Chapter 10: JavaServer Pages

Outline

Chapter 10: JavaServer Pages

10.1 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`**

10.2 JavaServer Pages Overview

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

10.2 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

10.2 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

10.2 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

10.2 JavaServer Pages Overview (cont.)

- Custom Tag Library
  - JSP's tag extension mechanism
  - Enables programmers to define new tags
    - Tags encapsulate complex functionality
  - Tags can manipulate JSP content

10.2 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

10.2 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

10.2 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

10.3 A First JavaServer Page Example

- Simple JSP example (Fig. 10.1)
  - 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. 10.1: 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
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>
```

Outline

Fig. 10.1 Using a JSP expression to insert the date and time in a Web page (Part 1).

Line 10

meta element refreshes the Web page every **60** seconds

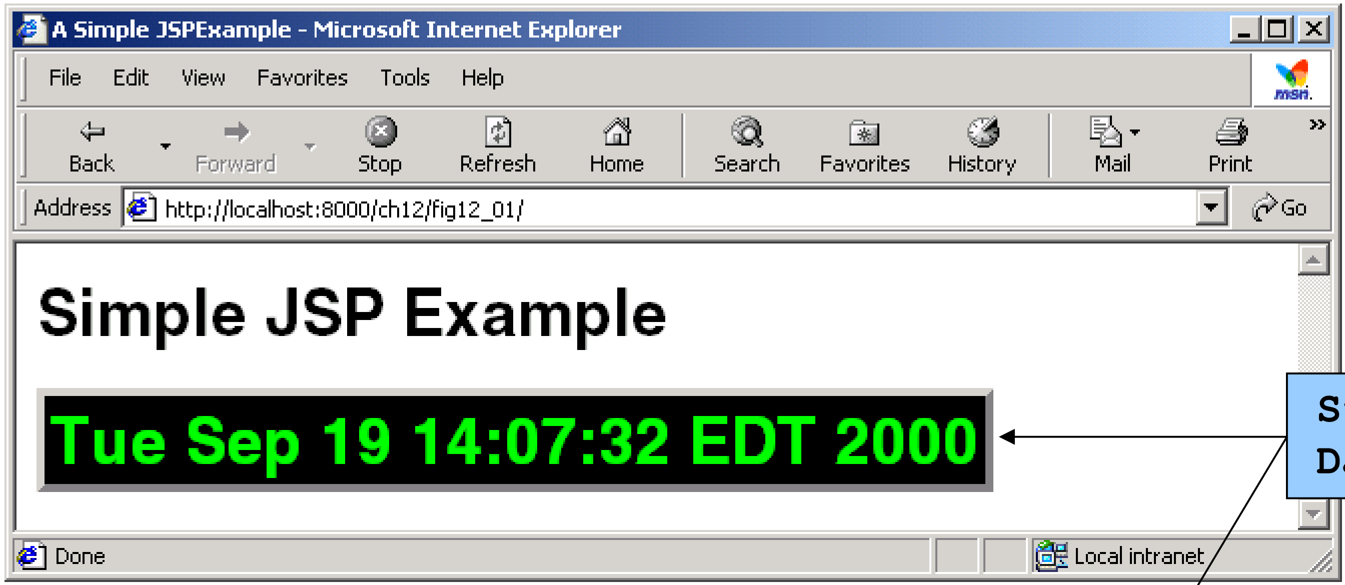
Creates **Date** object that is converted to a **String** implicitly and displayed in paragraph (**p**) element

