CNT 4714: Enterprise Computing Spring 2010

Introduction to JavaServer Pages (JSP) – Part 1

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cnt4714/spr2010

School of Electrical Engineering and Computer Science
University of Central Florida

Introduction to JavaServer Pages (JSP)

- JavaServer Pages (JSP) is an extension of servlet technology.
- Like servlets, JSPs simplify the delivery of dynamic web content. They allow web programmers to create dynamic content by reusing predefined components and by interacting with components using server-side scripting.
- JSPs can reuse JavaBeans and create custom tag libraries that encapsulate complex, dynamic functionality.
- JSP classes and interfaces can be found in packages `javax.servlet.jsp` and `javax.servlet.jsp.tagext`.

Introduction to JSP (cont.)

- There are four key components to JSPs

1. Directives: messages to the JSP container (server component executing the JSP) that enable the programmer to specify page settings, include content from other resources and specify custom tag libraries to use in a JSP.

2. Actions: encapsulate functionality based on the information sent to the server as part of a specific client request. They can also create Java objects for use in JSP scriplets.

3. Scripting elements: enable the programmer to insert Java code that interacts with components in a JSP to perform request processing.

4. Tag libraries: are part of the tag extension mechanism that enables programmers to create custom tags. Typically, most useful for web page designers with little knowledge of Java.

Introduction to JSP (cont.)

- In some ways, JSPs look like standard XHTML or XML documents.
- JSPs normally include XHTML or XML markup. Such markup is known as fixed-template data or fixed-template text.
  - Fixed-template data/text often helps a programmer decide whether to use a servlet or a JSP. Recall that JSPs are most often used when most of the content sent to the client is fixed-template data and little or none of the content is generated dynamically with Java code. Servlets are more commonly used when only a small amount of the content returned to the client is fixed-template data.

Introduction to JSP (cont.)

- When a JSP-enabled server receives the first request for a JSP, the JSP container translates the JSP into a Java servlet that handles the current request as well as all future requests to the JSP.
- Literal text in the JSP becomes string literals in the servlet that represents the translated JSP.
- Any errors that occur in compiling the new servlet result in translation-time errors.
- The JSP container places the Java statements that implement the JSP's response in method `_jspService` at translation time.
- If the new servlet compiles properly, the JSP container invokes method `_jspService` to process the request.
- The JSP may respond directly or may invoke other web application components to assist in processing the request. Any errors that occur during request processing are known as request-time errors.

Introduction to JSP (cont.)

- Overall, the request-response mechanism and the JSP life-cycle are the same as those of a servlet.
- JSPs can override methods `jspInit` and `jspDestroy` (similar to servlet methods `init` and `destroy`), which the JSP container invokes when initializing and terminating a JSP.
- A JSP programmer defines these methods using JSP declarations which are part of the scripting mechanism.

