property="title" />
    等价于下面的JSP表达式
    <%= book1.getTitle() %>

设置简单的bean属性：jsp:setProperty

- 格式
  - <jsp:setProperty name="*name*"
    property="*property*"
    value="*value*" />
- 目的
  - 不需用到显式的Java编程就可以设置bean的属性（即调用setXxx方法）
- 注意
  - <jsp:setProperty name="book1"
    property="title"
    value="Core Servlets and JavaServer Pages" />
    等价于下面的scriptlet
    <% book1.setTitle("Core Servlets and JavaServer Pages"); %>

示例：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;
  }
}
```

- **bean安装在常规的Java目录中**
  - …/WEB-INF/classes/*directoryMatchingPackageName*
- **bean (和实用工具类) 一定要在包中!**

使用StringBean的JSP页面(代码)

```
<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>
```

使用StringBean的JSP页面(结果)

设置bean的属性 案例1：显式的转换和赋值

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

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

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

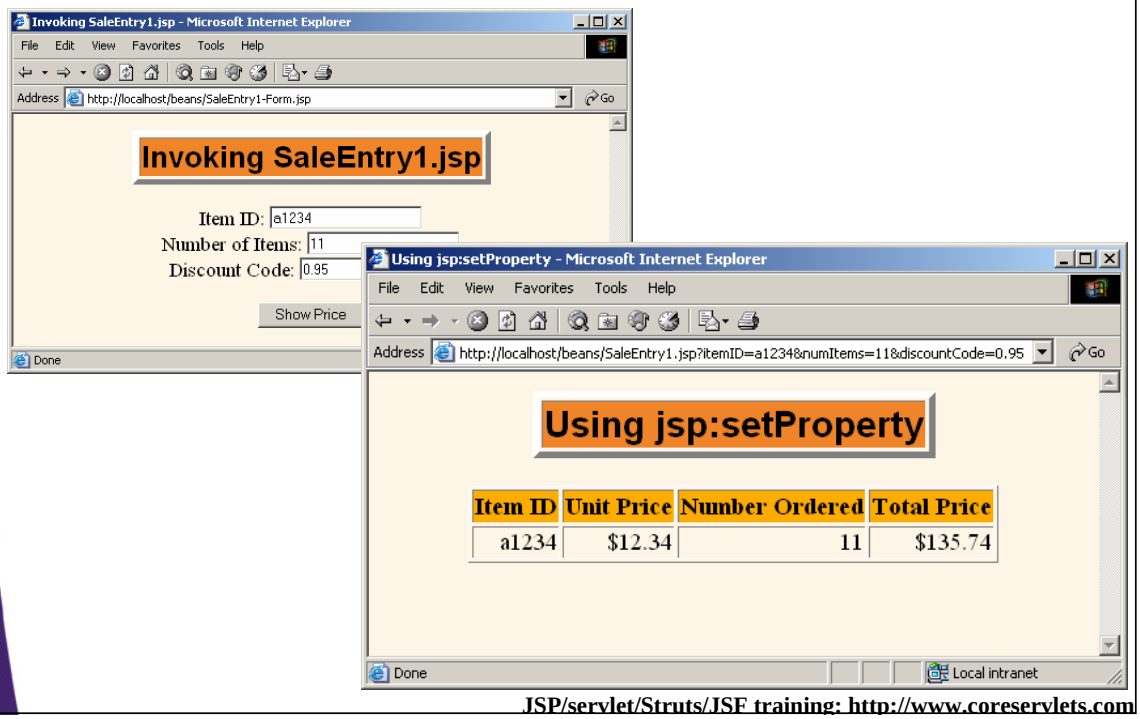
设置bean的属性 案例1：显式的转换和赋值

```
<%
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 %>" />
```


设置bean的属性 案例1：显式的转换和赋值