```
      </body>

   </html>
```

Outline

Fig. 10.1 Using a JSP expression to insert the date and time in a Web page (Part 2).

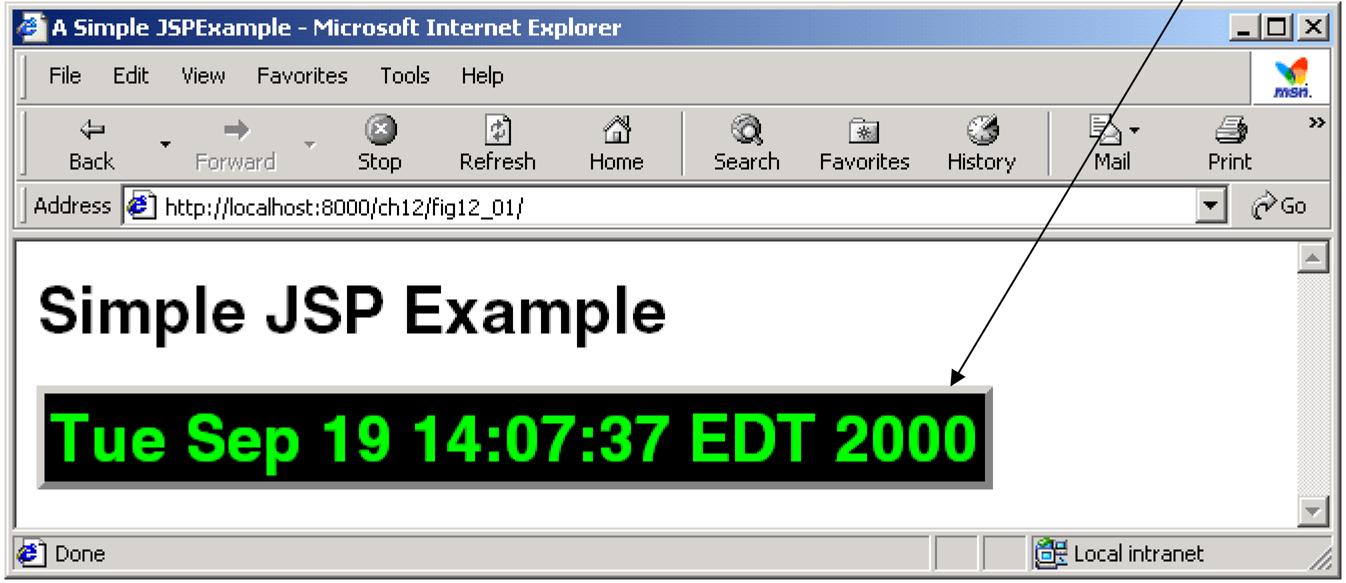
Program Output


A Simple JSPExample - Microsoft Internet Explorer

http://localhost:8000/ch12/fig12_01/

Simple JSP Example

Tue Sep 19 14:07:32 EDT 2000


A Simple JSPExample - Microsoft Internet Explorer

http://localhost:8000/ch12/fig12_01/

Simple JSP Example

Tue Sep 19 14:07:37 EDT 2000

`String` representation of **`Date`** object appears here.

10.4 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 when 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

10.4 Implicit Objects (cont.)

- JSP implicit objects
  - Extend classes or implement interfaces
    - Discussed in Chapter 9
  - Such objects can invoke public aspects of classes/interfaces

10.4 Implicit Objects (cont.)

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.

Fig. 10.2 JSP implicit objects (part 1 of 2).

10.4 Implicit Objects (cont.)

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.

Fig. 10.2 JSP implicit objects (part 2 of 2).

10.5 Scripting

- Scripting
  - How JSP programmers can insert Java code and logic
  - Currently, JSP support scripting only with Java

10.5.1 Scripting Components

- JSP scripting components
  - Scriptlets (delimited by **`<%`** and **`%>`**)
  - Comments (delimited by **`<%--`** and **`--%>`**)
  - Expressions (delimited by **`<%=`** and **`%>`**)
  - Declarations
  - Escape sequences

10.5.1 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. 10.3 JSP escape sequences.

10.5.2 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. 10.4: 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.getParameter( "firstName" );
20
21            if ( name != null ) {
22
23         %> <%-- end scriptlet to insert fixed template data --%>
24
25               <h1>
26                  Hello <%= name %>, <br />
27                  Welcome to JavaServer Pages!
28               </h1>
29
30         <% // continue scriptlet
31
32            }  // end if
```

Outline

Fig. 10.4 Scripting a JavaServer Page -- **`welcome.jsp`** (Part 1).

