JSP Syntax

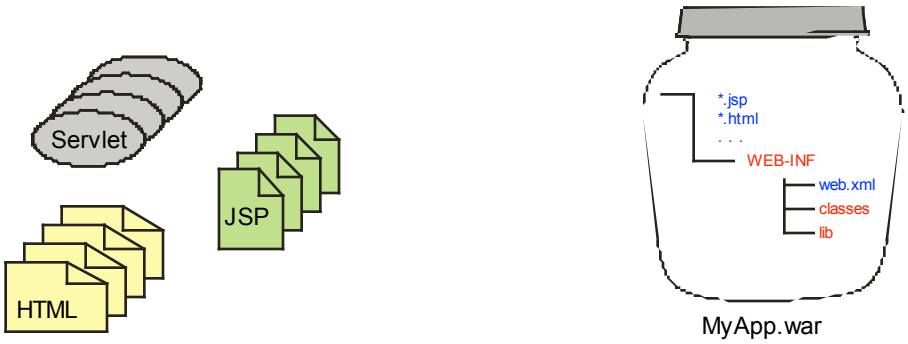
- Elements of a JSP
- JSP Exception Handling

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Introduction to Web Applications

- A **web application** is a collection of servlets, HTML, and JavaServer Pages that work together as a single program
- Optionally, you can package web applications into WAR files, which are similar to JAR files
- This chapter covers the fundamentals of the JSP portion of a web application

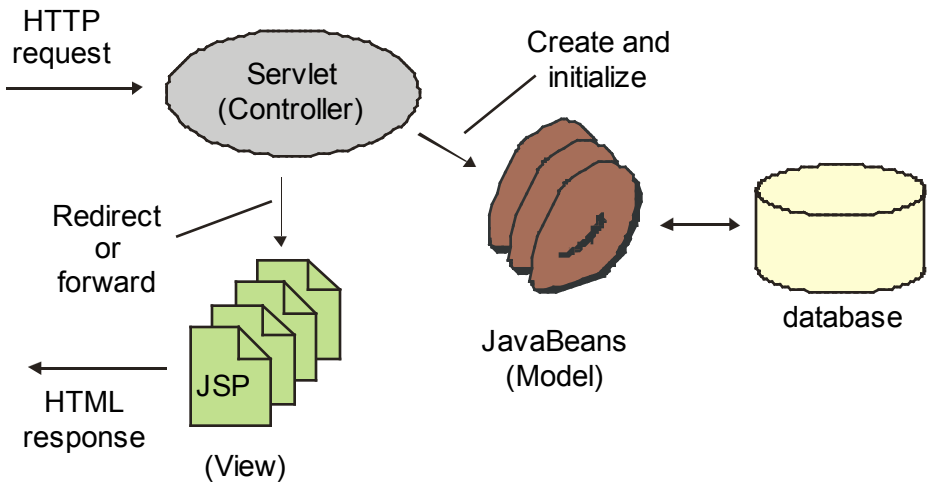


The notion of web applications is precisely defined by the J2EE specification, which mandates a required directory structure and so forth. We will cover web applications in more detail later in the course, but it's a good idea to start thinking about the big picture.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Introduction to Model-View-Controller

- One very useful pattern you can use to combine servlets and JSPs is referred to as MVC
- Until we cover MVC in detail, keep this in mind as we cover JSP fundamentals



2 - 3

There are two basic ways to use JSPs: In the first, sometimes referred to as "Model 1", the JSP directly responds to HTML forms and generates HTML. In the second, referred to as "Model 2" or "Model-View-Controller", we use servlets to respond to the forms. The servlet then transfers control to JSPs which generate the HTML.

Since it's easier, this course first covers Model 1 and then later adds the additional power (and complexity) of MVC.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A JSP Consists Of...

- **Template Data**
  - Typically HTML or XML
  - Written unmodified back to the client via the implicit **out** variable
- Comments
- Elements
  - Use XML or non-XML syntax
  - Element types:
    - Directives
    - Scripting elements (scriptlets)
    - Action elements

Minimal JSP

```
<html>
</html>
```


Here we show the two basic components of a JSP page: template data and elements. Note that there are three different kinds of elements: we will cover each type in this chapter.