```
<%
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 %>" />
```

设置bean的属性 案例1：显式的转换和赋值



案例2: 将单个属性与输入参数关联

- **jsp:setProperty的param属性可以用来标示：**
  - 值应该来自于指定的请求参数
  - 对于所期望的值属于标准类型的属性，应该自动执行类型转换
    - boolean, Boolean, byte, Byte, char, Character, double, Double, int, Integer, float, Float, long, or Long.

案例2: 将单个属性与输入参数关联

```
<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" />
```


案例 3: 将所有属性与输入参数关联

- 使用"*"作为jsp:setProperty的property属性的值来标示
  - 值应该来自于名称与属性名相匹配的请求参数
  - 应该执行简单的自动类型转换

案例 3: 将所有属性与输入参数关联

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

- 对于生成“表单bean”（属性由表单提交的数据来填充的对象）来说这项功能极为方便。
  - 我们甚至可以用多个表单来完成这个过程，每次提交填充对象的部分数据。

共享bean

- 我们可以使用scope属性来指定bean存储的其他位置
  - 依旧需要绑定到_jspService中的局部变量
  - <jsp:useBean id="…" class="…" scope="…" />
- 可以让多个servlet或JSP页面共享数据
- 还允许根据条件创建bean
  - 仅当不存在已有对象的情况下才创建新的对象

scope属性的取值

- **page (<jsp:useBean … scope="page"/>或 <jsp:useBean…>)**
  - 默认值。在处理当前请求的过程中，bean对象都应该存放在 PageContext对象中。让同一servlet中的其他方法可以访问该bean。
- **application (<jsp:useBean … scope="application"/>)**
  - Bean将存储在ServletContext中（通过 application变量或调用getServletContext()来访问）。ServletContext由同一Web应用中的所有 servlet共享（或服务器上的所有servlet——在没有另外定义Web应用的情况下）。

scope属性的取值

- **session (<jsp:useBean … scope="session"/>)**
  - bean会被存储在与当前请求关联的HttpSession中，和普通的会话对象一样，在常规servlet中可以使用getAttribute和setAttribute访问到它们。
- **request (<jsp:useBean … scope="request"/>)**
  - 处理当前请求的过程中，bean对象应存储在ServletRequest对象中，可以通过getAttribute访问到它。

bean的条件化操作

- **bean的条件化创建**
  - 仅当找不到相同id和scope的bean时，jsp:useBean才会引发bean新实例的创建。
  - 如果找到相同id和scope的bean，则仅仅是将已经存在的bean绑定到由id指定的变量。
- **bean属性的条件化设置**
  - <jsp:useBean ... />
    替换为
    <jsp:useBean ...>语句</jsp:useBean>
  - 这些语句（jsp:setProperty元素）仅当创建新的bean时才执行，如果找到现有的bean，则不执行。

bean的条件化创建：AccessCountBean

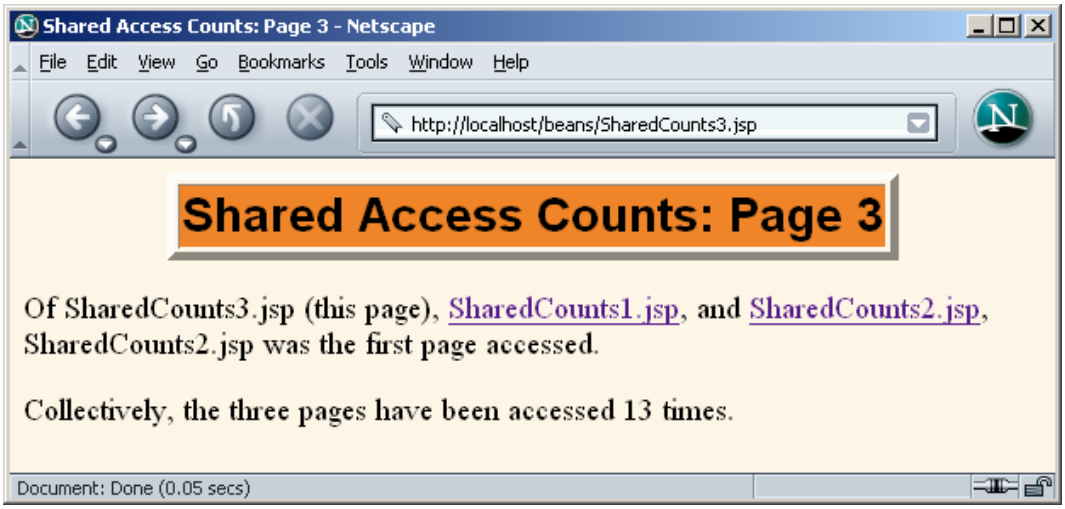
```
public class AccessCountBean {
  private String firstPage;
  private int accessCount = 1;