Lines 17-23, 30-35

Line 19

Scriptlets used to insert Java code

Use **`request`** implicit object to get parameter

JSP declaration

```
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 -->
```

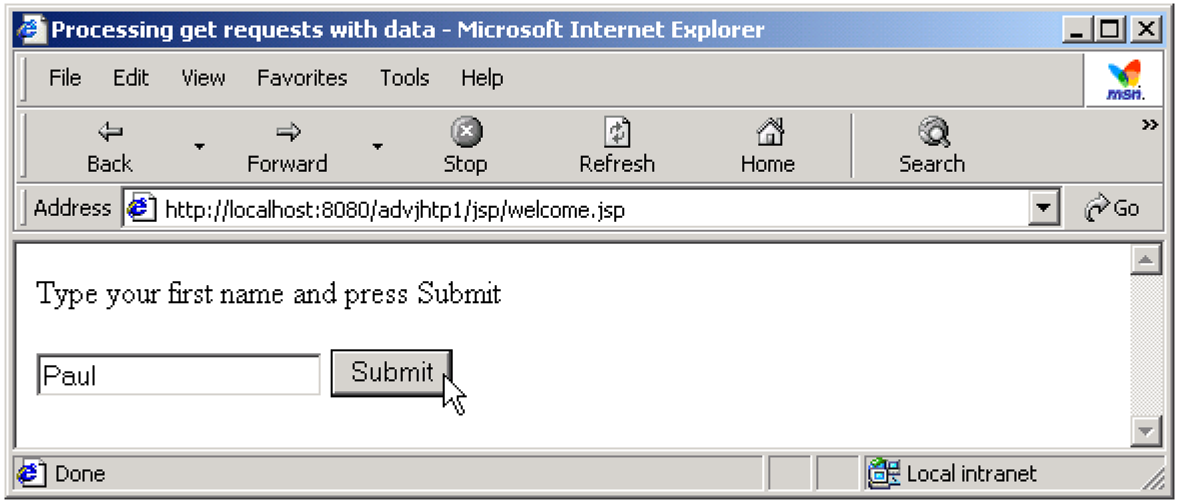
Outline

Fig. 10.4 Scripting a JavaServer Page -- **`welcome.jsp`** (Part 2).