A well designed JSP uses mostly template data, directives and action elements. That's because scripting elements, which contain Java code, can be difficult for non-programmers to work with.

We'll start by discussing template data.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Quoting and Escapes

- To insert special characters or sequences, you can "escape" them, depending on the context
- In a scripting element:
  - To insert %>, use %/>
- In template data:
  - To insert <%, use <\%
- In attributes:
  - To insert ', use \'
  - To insert ", use \"
  - To insert %>, use %/>
  - To insert <%, use <\%

The container's JSP processor treats some character sequences specially, for example the greater-than and less-than characters. If you want to use those characters literally, you can "escape" them as shown on this page.

Note: by "attributes", we mean the name-value pairs used within some elements. For example: In the jsp:useBean id="mybean" element, "id" is the attribute's name and "mybean" is the value. You only need to worry about escaping characters in the attribute value, not the name.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Template Data

- The JSP engine passes template data as an argument to *out.print*

```
1    public void _jspService(HttpServletRequest request
2                          , HttpServletResponse response)
3         throws ServletException, java.io.IOException
4    {
5        . . .
6        PageContext pageContext =
7            __jspFactory.getPageContext(this, request,
8                        response,  null, true, 8192, true);
9        JspWriter out = pageContext.getOut();
10
11       out.print("<html>\r\n</html>");
12
13       . . .
```

This page shows part of the generated servlet from the minimal JSP on the last page. Note in line 11 how the container translated the first line of template data into a string which is then output back to the client's browser.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A JSP Consists Of...

- Template Data
  - Typically HTML or XML
  - Written unmodified back to the client via the implicit **out** variable
- **Comments**
- Elements
  - Use XML or non-XML syntax
  - Element types:
    - Directives
    - Scripting elements (scriptlets)
    - Action elements

Next we will cover how to write comments in a JSP.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

JSP Comments

- You can create comments that are sent to the output stream:

```
<!-- Define a table -->
```

- You can create comments that are not sent to the output stream:

```
<%-- Initialize counter --%>
<% /* Java
      comment */
   // another Java comment
%>
```

The first comment style shown is the same as HTML comments. You should note that if the user does a "View-Source" in their browser, they will see such comments. You should therefore avoid putting any sensitive information in such comments.

To create comments that are not emitted into the generated servlet, use the two styles shown at the bottom of the page. Note that the second example is just a normal Java comment within a scripting element.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A JSP Consists Of...

- Template Data
  - Typically HTML or XML
  - Written unmodified back to the client via the implicit **out** variable
- Comments
- **Elements**
  - Use XML or non-XML syntax
  - Element types:
    - Directives
    - Scripting elements (scriptlets)
    - Action elements

Now we will start examining elements.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Element Syntax

- **Directives** and **scriptlets** non-XML syntax while **action elements** use actual XML syntax
- Some XML syntax rules:
  - Start tags and end tags
  - Case sensitive
  - Can have attributes (must be quoted)
  - Empty elements OK

While Sun was developing the JSP syntax, XML became popular and useful, but XML wasn't a big deal when they decided on the syntax for elements. That's why some elements use non-XML syntax while others use actual XML. This page lists some of the XML rules for proper syntax.

You should also note that Sun has since defined true XML-syntax replacements for the elements that don't use XML -- refer to the "JSP Pages as XML Documents" section of the JSP specification for details.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is a Directive?

- Directives are commands to the JSP container
- Directive syntax is of the form:

<%@ directive {attr="value"}* %>

- Note that directive syntax is not strictly XML-compatible
- Directives include:
    - page
    - include
    - taglib

```
<!-- Example directive -->
<%@ page language="java" %>
```

Directives give information to the container, but themselves do not produce any output.