  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;
  }
}
```

bean的条件化创建：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" />
```


访问SharedCounts1, SharedCounts2, SharedCounts3

- SharedCounts2.jsp被首先访问。
- 之前页面已经被任意数目的客户访问了12次。



共享bean的四种不同方式

- 使用非共享 (作用域为页面)的bean。
- 共享作用域为请求的bean。
- 共享作用域为会话的bean。
- 共享作用域为应用 (即作用域为ServletContext)的bean。

- 注意：
  - 不同的bean要使用不同的名字 (也就是jsp:useBean中的id)。

四种共享bean的方式：bean代码

```
package coreservlets;

public class BakedBean {
  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;
  }
}
```

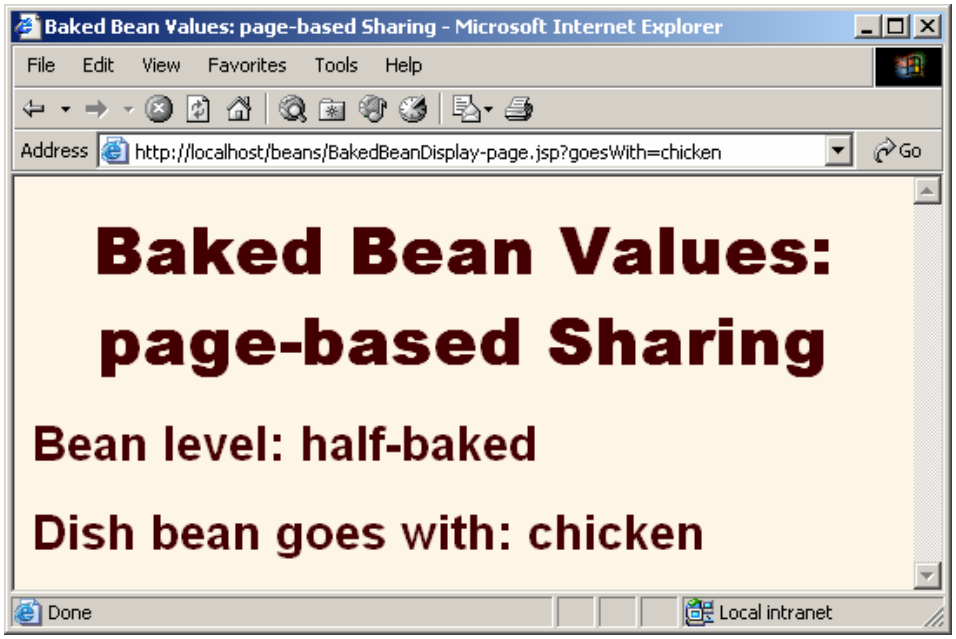
共享bean之例1：作用域为页面（非共享）

- **创建bean**
  - 使用jsp:useBean，指定scope=“page”（或者根本不指明作用域，因为页面是作用域）。
- **修改bean**
  - 使用jsp:setProperty，指定property="*"。
  - 然后，提供与bean的属性名相匹配的请求参数。
- **访问bean**
  - 使用jsp:getProperty.