Lines 45-49

Scriptlets used to insert Java code

Outline

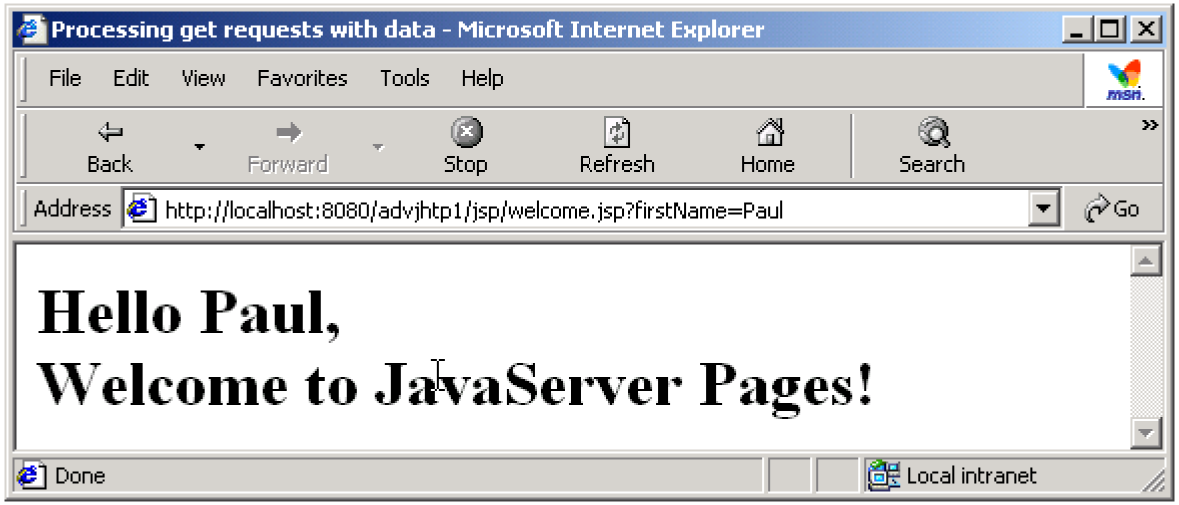

Processing get requests with data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address
Go
Done
Local intranet

http://localhost:8080/advjhtp1/jsp/welcome.jsp

Type your first name and press Submit

Paul

Submit


Processing get requests with data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address
Go
Done
Local intranet

http://localhost:8080/advjhtp1/jsp/welcome.jsp?firstName=Paul

Hello Paul,
Welcome to JavaServer Pages!

Fig. 10.4 Scripting a JavaServer Page -- **`welcome.jsp`** (Part 3).

10.6 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***`>`**

10.6 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.

Fig. 10.5 JSP standard actions (part 1 of 2).

10.6 Standard Actions (cont.)

Action	Description
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. 10.5 JSP standard actions (part 2 of 2).

10.6.1 `<jsp:include>` Action

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

10.6.1 `<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. 10.6 Action **`<jsp:include>`** attributes.

```
<!-- Fig. 10.7: banner.html                   -->
<!-- banner to include in another document -->
<div style = "width: 580px">
   <p>
      Java(TM), C, C++, Visual Basic(R),
      Object Technology, and <br /> Internet and
      World Wide Web Programming Training&nbsp;<br />
      On-Site Seminars Delivered Worldwide
   </p>

   <p>
      <a href = "mailto:deitel@deitel.com">
         deitel@deitel.com</a><br />

      978.579.9911<br />
      490B Boston Post Road, Suite 200,
      Sudbury, MA 01776
   </p>
</div>
```

Fig. 10.7 Banner (**`banner.html`**) to include across the top of the XHTML document created by Fig. 10.10.

```
<!-- Fig. 10.8: toc.html                      -->
<!-- contents to include in another document -->

<p><a href = "http://www.deitel.com/books/index.html">
   Publications/BookStore
</a></p>

<p><a href = "http://www.deitel.com/whatsnew.html">
   What's New
</a></p>

<p><a href = "http://www.deitel.com/books/downloads.html">
   Downloads/Resources
</a></p>

<p><a href = "http://www.deitel.com/faq/index.html">
   FAQ (Frequently Asked Questions)
</a></p>

<p><a href = "http://www.deitel.com/intro.html">
   Who we are
</a></p>

<p><a href = "http://www.deitel.com/index.html">
   Home Page
</a></p>

<p>Send questions or comments about this site to
   <a href = "mailto:deitel@deitel.com">
      deitel@deitel.com
   </a><br />
   Copyright 1995-2002 by Deitel &amp; Associates, Inc.
   All Rights Reserved.
</p>
```

Fig. 10.8 Table of contents (**`toc.html`**) to include down the left side of the XHTML document created by Fig.10.10.

```
1    <!-- Fig. 10.9: 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>
```

Outline

Fig. 10.9 JSP **`clock2.jsp`** to include as the main content in the XHTML document created by Fig.10.10.

Lines 14-20

Use **`Locale`** to format **`Data`** with specified **`DataFormat`**

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

<!-- Fig. 10.7: include.jsp -->

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

   <head>
      <title>Using jsp:include</title>

      <style type = "text/css">
         body {
            font-family: tahoma, helvetica, arial, sans-serif;
         }

         table, tr, td {
            font-size: .9em;
            border: 3px groove;
            padding: 5px;
            background-color: #dddddd;
         }
      </style>
   </head>

   <body>
      <table>
         <tr>
            <td style = "width: 160px; text-align: center">
               <img src = "images/logotiny.png"
                  width = "140" height = "93"
                  alt = "Deitel & Associates, Inc. Logo" />
            </td>

```

Outline

Fig. 10.10 JSP **`include.jsp`** includes resources with **`<jsp:include>`** (Part 1).

