Trabalhando com JavaServer Pages (JSP)

7.2.1 Introduction

- JavaServer Pages
  - Extension of Servlet technology
- Web content delivery
- Reuse existing Java components
  - Without programming Java
- Create custom tags
  - Encapsulate complex functionality
- Classes and interfaces specific to JSP
  - Package `javax.servlet.jsp`
  - Package `javax.servlet.jsp.tagext`

7.2.2 JavaServer Pages Overview

- Key components
  - Directives
  - Actions
  - Scriptlets
  - Tag libraries

7.2.2 JavaServer Pages Overview (cont.)

- Directive
  - Message to JSP container
    - i.e., program that compiles/executes JSPs
  - Enable programmers to specify
    - Page settings
    - Content to include from other resources
    - Custom tag libraries used in the JSP

7.2.2 JavaServer Pages Overview (cont.)

- Action
  - Predefined JSP tags that encapsulate functionality
  - Often performed based on information from client request
  - Can be used to create Java objects for use in scriptlets

7.2.2 JavaServer Pages Overview (cont.)

- Scriptlet
  - Also called “Scripting Elements”
  - Enable programmers to insert Java code in JSPs
  - Performs request processing
    - Interacts with page elements and other components to implement dynamic pages

7.2.2 JavaServer Pages Overview (cont.)

- JSPs
  - Look like standard HTML or XHTML
    - Normally include HTML or XHTML markup
      - Known as fixed-template data
  - Used when content is mostly fixed-template data
    - Small amounts of content generated dynamically
- Servlets
  - Used when small amount of content is fixed-template data
    - Most content generated dynamically

7.2.2 JavaServer Pages Overview (cont.)

- Some servlets do not produce content
  - Invoke other servlets and JSPs
- JSPs execute as part of a Web server
  - JSP container
- JSP first request
  - JSP container translates a JSP into a servlet
    - Handle the current and future requests
- Code that represents the JSP
  - Placed in servlet's `_jspService` method

7.2.2 JavaServer Pages Overview (cont.)

- JSP errors
  - Translation-time errors
    - Occur when JSPs are translated into servlets
  - Request-time errors
    - Occur during request processing
- Methods `jspInit` and `jspDestroy`
  - Container invokes when initializing and terminating a JSP
- Methods are defined in JSP declarations
  - Part of the JSP scripting mechanism

7.2.3 A First JavaServer Page Example

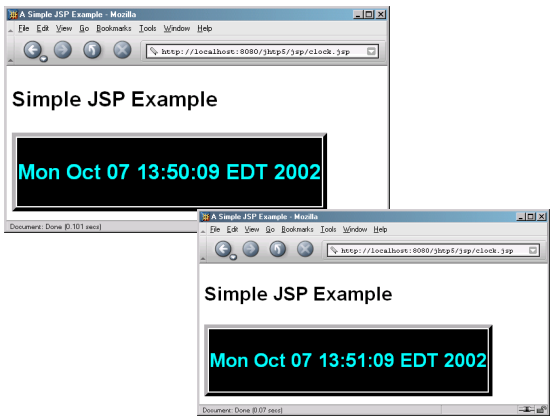
- Simple JSP example (Fig. 7.2.1)
  - Demonstrates
    - Fixed-template data (XHTML markup)
    - Creating a Java object (`java.util.Date`)
    - Automatic conversion of JSP expression to a `String`
    - `meta` element to refresh Web page at specified interval
  - First invocation of `clock.jsp`
    - Notice the delay while:
      - JSP container translates the JSP into a servlet
      - JSP container compiles the servlet
      - JSP container executes the servlet
    - Subsequent invocations should not experience the same delay


```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 7.2.1: clock.jsp -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8
9     <head>
10       <meta http-equiv = "refresh" content = "60" />
11
12       <title>A Simple JSP Example</title>
13
14       <style type = "text/css">
15          .big { font-family: helvetica, arial, sans-serif;
16                 font-weight: bold;
17                 font-size: 2em; }
18       </style>
19    </head>
20
21    <body>
22       <p class = "big">Simple JSP Example</p>
23
```