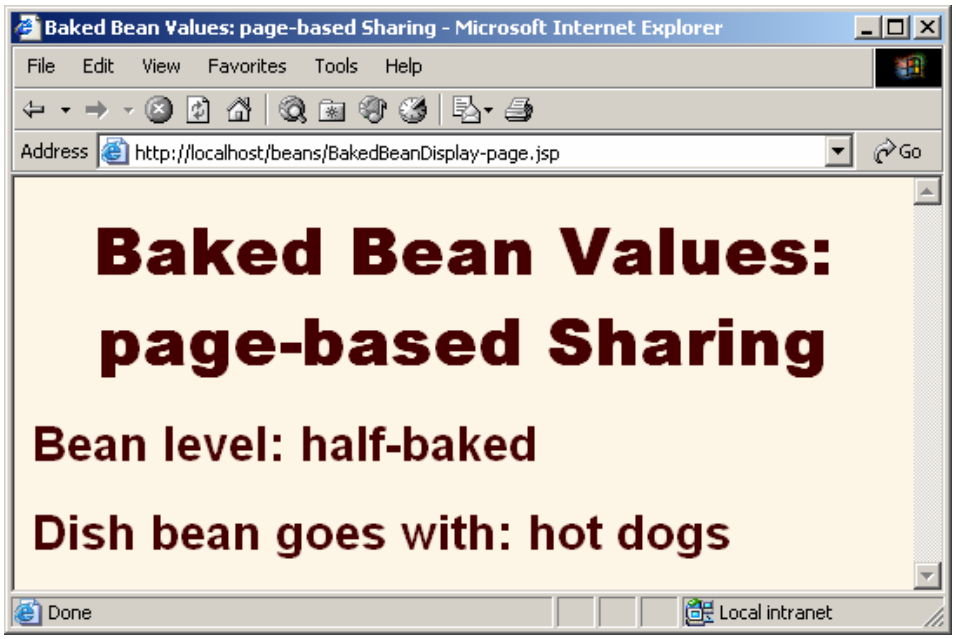
共享bean之例1：作用域为页面（非共享）

```
…
<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>
```

共享bean之例1： 结果(初次访问)



共享bean之例1： 结果(后续访问)



共享bean之例2：基于请求的共享

- **创建bean**
  - 使用jsp:useBean，指定scope="request"。
- **修改bean**
  - 使用jsp:setProperty，指定property="*"。
  - 然后，提供与属性名相匹配的请求参数。
- **在主页面中访问bean**
  - 使用jsp:getProperty.
  - 然后，使用jsp:include调用次级页面。
- **在次级页面（被包括）中访问bean**
  - 使用jsp:useBean，指定与主页面中相同的id，同样scope="request"。
  - 然后，使用jsp:getProperty。

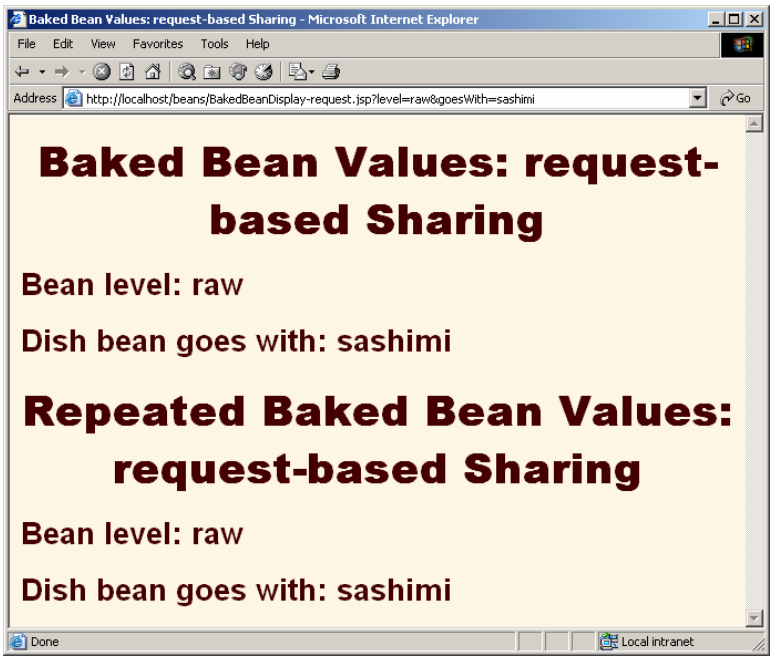
基于请求的共享：主页面的代码

```
…
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="BakedBeanDisplay-snippet.jsp" />
</BODY></HTML>
```