The directive syntax is not XML compliant since the characters that start the directive (left-angle, percent, ampersand) are different from the characters that end the directive (percent, right-angle). XML requires that end-tags be the same text as start-tags (the end-tag has a leading slash character). Note that as mentioned before, Sun has defined another strictly XML-compliant syntax for directives: see the JSP specification for details.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The page Directive

- The **page** directive defines characteristics of the current page
- If you write an unrecognized page directive, the container will fail to translate the JSP into a servlet

```
1    <%@ page import="javax.sql.*, java.io.*"
2        errorPage="errors.jsp"
3        info="Initial Screen" %>
```

The page directive lets a JSP communicate with the container and set a few important properties. The properties are defined as attributes and the value part of the attribute must be quoted. If you specify multiple attributes, you separate each attribute from the next with white space (e.g. space characters or new-line characters).

Next we will show the possible attributes you can use in a page directive.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The page Directive, cont'd

Attribute	Default	Description
language	Java	Specifies scripting language
extends	defined by container	Defines superclass of generated servlet
import	java.lang.*, java.servlet.*, java.servlet.jsp.*, java.servlet.http.*	Packages imported into generated servlet. Comma-separated list
session	true	Whether or not the JSP participates in a session
buffer	8KB minimum	***out*** stream buffer size. Use "none" for no buffering
autoFlush	true	Specifies whether the container should automatically write buffer to client when full

This table lists some of the page directive attributes and their default values.

Perhaps the most commonly used attribute is import, which saves typing in scripting elements (you can avoid using fully qualified package names). Note that the import list is comma-separated and that if you write your own import attribute (like the one shown on the last page), it augments the defaults rather than replacing the defaults.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The page Directive, cont'd

Attribute	Default	Description
info	N/A	String describing the JSP
isThreadSafe	true	If true, the container may route multiple simultaneous requests to the generated servlet
errorPage	defined by the container	Specifies the URL that the container invokes if an exception occurs. The error page must itself be a JSP.
isErrorPage	false	Specifies that the current page is an error page. The implicit **exception** variable is valid for the page.
contentType	text/html; charSet=ISO-8859-1	Specifies the MIME type and character set of the generated output

Most of these settings are fairly self-explanatory, but we will cover how error pages work in more detail later.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The include Directive

- The include directive lets you insert text into the current JSP
- You can use this directive to insert "boilerplate" template text
- There's also a <jsp:include> action element which we will cover later

```
<%@ include file="standard.html" %>
```


This directive lets you build up a JSP from external sources. It's typically used to insert copyright notices, common logos and so forth.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The taglib Directive

- You can use the taglib directive to reference a set of "tags" that extend the JSP syntax
- Common uses of taglibs include database access looping constructs, internationalization and so forth
- The Java Standard Tag Library (JSTL) contains many such tags
- In addition, several third-party tag libraries are available, including the Apache taglib project

We will cover tag libraries (also known as custom actions) in detail in a later chapter.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A JSP Consists Of...

- Template Data
  - Typically HTML or XML
  - Written unmodified back to the client via the implicit **out** variable
- Comments
- Elements
  - Use XML or non-XML syntax
  - Element types:
    - Directives
    - **Scripting elements (scriptlets)**
    - Action elements

Let's take a closer look at a topic we've already touched on: scripting elements.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Writing Scriptlets

- Scriptlets contain Java code -- you can write as many as you wish, but the JSP engine combines them all sequentially into the _jspService method

```
1     <html><body>
2     <%
3         for ( int i = 0; i < 5; i++ )
4         {
5     %>
6     <p> Hello world</p>
7     <%
8         }
9     %>
10    </body></html>
```


Here we show a JSP with two scripting elements and a line of template data in between (line 6). Sometimes, writing multiple elements like this is the most elegant way to produce output. Here we are avoiding an extra "out.println" to generate the HTML in line 6.