`meta` element refreshes the Web page every 60 seconds


```
24       <table style = "border: 6px outset;">
25          <tr>
26             <td style = "background-color: black;">
27                <p class = "big" style = "color: cyan;">
28
29                   <!-- JSP expression to insert date/time -->
30                   <%= new java.util.Date() %>
31
32                </p>
33             </td>
34          </tr>
35       </table>
36    </body>
37
38 </html>
```

Creates `Date` object that is converted to a `String` implicitly and displayed in paragraph (`p`) element



7.2.4 Implicit Objects

- Implicit Objects
  - Provide access to many servlet capabilities within a JSP
  - Four scopes
    - Application scope
      - Objects owned by the container application
      - Any servlet or JSP can manipulate these objects
    - Page scope
      - Objects that exist only in page in which they are defined
      - Each page has its own instance of these objects
    - Request scope
      - Objects exist for duration of client request
      - Objects go out of scope after response sent to client
    - Session scope
      - Objects exist for duration of client's browsing session
      - Objects go out of scope when client terminates session or when session timeout occurs

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 discussed in this table.
`response`	This object represents the response to the client and is normally 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.

Fig. 25.2 JSP implicit objects.

7.2.5 Scripting

- Scripting
  - Dynamically generated content
  - Insert Java code and logic in JSP using scripting

7.2.5.1 Scripting Components

- JSP scripting components
  - Scriptlets (delimited by <% and %>)
  - Comments
    - JSP comments (delimited by <%-- and --%>)
    - XHTML comments (delimited by <!-- and -->)
    - Java's comments (delimited by // and /* and */)
  - Expressions (delimited by <%= and %>)
  - Declarations (delimited by <%! And %>)
  - Escape sequences

7.2.5.1 Scripting Components (cont.)

Literal	Escape sequence	Description
<%	<\%	The character sequence <% normally indicates the beginning of a scriptlet. The <\% escape sequence places the literal characters <% in the response to the client.
%>	%\>	The character sequence %> normally indicates the end of a scriptlet. The %\> escape sequence places the literal characters %> in the response to the client.
' " \	\' \" \\	As with string literals in a Java program, the escape sequences for characters ', " and \ allow these characters to appear in attribute values. Remember that the literal text in a JSP becomes string literals in the servlet that represents the translated JSP.

Fig. 7.2.3 JSP escape sequences.

7.2.5.2 Scripting Example

- Demonstrate basic scripting capabilities
  - Responding to `get` requests


```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 7.2.4: welcome.jsp -->
6  <!-- JSP that processes a "get" request containing data. -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9
10    <!-- head section of document -->
11    <head>
12       <title>Processing "get" requests with data</title>
13    </head>
14
15    <!-- body section of document -->
16    <body>
17       <% // begin scriptlet
18
19          String name = request.getParamet
20
21          if ( name != null ) {
22
23       %> <%-- end scriptlet to insert fixe
24
```

Scriptlets used to insert Java code

Use `request` implicit object to get parameter


```
25             <h1>
26                Hello <%= name %>, <br />
27                Welcome to JavaServer Pages!
28             </h1>
29
30       <% // continue scriptlet
31
32          }  // end if
33          else {
34
35       %> <%-- end scriptlet to insert fixed template data --%>
36
37             <form action = "welcome.jsp" method = "get">
38                <p>Type your first name and press Submit</p>
39
40                <p><input type = "text" name = "firstName" />
41                   <input type = "submit" value = "Submit" />
42                </p>
43             </form>
44
45       <% // continue scriptlet
46
47          }  // end else
48
49       %> <%-- end scriptlet --%>
50    </body>
51
52 </html>  <!-- end XHTML document -->
```

JSP expression

Scriptlets used to insert Java code

7.2.6 Standard Actions

- JSP standard actions
  - Provide access to common tasks performed in a JSP
    - Including content from other resources
    - Forwarding requests to other resources
    - Interacting with JavaBeans
  - JSP containers process actions at request time
  - Delimited by `<jsp:`*action*`>` and `</jsp:`*action*`>`

