



JSP Scripting Elements

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

- **Basic syntax**
- **Types of JSP scripting elements**
- **Expressions**
- **Predefined variables**
- **Scriptlets**
- **Declarations**

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**

3

Scripting Elements

www.coreservlets.com

Design Strategy: Limit Java Code in JSP Pages

- **You have two options**
 - Put 25 lines of Java code directly in the JSP page
 - Put those 25 lines in a separate Java class and put 1 line in the JSP page that invokes it
- **Why is the second option *much* better?**
 - **Development.** You write the separate class in a Java environment (editor or IDE), not an HTML environment
 - **Debugging.** If you have syntax errors, you see them immediately at compile time. Simple print statements can be seen.
 - **Testing.** You can write a test routine with a loop that does 10,000 tests and reapply it after each change.
 - **Reuse.** You can use the same class from multiple pages.

4

Scripting Elements

www.coreservlets.com

Basic Syntax

- **HTML Text**
 - `<H1>Blah</H1>`
 - Passed through to client. Really turned into servlet code that looks like
 - `out.print("<H1>Blah</H1>");`
- **HTML Comments**
 - `<!-- Comment -->`
 - Same as other HTML: passed through to client
- **JSP Comments**
 - `<%-- Comment --%>`
 - Not sent to client
- **To get `<%` in output, use `<\%`**

Types of Scripting Elements

- **Expressions**
 - Format: `<%= expression %>`
 - Evaluated and inserted into the servlet's output. I.e., results in something like `out.print(expression)`
- **Scriptlets**
 - Format: `<% code %>`
 - Inserted verbatim into the servlet's `_jspService` method (called by service)
- **Declarations**
 - Format: `<%! code %>`
 - Inserted verbatim into the body of the servlet class, outside of any existing methods

JSP Expressions

- **Format**
 - `<%= Java Expression %>`
- **Result**
 - Expression evaluated, converted to String, and placed into HTML page at the place it occurred in JSP page
 - That is, expression placed in `_jspService` inside `out.print`
- **Examples**
 - Current time: `<%= new java.util.Date() %>`
 - Your hostname: `<%= request.getRemoteHost() %>`
- **XML-compatible syntax**
 - `<jsp:expression>Java Expression</jsp:expression>`
 - XML version not supported by Tomcat 3. Until JSP 1.2, servers are not required to support it. Even then, you cannot mix versions within a single page.

7

Scripting Elements

www.coreservlets.com

JSP/Servlet Correspondence

- **Original JSP**

```
<H1>A Random Number</H1>  
<%= Math.random() %>
```

- **Possible resulting servlet code**

```
public void _jspService(HttpServletRequest request,  
                        HttpServletResponse response)  
    throws ServletException, IOException {  
    response.setContentType("text/html");  
    HttpSession session = request.getSession(true);  
    JspWriter out = response.getWriter();  
    out.println("<H1>A Random Number</H1>");  
    out.println(Math.random());  
    ...  
}
```

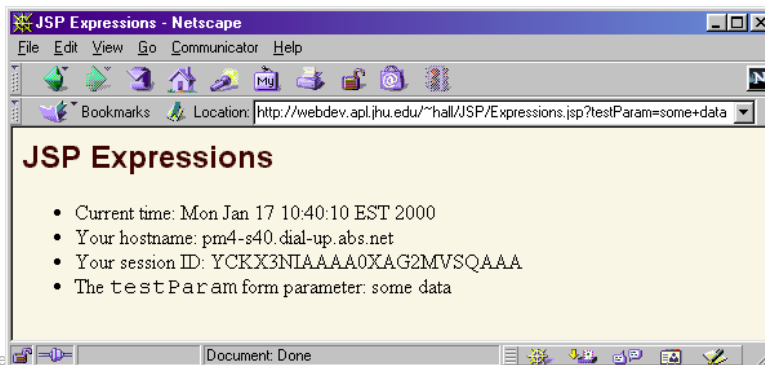
8

Scripting Elements

www.coreservlets.com

Example Using JSP Expressions

```
<BODY>
<H2>JSP Expressions</H2>
<UL>
  <LI>Current time: <%= new java.util.Date() %>
  <LI>Your hostname: <%= request.getRemoteHost() %>
  <LI>Your session ID: <%= session.getId() %>
  <LI>The <CODE>testParam</CODE> form parameter:
      <%= request.getParameter("testParam") %>
</UL>
</BODY>
```