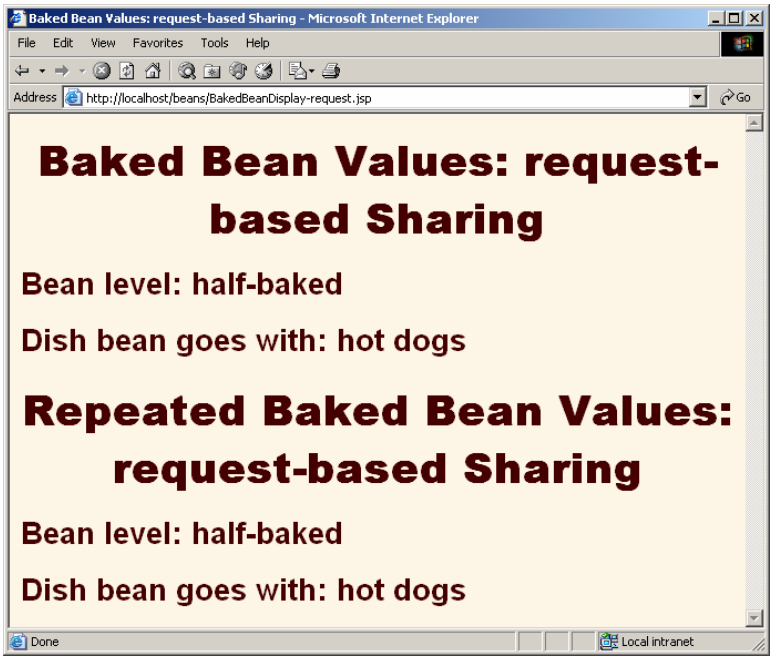
基于请求的共享：被包括页面的代码

```
<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>
```


基于请求的共享：结果(初次访问)



基于请求的共享：结果(后续请求)