7.2.6 Standard Actions

Action	Description
`<jsp:include>`	Dynamically includes another resource in a JSP. As the JSP executes, the referenced resource is included and processed.
`<jsp:forward>`	Forwards request processing to another JSP, servlet or static page. This action terminates the current JSP's execution.
`<jsp:plugin>`	Allows a plug-in component to be added to a page in the form of a browser-specific `object` or `embed` HTML element. In the case of a Java applet, this action enables the downloading and installation of the *Java Plug-in*, if it is not already installed on the client computer.
`<jsp:param>`	Used with the `include`, `forward` and `plugin` actions to specify additional name/value pairs of information for use by these actions.
JavaBean Manipulation	
`<jsp:useBean>`	Specifies that the JSP uses a JavaBean instance. This action specifies the scope of the bean and assigns it an ID that scripting components can use to manipulate the bean.
`<jsp:setProperty>`	Sets a property in the specified JavaBean instance. A special feature of this action is automatic matching of request parameters to bean properties of the same name.
`<jsp:getProperty>`	Gets a property in the specified JavaBean instance and converts the result to a string for output in the response.

Fig. 25.5 JSP standard actions.

7.2.6.1 `<jsp:include>` Action

- `<jsp:include>` action
  - Enables content to be included in a JSP dynamically
  - More flexible than `include` directive
    - Requires more overhead when page contents change frequently

7.2.6.1 <jsp:include> Action (cont.)

Attribute	Description
page	Specifies the relative URI path of the resource to include. The resource must be part of the same Web application.
flush	Specifies whether the buffer should be flushed after the include is performed. In JSP 1.1, this attribute is required to be true.

Fig. 25.6 Action <jsp:include> attributes.

Exemplo: include.jsp


```
<!-- banner to include in another document -->
<div style = "width: 580px">
   <p>
      Programação Orientada a Objetos <br /> Internet and
      World Wide Web Programming Training in Java<br />
      ITA - Stefanini
   </p>

   <p>
      <a href = "mailto:pauloac@ita.br">pauloac@ita.br</a>

   </p>
</div>
```

banner.html


```
<!-- contents to include in another document -->

<p><a href = "http://www.comp.ita.br/~pauloac/">
   Home Page
</a></p>

<p><a href = "http://www.ita.com.br">
ITA</a></p>

<p><a href = "http://www.stefanini.com.br">
   Stefanini
</a></p>


<p>Send questions or comments about this site to
   <a href = "mailto:pauloac@ita.br">
      pauloac@ita.br   </a><br />
   Copyright  2005 </p>
```

toc.html


```
1    <!-- Fig. 7.2.9: clock2.jsp                    -->
2    <!-- date and time to include in another document -->
3
4    <table>
5       <tr>
6          <td style = "background-color: black;">
7             <p class = "big" style = "color: cyan; font-size: 3em;
8                font-weight: bold;">
9
10               <%-- script to determine client local and --%>
11               <%-- format date accordingly              --%>
12               <%
13                  // get client locale
14                  java.util.Locale locale = request.getLocale();
15
16                  // get DateFormat for client's Locale
17                  java.text.DateFormat dateFormat =
18                     java.text.DateFormat.getDateTimeInstance(
19                        java.text.DateFormat.LONG,
20                        java.text.DateFormat.LONG, locale );
21
22               %>  <%-- end script --%>
23
24               <%-- output date --%>
25               <%= dateFormat.format( new java.util.Date() ) %>
26            </p>
27         </td>
28      </tr>
29   </table>
```