Scripting elements are not as useful as they were when JSPs first came out since we now have an expression language and tag libraries.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

JSP Expressions

- A JSP expression is a convenient way to write to the implicit **out** variable, converting to a string if required
- JSP expressions use the <%= . . . %> syntax

```
1    <html><body>
2    <%
3        for ( int i = 0; i < 5; i++ )
4        {
5    %>
6    <p> Counter is: <%= i %> </p>
7    <%
8        }
9    %>
10   </body></html>
```


Here we use a JSP expression to generate HTML that contains the contents of a variable. JSP expressions can contain a single line of Java code that evaluates to something that's convertible to a String. Note the lack of a semicolon in the expression -- the container's JSP translator takes care of that.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A JSP Consists Of...

- Template Data
  - Typically HTML or XML
  - Written unmodified back to the client via the implicit **out** variable
- Comments
- Elements
  - Use XML or non-XML syntax
  - Element types:
    - Directives
    - Scripting elements (scriptlets)
    - **Action elements**

Now we will examine action elements.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is an Action Element?

- An action element is a tag using XML syntax that directs the container to execute a command
- The JSP 1.1 specification defines several standard actions, and developers can extend these by writing custom tag libraries

```
1    <jsp:useBean id="myBean"
2       class="mypackage.MyBean" >
3
4        <jsp:setProperty name="myBean"
5            property="myProperty"
6            value="abcd" / >
7
8    </jsp:useBean>
```

Here we show an example of the jsp:useBean and jsp:setProperty action elements. At this point, don't worry too much about what they do (we will cover them later). Instead, take a close look at the syntax and note how the syntax strictly follows XML rules.

Also note that the jsp:setProperty element is an empty element while the jsp:useBean element has a body.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Standard Action Elements

Element	Description
jsp:useBean	Assigns a JavaBean reference to a variable so the JSP can access the Bean's properties and methods
jsp:setProperty	Modifies a Bean's property
jsp:getProperty	Retrieves the value of a Bean's property
jsp:include	Inserts external content into the JSP. Similar to the **include directive**
jsp:forward	Dispatches the current request to another URL
jsp:param	Provides key-value information to jsp:forward, jsp:include or jsp:plugin elements
jsp:plugin	Lets the page insert the browser-specific HTML that will cause the loading of a browser plugin

This table lists the standard action elements. We will cover some of these more detail later, especially the first three, which let a JSP manipulate JavaBeans.

The jsp:include action element is similar to the include directive, but works a bit differently. The include directive inserts the content as the container translates the JSP into a servlet, while the jsp:include action element inserts the content during request processing. That allows jsp:include to be more dynamic (e.g. it could include the output of a JSP or servlet), but possibly taking a performance hit.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Exceptions and Error Pages

- Code in a scriptlet or an external JavaBean can throw Java exceptions
- If that happens, the container creates a **Throwable** object and then forwards the request to the offending page's error page (if any)

```
1    <html><body>
2    <%@ page errorPage="error.jsp" %>
3    <%
4        String s = null;
5        out.println ( s.length() );
6    %>
7    </body></html>
```


Recall the page directive that we covered earlier -- it lets you define an error page that the container invokes if the current page throws an exception. Here we have a simple JSP that has an obvious NullPointerException in line 6. When the exception is thrown, the container will terminate the offending page and forward the request to the "error.jsp" page.

If the offending page does not specify an error page, the container returns an error response back to the user's browser, typically an Internal Server Error (500) response.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Exceptions and Error Pages, cont'd

- An error page can access the implicit **exception** variable to retrieve information about the problem

```
1    <html><body>
2
3    <%@ page isErrorPage="true" %>
4
5    <h2>Error details:</h2>
6    <%= exception.toString() %>
7
8    </body></html>
```


Here we show the rudimentary "error.jsp" referenced on the previous page. Note the page directive in line 3 that defines this as an error page -- that definition ensures that the page can access the implicit "exception" variable, as shown in line 6.

Note that your web application can have as many different error pages as you deem necessary, but that any individual JSP can only reference a single error page.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned:

- Just about everything there is to know about JSP syntax, including elements, template data and comments
- How to write error pages and reference them from other JSPs

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum