

JSP Scripting Elements

Lecture 8 – Chapter 10

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**

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

JSP Versions

- **JSP 0.92 (obsolete)**
 - Tomcat 1.x
- **JSP 1.0**
 - First upwardly compatible JSP release
 - Tomcat 2.x
- **JSP 1.1**
 - Based on Servlets 2.2 specification
 - Links JSP releases with Servlet releases
 - Tomcat 3.x
- **JSP 1.2**
 - Current version of JSP technology
 - Based on Servlets 2.3 specification
 - Tomcat 4.x

www.coreservlets.com

Basic Syntax

- **HTML Text (template 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 `<%=`**

www.coreservlets.com

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

www.coreservlets.com

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.

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());
    ...
}
```

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>
```



Scripting Elements

www.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 `getServletConfig().getContext()`

Scripting Elements

www.coreservlets.com

Predefined Variables

- **config**
 - The `ServletConfig` object for this page
- **pageContext**
 - Introduced in JSPs to provide a single point of access to page attributes and a convenient place to store shared data. Each page has a `PageContext` associated with it.
- **page**
 - A synonym for this. Not very useful in Java. Can be used as a handle to this page from another server-side scripting language.

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>`

Scripting Elements

www.coreservlets.com

JSP Scriptlet Uses

- **Scriptlet advantages over Expressions**
 - There are some things scriptlets can do that cannot be done in JSP expressions:
 - Setting response headers
 - Setting status codes
 - Invoking side effects in other methods
 - Writing to the server log
 - Updating a database
 - Programming logic (loops, conditionals, etc.)

Scripting Elements

www.coreservlets.com

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();
    ...
}
```

Scripting Elements

www.coreservlets.com

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";
}
%>
```

Scripting Elements

www.coreservlets.com

Example Using JSP Scriptlets (Continued)

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

Scripting Elements

www.coreservlets.com

JSP Scriptlets: Results



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 nice day!");`
`} else {`
`out.println("Have a lousy day!");`
`}`

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(...) {...} %>`
- **XML-compatible syntax**
 - `<jsp:declaration>Java Code</jsp:declaration>`

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)**

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());
 ...
 }
}
```

Scripting Elements

www.coreservlets.com

## Example Using JSP Declarations

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>JSP Declarations</TITLE>
<LINK REL=STYLE SHEET
 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>

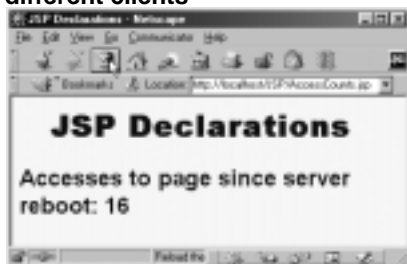
</BODY>
</HTML>
```

Scripting Elements

www.coreservlets.com

## JSP Declarations: Result

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



Scripting Elements

www.coreservlets.com

## 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).

Scripting Elements

www.coreservlets.com

## 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`

Scripting Elements

www.coreservlets.com

## 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() %>' />`

Scripting Elements

www.coreservlets.com

## 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

Scripting Elements

www.coreservlets.com

## 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

Scripting Elements

www.coreservlets.com



## Questions?

Core Servlets & JSP book: [www.coreservlets.com](http://www.coreservlets.com)  
More Servlets & JSP book: [www.moreservlets.com](http://www.moreservlets.com)  
Servlet and JSP Training Courses: [courses.coreservlets.com](http://courses.coreservlets.com)

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