9

Scripting Elements

Document: Done

coreservlets.com

Predefined Variables

- **request**
 - The HttpServletRequest (1st argument to service/doGet)
- **response**
 - The HttpServletResponse (2nd arg to service/doGet)
- **out**
 - The Writer (a buffered version of type JspWriter) used to send output to the client
- **session**
 - The HttpSession associated with the request (unless disabled with the session attribute of the page directive)
- **application**
 - The ServletContext (for sharing data) as obtained via `getServletContext()`.

10

Scripting Elements

www.coreservlets.com

JSP Scriptlets

- **Format**
 - `<% Java Code %>`
- **Result**
 - Code is inserted verbatim into servlet's `_jspService`
- **Example**
 - `<%`
String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData);
`%>`
 - `<% response.setContentType("text/plain"); %>`
- **XML-compatible syntax**
 - `<jsp:scriptlet>Java Code</jsp:scriptlet>`

JSP/Servlet Correspondence

- **Original JSP**

```
<%= foo() %>  
<% bar(); %>
```
- **Possible resulting servlet code**

```
public void _jspService(HttpServletRequest request,  
                        HttpServletResponse response)  
    throws ServletException, IOException {  
    response.setContentType("text/html");  
    HttpSession session = request.getSession(true);  
    JspWriter out = response.getWriter();  
    out.println(foo());  
    bar();  
    ...  
}
```

Example Using JSP Scriptlets

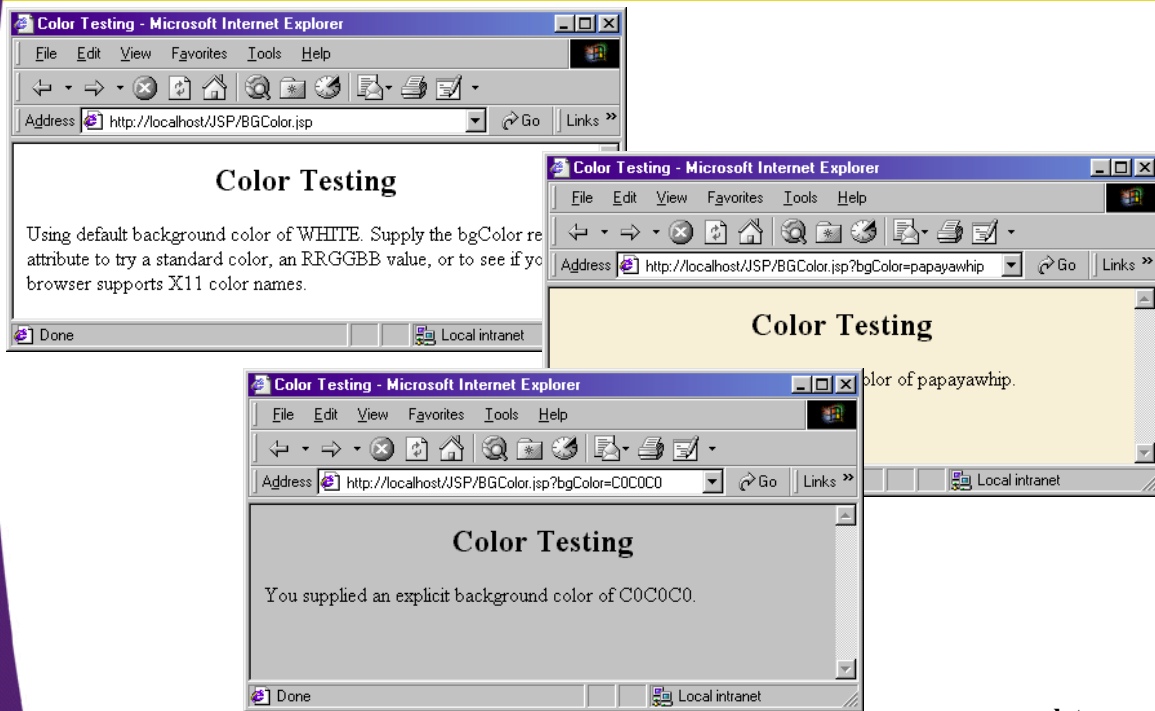
```
<!DOCTYPE HTML PUBLIC
    "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
    <TITLE>Color Testing</TITLE>
</HEAD>
<%
String bgColor = request.getParameter("bgColor");
boolean hasExplicitColor;
if (bgColor != null) {
    hasExplicitColor = true;
} else {
    hasExplicitColor = false;
    bgColor = "WHITE";
}
%>
```