Use `Locale` to format `Date` with specified `DateFormat`

clock2.jsp


```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 7.2.10: include.jsp -->
6
7   <html xmlns = "http://www.w3.org/1999/xhtml">
8
9      <head>
10        <title>Using jsp:include</title>
11
12        <style type = "text/css">
13           body {
14              font-family: tahoma, helvetica, arial, sans-serif;
15           }
16
17           table, tr, td {
18              font-size: .9em;
19              border: 3px groove;
20              padding: 5px;
21              background-color: #dddddd;
22           }
23        </style>
24     </head>
25
```



```
26     <body>
27        <table>
28           <tr>
29              <td style = "width: 160px; text-align: center">
30                 <img src = "images/logotiny.png"
31                    width = "140" height = "93"
32                    alt = "Deitel & Associates, Inc. Logo" />
33              </td>
34
35              <td>
36
37                 <%-- include banner.html in this JSP --%>
38                 <jsp:include page = "banner.html"
39                    flush = "true" />
40
41              </td>
42           </tr>
43
44           <tr>
45              <td style = "width: 160px">
46
47                 <%-- include toc.html in this JSP --%>
48                 <jsp:include page = "toc.html" flush = "true" />
49
50              </td>
51
```

Use JSP action to include `banner.html`

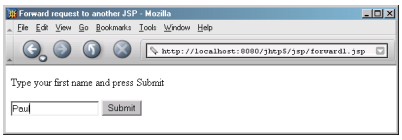
Use JSP action to include `toc.html`

```
52            <td style = "vertical-align: top">
53
54               <%-- include clock2.jsp in this JSP --%>
55               <jsp:include page = "clock2.jsp"
56                  flush = "true" />
57
58            </td>
59         </tr>
60      </table>
61   </body>
62 </html>
```

Use JSP action to include `clock2.jsp`

7.2.6.2 <jsp:forward> Action

- `<jsp:forward>` action
  - Enables JSP to forward request to different resources
    - Can forwarded requests only resources in same context
- `<jsp:param>` action
  - Specifies name/value pairs of information
    - Name/Value pairs are passed to other actions




```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 7.2.11: forward1.jsp -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8
9  <head>
10    <title>Forward request to another JSP</title>
11 </head>
12
13 <body>
14    <% // begin scriptlet
15
16       String name = request.getParameter( "firstName" );
17
18       if ( name != null ) {
19
20    %> <%-- end scriptlet to insert fixed template data --%>
21
22       <jsp:forward page = "forward2.jsp">
23          <jsp:param name = "date"
24             value = "<%= new java.util.Date() %>" />
25       </jsp:forward>
26
```

Forward request to `forward2.jsp`


```
27    <% // continue scriptlet
28
29       } // end if
30       else {
31
32    %> <%-- end scriptlet to insert fixed template data --%>
33
34          <form action = "forward1.jsp" method = "get">
35             <p>Type your first name and press Submit</p>
36
37             <p><input type = "text" name = "firstName" />
38                <input type = "submit" value = "Submit" />
39             </p>
40          </form>
41
42    <%  // continue scriptlet
43
44       } // end else
45
46    %> <%-- end scriptlet --%>
47 </body>
48
49 </html>  <!-- end XHTML document -->
```

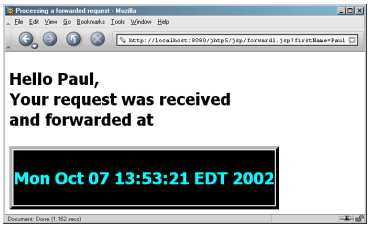


```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- forward2.jsp -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml"v
8
9  <head>
10    <title>Processing a forwarded request</title>
11
12    <style type = "text/css">
13       .big {
14          font-family: tahoma, helvetica, arial, sans-serif;
15          font-weight: bold;
16          font-size: 2em;
17       }
18    </style>
19 </head>
20
21 <body>
22    <p class = "big">
23       Hello <%= request.getParameter( "firstName" ) %>, <br />
24       Your request was received <br /> and forwarded at
25    </p>
26
```

Receive request from `forward1.jsp`, then get `firstName` parameter from request


```
27    <table style = "border: 6px outset;">
28       <tr>
29          <td style = "background-color: black;">
30             <p class = "big" style = "color: cyan;">
31                <%= request.getParameter( "date" ) %>
32             </p>
33          </td>
34       </tr>
35    </table>
36 </body>
37
38 </html>
```

Get `date` parameter from request



7.2.7 Directives

- JSP directives
  - Messages to JSP container
  - Enable programmer to:
    - Specify page settings
    - Include content from other resources
    - Specify custom-tag libraries
  - Delimited by <%@ and %>

7.2.7 Directives (cont.)

Directive	Description
`page`	Defines page settings for the JSP container to process.
`include`	Causes the JSP container to perform a translation-time insertion of another resource's content. As the JSP is translated into a servlet and compiled, the referenced file replaces the `include` directive and is translated as if it were originally part of the JSP.
`taglib`	Allows programmers to define new tags in the form of *tag libraries*, which can be used to encapsulate functionality and simplify the coding of a JSP.