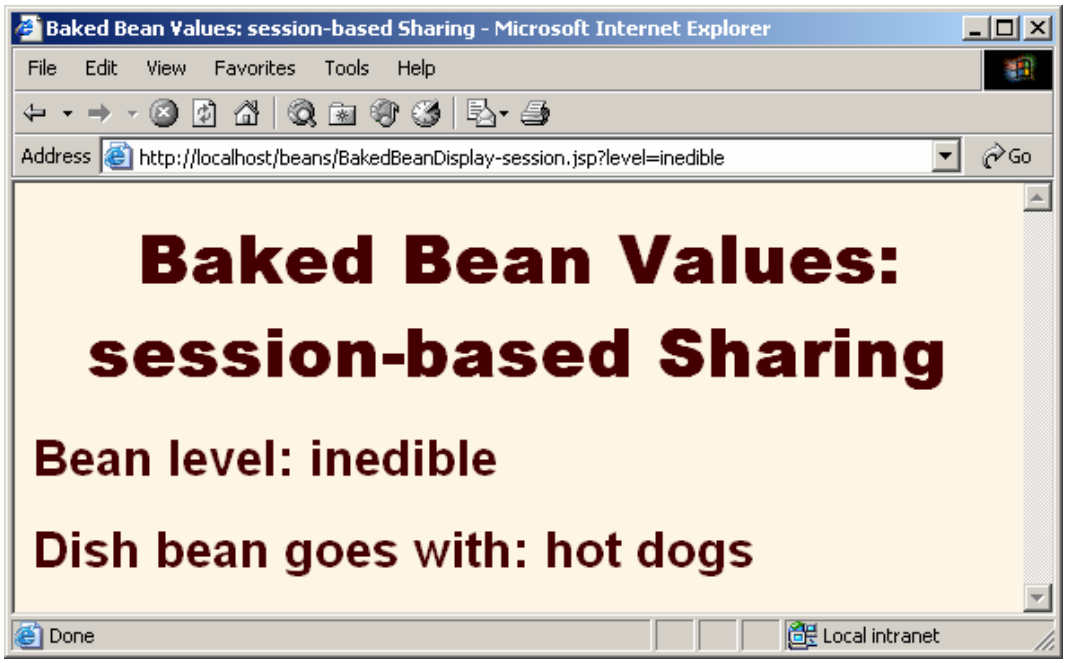
共享bean的例3：基于会话的共享

- **创建bean**
  - 使用jsp:useBean，指定scope="session"。
- **修改bean**
  - 使用jsp:setProperty，指定property="*"。
  - 然后，提供与bean的属性名相匹配的请求参数。
- **在初始的请求中访问bean**
  - 在jsp:setProperty被调用的请求中，使用jsp:getProperty。
- **对bean的后续访问**
  - 在不包括请求参数（因而也就不会调用jsp:setProperty）的请求中，使用jsp:getProperty。如果该请求来自于相同的用户（在会话超时之前），就能够得到之前修改的值。如果请求来自于不同的客户（或在会话超时之后），则得到是新创建的bean。

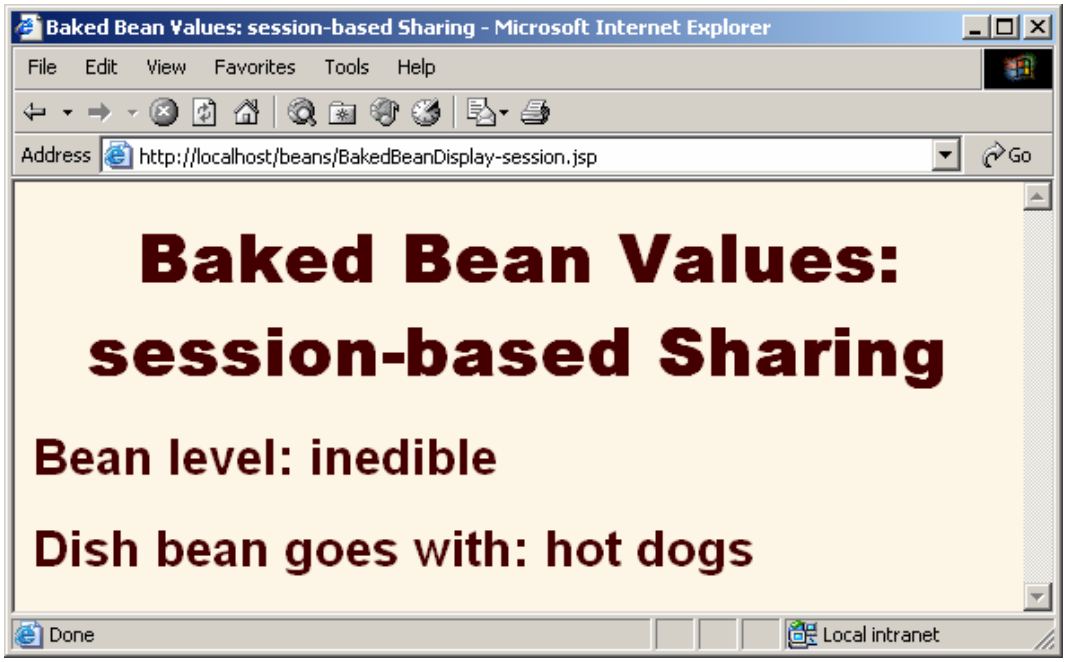
基于会话的共享：代码

```
…
<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>
```


基于会话的共享： 结果(初始请求)



基于会话的共享： 结果 (后续请求– 同一客户)



基于会话的共享： 结果 (后续请求– 新客户)