Example Using JSP Scriptlets (Continued)

```
<BODY BGCOLOR="<%= bgColor %>">
<H2 ALIGN="CENTER">Color Testing</H2>
<%
if (hasExplicitColor) {
    ...
} else {
    ...
}
%>

</BODY>
</HTML>
```

JSP Scriptlets: Results



15

Scripting Elements

www.coreservlets.com

Using Scriptlets to Make Parts of the JSP File Conditional

- **Point**
 - Scriptlets are inserted into servlet exactly as written
 - Need not be complete Java expressions
 - Complete expressions are usually clearer and easier to maintain, however
- **Example**
 - ```
<% if (Math.random() < 0.5) { %>
Have a nice day!
<% } else { %>
Have a lousy day!
<% } %>
```
- **Representative result**
  - ```
if (Math.random() < 0.5) {
    out.println("Have a <B>nice</B> day!");
} else {
    out.println("Have a <B>lousy</B> day!");
}
```

16

Scripting Elements

www.coreservlets.com

JSP Declarations

- **Format**
 - `<%! Java Code %>`
- **Result**
 - Code is inserted verbatim into servlet's class definition, outside of any existing methods
- **Examples**
 - `<%! private int someField = 5; %>`
 - `<%! private void someMethod(...) {...} %>`
- **Design consideration**
 - Fields are clearly useful. For methods, it is usually better to define the method in a separate Java class.
- **XML-compatible syntax**
 - `<jsp:declaration>Java Code</jsp:declaration>`

17

Scripting Elements

www.coreservlets.com

JSP/Servlet Correspondence

- **Original JSP**

```
<H1>Some Heading</H1>
<%!
    private String randomHeading() {
        return("<H2>" + Math.random() + "</H2>");
    }
%>
<%= randomHeading() %>
```

- **(Alternative: make randomHeading a static method in a separate Java class)**

18

Scripting Elements

www.coreservlets.com

JSP/Servlet Correspondence

- Possible resulting servlet code

```
public class xxxx implements HttpJspPage {
    private String randomHeading() {
        return("<H2>" + Math.random() + "</H2>");
    }

    public void _jspService(HttpServletRequest request,
                            HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession(true);
        JspWriter out = response.getWriter();
        out.println("<H1>Some Heading</H1>");
        out.println(randomHeading());
        ...
    }
}
```