The First JSP Example

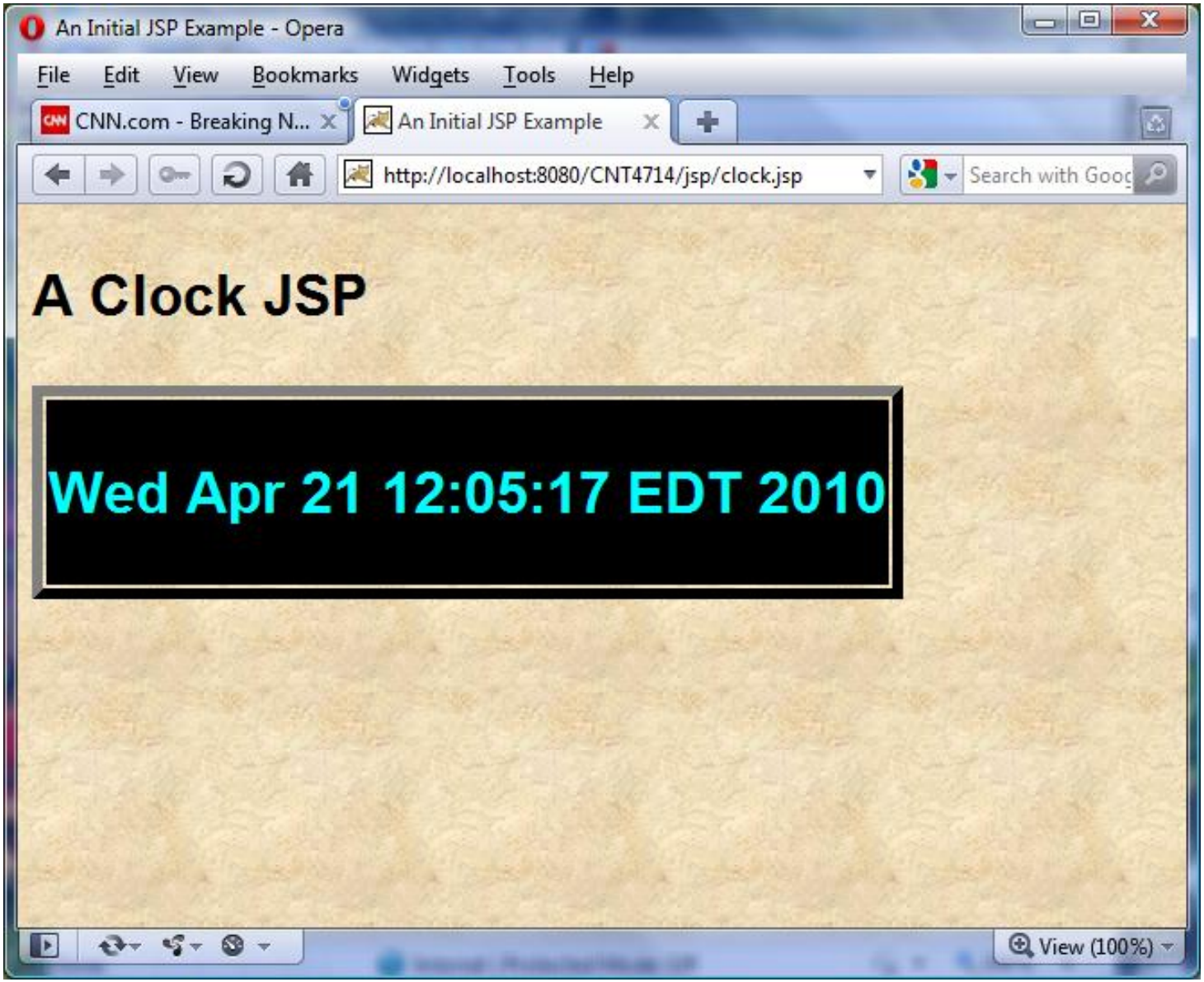
- Our first look at a JSP is with a simple clock JSP which displays the current date and time inserted into a web page using a JSP expression.
- To execute this `clock.jsp` from your own system, as with the servlet examples we've been running – copy the clock.jsp file into the webapps subdirectory you created for your servlet examples.
  - My Tomcat webapps subdirectory is named `CNT4714` and I created a subdirectory named JSP in this directory to hold all the JSP examples. From the index page I created – the JSPs can be executed directly, otherwise…type http://localhost:8080/CNT4714/jsp/clock.jsp to execute this JSP.

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- A clock.jsp -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <meta http-equiv = "refresh" content = "60" />
      <title>An Initial JSP Example</title>
      <style type = "text/css">
         .big { font-family: helvetica, arial, sans-serif;
                font-weight: bold;
                font-size: 2em; }
      </style>
   </head>
   <body>
      <p class = "big">A Clock JSP</p>
      <table style = "border: 6px outset;">
         <tr>
            <td style = "background-color: black;">
               <p class = "big" style = "color: cyan;">