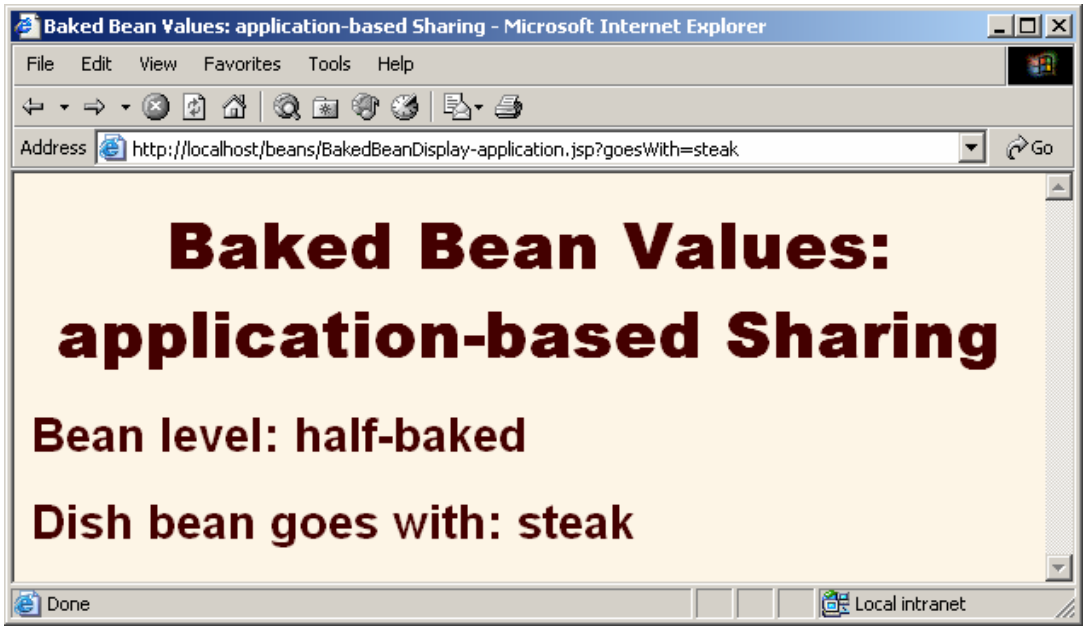
共享bean之例4：基于应用的共享

- **创建bean**
  - 使用jsp:useBean，指定scope="application"。
- **修改bean**
  - 使用jsp:setProperty，指定property="*"。
  - 然后，提供与bean的属性名相匹配的请求参数。
- **在初始的请求中访问bean**
  - 在jsp:setProperty被调用的请求中，使用jsp:getProperty
- **对bean的后续访问**
  - 在不包括请求参数（因而也就不会调用jsp:setProperty）的请求中，使用jsp:getProperty。不管这个请求是来自于相同客户或不同客户（不管会话是否超时），都能够看到之前更改的值。

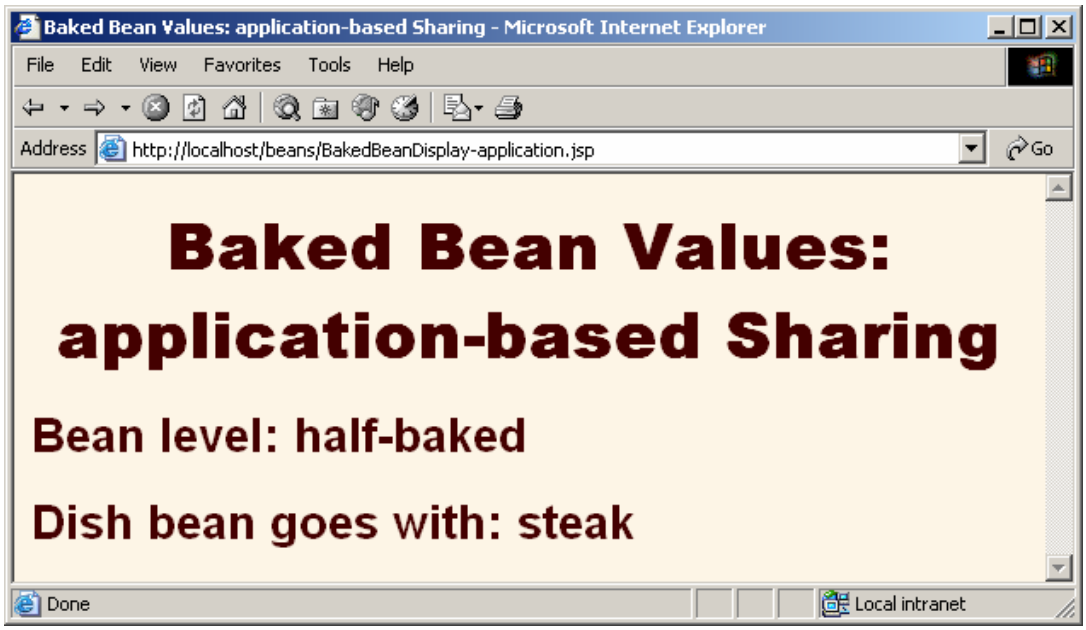
基于应用的共享：代码

```
<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>
```