Example Using JSP Declarations

```
<!DOCTYPE HTML PUBLIC
    "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>JSP Declarations</TITLE>
<LINK REL=STYLESHEET
    HREF="JSP-Styles.css"
    TYPE="text/css">
</HEAD>

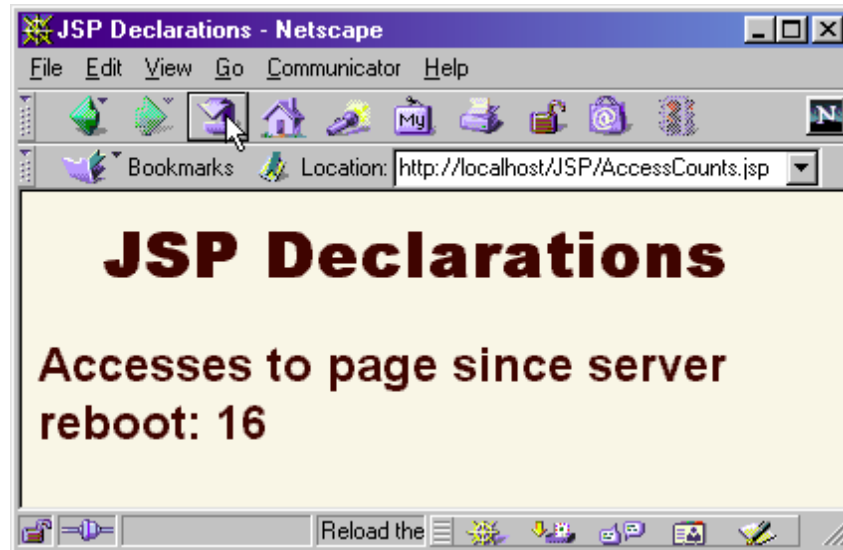
<BODY>
<H1>JSP Declarations</H1>

<%! private int accessCount = 0; %>
<H2>Accesses to page since server reboot:
<%= ++accessCount %></H2>

</BODY>
</HTML>
```

JSP Declarations: Result

- After 15 total visits by an arbitrary number of different clients



JSP Declarations: the jspInit and jspDestroy Methods

- JSP pages, like regular servlets, sometimes want to use init and destroy
- Problem: the servlet that gets built from the JSP page might already use init and destroy
 - Overriding them would cause problems.
 - Thus, it is illegal to use JSP declarations to declare init or destroy.
- Solution: use **jspInit** and **jspDestroy**.
 - The auto-generated servlet is guaranteed to call these methods from init and destroy, but the standard versions of jspInit and jspDestroy are empty (placeholders for you to override).

JSP Declarations and Predefined Variables

- **Problem**

- The predefined variables (request, response, out, session, etc.) are *local* to the `_jspService` method. Thus, they are not available to methods defined by JSP declarations or to methods in helper classes. What can you do about this?

- **Solution: pass them as arguments. E.g.**

```
<%!  
private void someMethod(HttpSession s) {  
    doSomethingWith(s);  
}  
%>  
<% someMethod(session); %>
```

- **Note that the `println` method of `JspWriter` throws `IOException`**

- Use “throws `IOException`” for methods that use `println`

Using JSP Expressions as Attribute Values

- **Static Value**

- `<jsp:setProperty name="author" property="firstName" value="Marty" />`

- **Dynamic Value**

- `<jsp:setProperty name="user" property="id" value='<%= "UserID" + Math.random() %>' />`

Attributes That Permit JSP Expressions

- **The name and value properties of `jsp:setProperty`**
 - See upcoming section on beans
- **The page attribute of `jsp:include`**
 - See upcoming section on including files and applets
- **The page attribute of `jsp:forward`**
 - See upcoming section on integrating servlets and JSP
- **The value attribute of `jsp:param`**
 - See upcoming section on including files and applets

Summary

- **JSP Expressions**
 - Format: `<%= expression %>`
 - Wrapped in `out.print` and inserted into `_jspService`
- **JSP Scriptlets**
 - Format: `<% code %>`
 - Inserted verbatim into the servlet's `_jspService` method
- **JSP Declarations**
 - Format: `<%! code %>`
 - Inserted verbatim into the body of the servlet class
- **Predefined variables**
 - `request`, `response`, `out`, `session`, `application`
- **Limit the Java code that is directly in page**
 - Use helper classes, beans, custom tags, servlet/JSP combo