                  <!-- JSP expression to insert date/time -->
                  <%= new java.util.Date() %>
               </p>
            </td>
         </tr>
      </table>
   </body>
</html>
```

XHTML meta-element sets a refresh interval of 60 seconds

JSP expressions are delimited by <%= … %.

Creates a new instance of class Date (package java.util). When the client requests this JSP, this expression inserts the String representation of the date and time in the response to the client.


An Initial JSP Example - Opera
File Edit View Bookmarks Widgets Tools Help
CNN.com - Breaking N...
An Initial JSP Example
http://localhost:8080/CNT4714/jsp/clock.jsp
Search with Goog
A Clock JSP
Wed Apr 21 12:05:17 EDT 2010
View (100%)

Implicit Objects

- Implicit objects provide access to many servlet capabilities in the context of a JSP.
- Implicit objects have four scopes:

1. Application: the JSP container owns objects with application scope. Any JSP can manipulate such objects.
2. Page: objects with page scope can only be manipulated in the page that defines them. Each page has its own instances of the page-scope implicit objects.
3. Request: these objects go out of scope when request processing completes with a response to the client.
4. Session: these objects exist for the client's entire browsing session.

Implicit Objects

Implicit Object	Description
Application Scope	
application	This **javax.servlet.ServletContext** object represents the container in which the JSP executes.
Page Scope	
config	This **javax.servlet.ServletConfig** object represents the JSP configuration options. As with servlets, configuration options can be specified in a Web application descriptor.
exception	This **java.lang.Throwable** object represents the exception that is passed to the JSP error page. This object is available only in a JSP error page.
out	This **javax.servlet.jsp.JspWriter** object writes text as part of the response to a request. This object is used implicitly with JSP expressions and actions that insert string content in a response.
page	This **java.lang.Object** object represents the **this** reference for the current JSP instance.
pageContext	This javax.servlet.jsp.PageContext object hides the implementation details of the Underlying servlet and JSP container and provides JSP programmers with Access to the implicit objects listed in this table.

Implicit Objects

Implicit Object	Description
`response`	This object represents the response to the client. The object normally is an instance of a class that implements **`HttpServletResponse`** (package **`javax.servlet.http`**). If a protocol other than HTTP is used, this object is an instance of a class that implements **`javax.servlet.ServletResponse`**.
Request Scope	
`request`	This object represents the client request. The object normally is an instance of a class that implements **`HttpServletRequest`** (package **`javax.servlet.http`**). If a protocol other than HTTP is used, this object is an instance of a subclass of **`javax.servlet.ServletRequest`**.
Session Scope	
`session`	This **`javax.servlet.http.HttpSession`** object represents the client session information if such a session has been created. This object is available only in pages that participate in a session.
.	

Scripting

- JSPs often present dynamically generated content as part of an XHTML document that is sent to the client in response to a request.
- In some cases, the content is static, but is output only if certain conditions are met during a request (e.g., providing values in a form that submits a request).
- JSP programmers can insert Java code and logic in a JSP using scripting.

Scripting Components

- JSP scripting components include scriplets, comments, expressions, declarations, and escape sequences.
- Scriplets are blocks of code delimited by <% and %>. They contain Java statements that the container places in method `_jspService` at translation time.
- Comments come in three flavors in JSPs: JSP comments, XHTML comments, and scripting language comments.
  - JSP comments are delimited by <%-- and --%>. Can be placed throughout the JSP except inside scriplets.
  - XHTML comments are delimited by <!-- and -->. Can be placed anywhere in the JSP except inside scriplets.
  - Scripting language comments are Java comments (Java is currently the only JSP scripting language which is allowed). Scriplets can use either // or /* and */ as in normal Java.

Scripting Components (cont.)

- JSP comments and scripting language comments are ignored and do not appear in the response to a client. When clients view the source code of a JSP response, they will see only the XHTML comments in the source code.
  - The different comment styles are useful for separating comments that the user should be able to see from those that document logic processed on the server-side.
- Expressions are delimited by <%= and %> and contain a Java expression that is evaluated when a client requests the JSP containing the expression. The container converts the result of a JSP expression to a `String` object, then outputs the `String` as part of the response to the client.

Scripting Components (cont.)

- Declarations are delimited by <%! and %>. Declarations enable the JSP programmer to define variables and methods for use in a JSP. Variables become instance variables of the servlet class that represents the translated JSP. Similarly, methods become members of the class that represents the translated JSP. Declaration of variables and methods in a JSP use Java syntax such as:

```
<%! int increment = 0; %>
```

- Escape sequences are necessary to include special characters or character sequences that the JSP container normally uses to delimit JSP code.
  - Example: literal: <%, escape sequence is: <\%

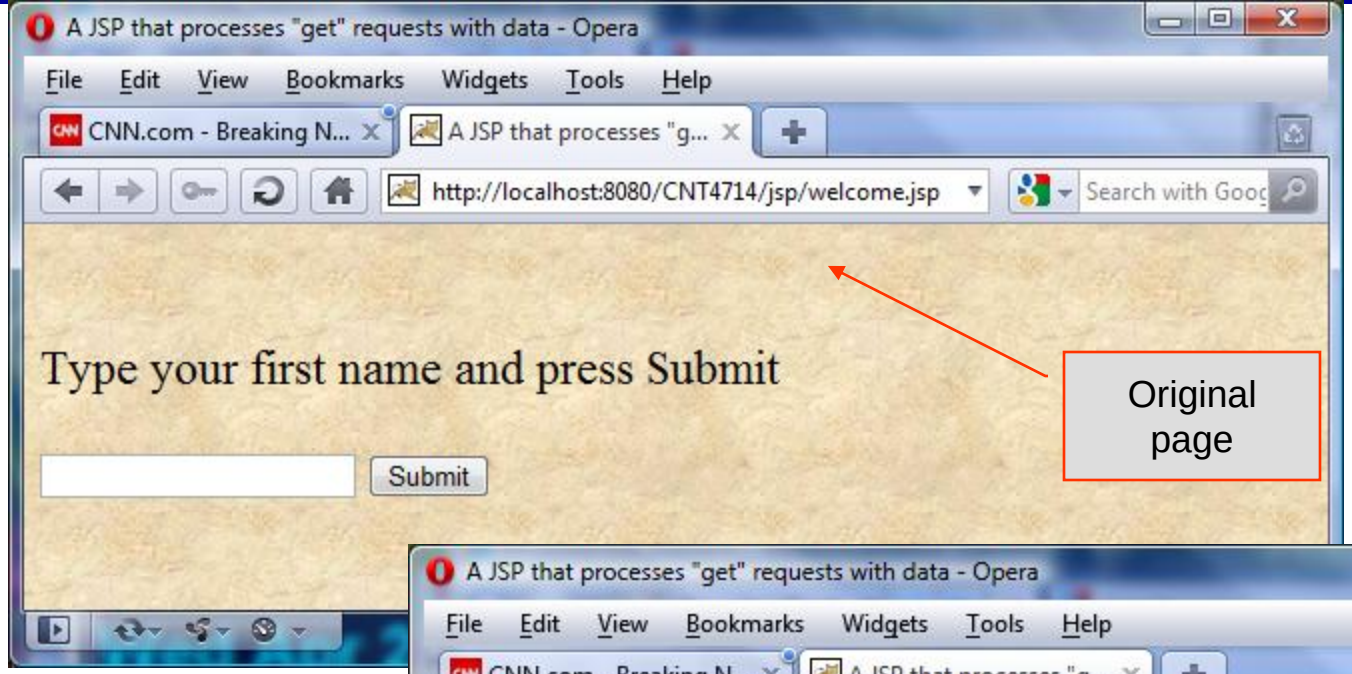
Scripting Example – welcome.jsp

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- welcome.jsp -->
<!-- JSP that processes a "get" request containing data. -->

<html xmlns = "http://www.w3.org/1999/xhtml">

   <!-- head section of document -->
   <head>
      <title>A JSP that processes "get" requests with data</title>
   </head>

   <!-- body section of document -->
   <body>
      <% // begin scriptlet
         String name = request.getParameter( "firstName" );
         if ( name != null )
         {
      %> <%-- end scriptlet to insert fixed template data --%>
```

XHTML comments shown in blue.

Scriplets shown in green.

```
      <h1>
               Hello <%= name %>, <br />
               Welcome to JavaServer Pages Technology!
            </h1>

      <% // continue scriptlet

         } // end if
         else {

      %> <%-- end scriptlet to insert fixed template data --%>

            <form action = "welcome.jsp" method = "get">
               <p>Type your first name and press Submit</p>

               <p><input type = "text" name = "firstName" />
                  <input type = "submit" value = "Submit" />
               </p>
            </form>

      <% // continue scriptlet
          } // end else
      %> <%-- end scriptlet --%>
   </body>

</html>  <!-- end XHTML document -->
```

Original page

Execution of JSP