Fig. 25.17 JSP directives.

7.2.7.1 page Directive

- JSP `page` directive
  - Specifies JSP's global settings in JSP container

7.2.7.1 page Directive (cont.)

Attribute	Description
`language`	The scripting language used in the JSP. Currently, the only valid value for this attribute is `java`.
`extends`	Specifies the class from which the translated JSP will be inherited. This attribute must be a fully qualified class name.
`import`	Specifies a comma-separated list of fully qualified type names and/or packages that will be used in the current JSP. When the scripting language is `java`, the default import list is `java.lang.*`, `javax.servlet.*`, `javax.servlet.jsp.*` and `javax.servlet.http.*`. If multiple `import` properties are specified, the package names are placed in a list by the container.
`session`	Specifies whether the page participates in a session. The values for this attribute are `true` (participates in a session—the default) or `false` (does not participate in a session). When the page is part of a session, implicit object `session` is available for use in the page. Otherwise, `session` is not available, and using `session` in the scripting code results in a translation-time error.

Fig. 25.18 Attributes of the `page` directive.

7.2.7.1 page Directive (cont.)

Attribute	Description
`buffer`	Specifies the size of the output buffer used with the implicit object `out`. The value of this attribute can be `none` for no buffering, or a value such as `8kb` (the default buffer size). The JSP specification indicates that the buffer used must be at least the size specified.
`autoFlush`	When set to `true` (the default), this attribute indicates that the output buffer used with implicit object `out` should be flushed automatically when the buffer fills. If set to `false`, an exception occurs if the buffer overflows. This attribute's value must be `true` if the buffer attribute is set to `none`.
`isThreadSafe`	Specifies if the page is thread safe. If `true` (the default), the page is considered to be thread safe, and it can process multiple requests at the same time. If `false`, the servlet that represents the page implements interface `java.lang.SingleThreadModel` and only one request can be processed by that JSP at a time. The JSP standard allows multiple instances of a JSP to exists for JSPs that are not thread safe. This enables the container to handle requests more efficiently. However, this does not guarantee that resources shared across JSP instances are accessed in a thread-safe manner.
`info`	Specifies an information string that describes the page. This string is returned by the `getServletInfo` method of the servlet that represents the translated JSP. This method can be invoked through the JSP's implicit `page` object.

Fig. 7.2.18 Attributes of the `page` directive.

7.2.7.1 page Directive (cont.)

Attribute	Description
`errorPage`	Any exceptions in the current page that are not caught are sent to the error page for processing. The error page implicit object `exception` references the original exception.
`isErrorPage`	Specifies if the current page is an error page that will be invoked in response to an error on another page. If the attribute value is `true`, the implicit object `exception` is created and references the original exception that occurred. If `false` (the default), any use of the `exception` object in the page results in a translation-time error.
`contentType`	Specifies the MIME type of the data in the response to the client. The default type is `text/html`.

Fig. 25.18 Attributes of the `page` directive.

Exercício

- Criar um programa JSP que liste o nome dos autores cadastrados no banco de dados: books.fdb
  - Utilize o programa exemplo FirstJDBCProgram como base

7.2.9 Referências sobre JSP

- JSP at Sun Microsystems
  - java.sun.com/products/jsp
- Servlets at Sun Microsystems
  - java.sun.com/products/servlet
- J2EE at Sun Microsystems
  - java.sun.com/j2ee
- Core Servlets and JavaServer Pages, Martin Hall, Sun Press