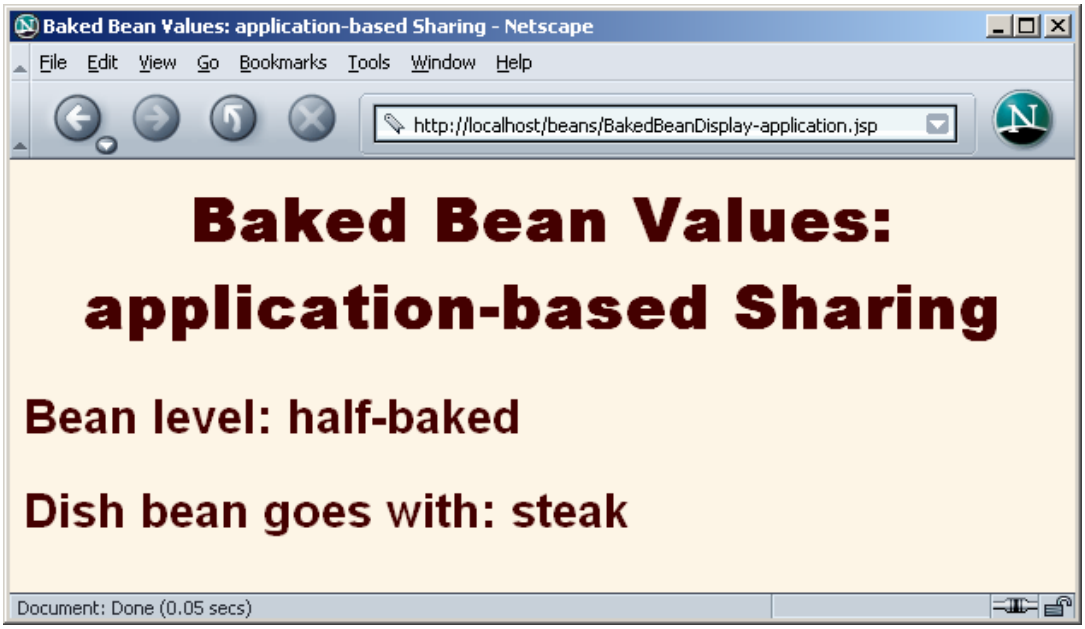
基于应用的共享：结果 (初始请求)



基于应用的共享：结果 (后续请求 – 同一客户)



基于应用的共享：结果 (后续请求 – 新客户)



小结

- **jsp:useBean的优点**
  - 隐藏了Java语法
  - 将请求参数与Java对象关联起来更加容易(bean属性)
  - 使得在多个请求或servlets/JSP间共享对象得到简化
- **jsp:useBean**
  - 创建或使用bean
- **jsp:getProperty**
  - 将bean的属性property (即get*Xxx 调用*)放到servlet的输出中
- **jsp:setProperty**
  - 设置bean属性（即向setXxx传递值）

问题?

JSP, Servlet, & Struts Training Courses: http://courses.coreservlets.com
Available in US, China, Taiwan, HK, and Worldwide

JSP and Servlet Books from Sun Press: http://www.coreservlets.com
Available in English, Chinese (simplified and traditional script), and 12 other languages