```
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
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 **banner.html**

Use JSP action to include **toc.html**

Use JSP action to include **clock2.jsp**

Outline

include.jsp

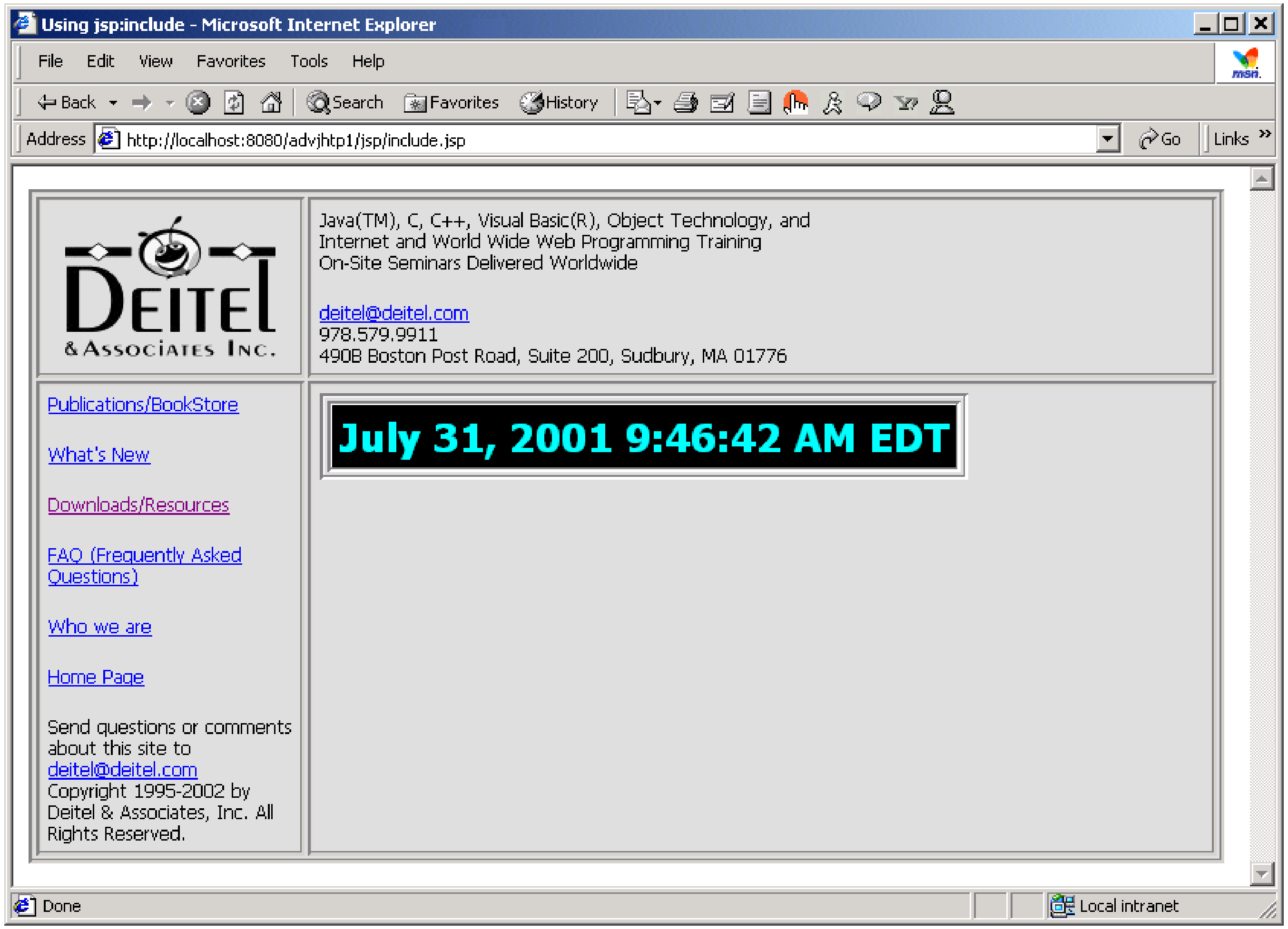
includes resources

2).

Lines 38-39

Line 48

Lines 55-56


Using jsp:include - Microsoft Internet Explorer

Address http://localhost:8080/advjhtp1/jsp/include.jsp

Java(TM), C, C++, Visual Basic(R), Object Technology, and
Internet and World Wide Web Programming Training
On-Site Seminars Delivered Worldwide

deitel@deitel.com
978.579.9911
490B Boston Post Road, Suite 200, Sudbury, MA 01776

Publications/BookStore

What's New

Downloads/Resources

FAQ (Frequently Asked Questions)

Who we are

Home Page

Send questions or comments about this site to deitel@deitel.com
Copyright 1995-2002 by Deitel & Associates, Inc. All Rights Reserved.

July 31, 2001 9:46:42 AM EDT

Done

Local intranet

Fig. 10.10 JSP `include.jsp` includes resources with `<jsp:include>` (Part 3).

10.6.2 `<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. 10.11: 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
27       <% // continue scriptlet
28
29          }  // end if
30          else {
31
32       %> <%-- end scriptlet to insert fixed template data --%>
33
```

Fig. 10.11 JSP **forward1.jsp** receives a **firstName** parameter, adds a date to the request parameters and forwards the request to **forward2.jsp** for further processing (Part 1).

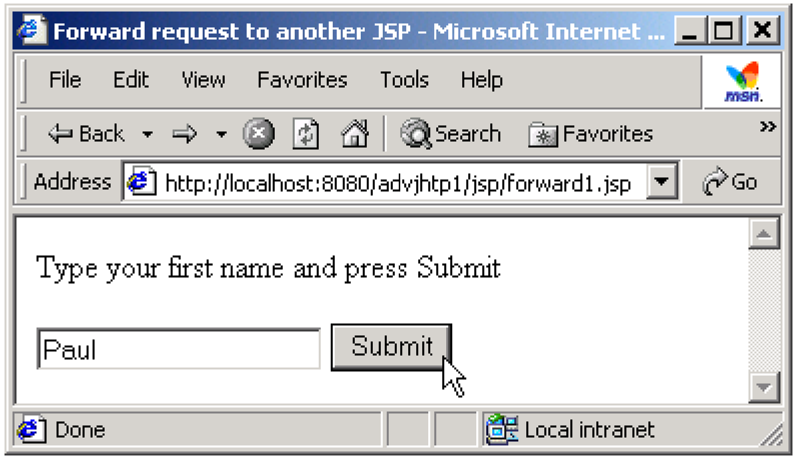
Lines 22-25

Forward request to **forward2.jsp**

```
            <form action = "forward1.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 -->
```

Outline

Fig. 10.11 JSP **`forward1.jsp`** receives a **`firstName`** parameter, adds a date to the request parameters and forwards the request to **`forward2.jsp`** for further processing (Part 2).


Forward request to another JSP - Microsoft Internet ...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/advjhtp1/jsp/forward1.jsp Go
Done Local intranet

Type your first name and press Submit

Paul Submit

Outline

Fig. 10.12 JSP **forward2.jsp** receives a request (from **forward1.jsp** in this example) and uses the request parameters as part of the response to the

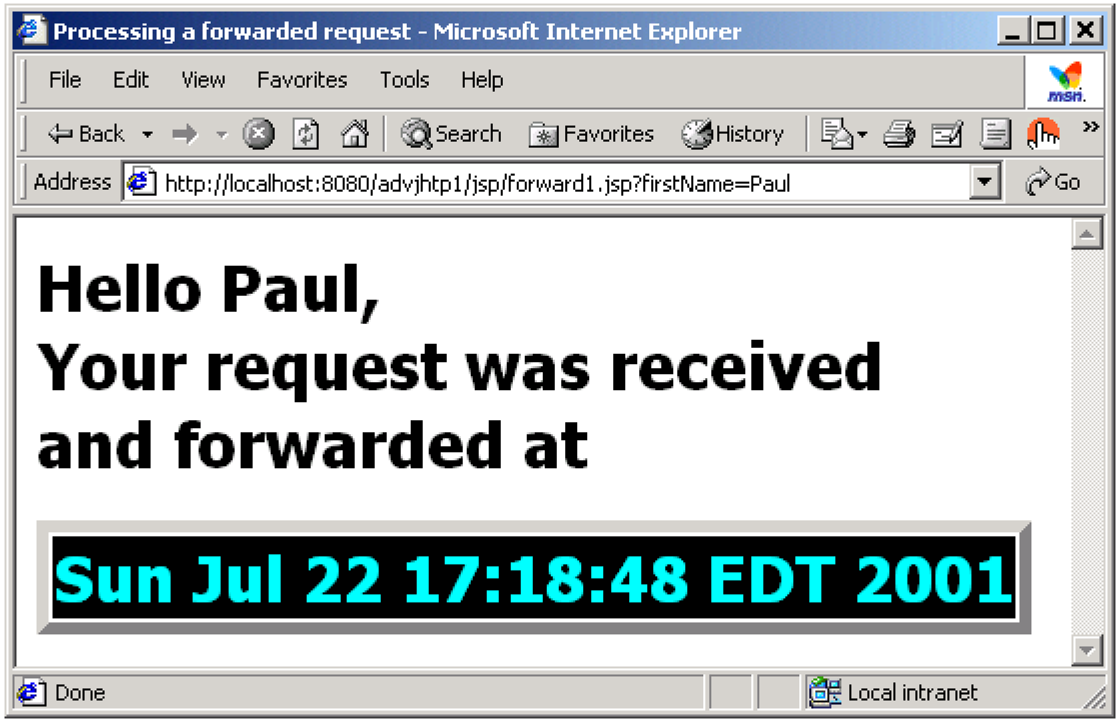
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- forward2.jsp -->

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

<head>
   <title>Processing a forwarded request</title>

   <style type = "text/css">
      .big {
         font-family: tahoma, helvetica, arial, sans-serif;
         font-weight: bold;
         font-size: 2em;
      }
   </style>
</head>

<body>
   <p class = "big">
      Hello <%= request.getParameter( "firstName" ) %>, <br />
      Your request was received <br /> and forwarded at
   </p>

   <table style = "border: 6px outset;">
      <tr>
         <td style = "background-color: black;">
            <p class = "big" style = "color: cyan;">
               <%= request.getParameter( "date" ) %>
            </p>
         </td>
      </tr>
   </table>
```

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

Get **data** parameter from request

```
</body>

</html>
```

Fig. 10.12 JSP **`forward2.jsp`** receives a request (from **`forward1.jsp`** in this example) and uses the request parameters as part of the response to the client (Part 2).


Processing a forwarded request - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/advjhtp1/jsp/forward1.jsp?firstName=Paul
Go
Done
Local intranet

Hello Paul,
Your request was received and forwarded at

Sun Jul 22 17:18:48 EDT 2001

10.6.3 `<jsp:plugin>` Action

- **`<jsp:plugin>`** action
  - Adds an applet or JavaBean to a Web page
  - Also enables client to download and install Java Plug-in

10.6.3 `<jsp:plugin>` Action (cont.)

Attribute	Description
`type`	Component type—bean or applet.
`code`	Class that represents the component.
`codebase`	Location of the class specified in the **`code`** attribute and the archives specified in the **`archive`** attribute.
`align`	Alignment of the component.
`archive`	A space-separated list of archive files that contain resources used by the component. Such an archive may include the class specified by the **`code`** attribute.
`height`	Component height in the page specified in pixels or percentage.
`hspace`	Number of pixels of space that appear to the left and to the right of the component.
`jreversion`	Version of the Java Runtime Environment and plug-in required to execute the component. The default value is 1.1.
`name`	Name of the component.
`vspace`	Number of pixels of space that appear above and below the component.
`title`	Text that describes the component.
`width`	Component width in the page specified in pixels or percentage.
`nspluginurl`	Location for download of the Java Plug-in for Netscape Navigator.
`iepluginurl`	Location for download of the Java Plug-in for Internet Explorer.

Fig. 10.13 Attributes of the **`<jsp:plugin>`** action.

```java
1     // Fig. 10.14: ShapesApplet.java
2     // Applet that demonstrates a Java2D GeneralPath.
3     package com.deitel.advjhtp1.jsp.applet;
4
5     // Java core packages
6     import java.applet.*;
7     import java.awt.event.*;
8     import java.awt.*;
9     import java.awt.geom.*;
10
11    // Java extension packages
12    import javax.swing.*;
13
14    public class ShapesApplet extends JApplet {
15
16       // initialize the applet
17       public void init()
18       {
19          // obtain color parameters from XHTML file
20          try {
21             int red = Integer.parseInt( getParameter( "red" ) );
22             int green = Integer.parseInt( getParameter( "green" ) );
23             int blue = Integer.parseInt( getParameter( "blue" ) );
24
25             Color backgroundColor = new Color( red, green, blue );
26
27             setBackground( backgroundColor );
28          }
29
30          // if there is an exception while processing the color
31          // parameters, catch it and ignore it
32          catch ( Exception exception ) {
33             // do nothing
34          }
35       }
```

Create **JApplet** to embed in JSP

Set **JApplet** background color based on parameter values

Outline

Fig. 10.14 An applet to demonstrate **<jsp:plugin>** in Fig. 10.15 (Part 1).

Line 14

Lines 21-27

```
36
37       public void paint( Graphics g )
38       {
39          // create arrays of x and y coordinates
40          int xPoints[] =
41             { 55, 67, 109, 73, 83, 55, 27, 37, 1, 43 };
42          int yPoints[] =
43             { 0, 36, 36, 54, 96, 72, 96, 54, 36, 36 };
44
45          // obtain reference to a Graphics2D object
46          Graphics2D g2d = ( Graphics2D ) g;
47
48          // create a star from a series of points
49          GeneralPath star = new GeneralPath();
50
51          // set the initial coordinate of the GeneralPath
52          star.moveTo( xPoints[ 0 ], yPoints[ 0 ] );
53
54          // create the star--this does not draw the star
55          for ( int k = 1; k < xPoints.length; k++ )
56             star.lineTo( xPoints[ k ], yPoints[ k ] );
57
58          // close the shape
59          star.closePath();
60
61          // translate the origin to (200, 200)
62          g2d.translate( 200, 200 );
63
64          // rotate around origin and draw stars in random colors
65          for ( int j = 1; j <= 20; j++ ) {
66             g2d.rotate( Math.PI / 10.0 );
67
```

Use **`GeneralPath`** to display several colored stars

Outline

Fig. 10.14 An applet to demonstrate **`<jsp:plugin>`** in Fig. 10.15 (Part 2).

Lines 40-66

```
68              g2d.setColor(
69                 new Color( ( int ) ( Math.random() * 256 ),
70                            ( int ) ( Math.random() * 256 ),
71                            ( int ) ( Math.random() * 256 ) ) );
72
73              g2d.fill( star );   // draw a filled star
74           }
75        }
76     }
```

Outline

Use **`GeneralPath`** to display several colored stars

et

`<jsp:plugin>` in Fig. 10.15 (Part 3).

Lines 68-73

```
1    <!-- Fig. 10.15: plugin.jsp -->
2
3    <html>
4
5       <head>
6          <title>Using jsp:plugin to load an applet</title>
7       </head>
8
9       <body>
10         <jsp:plugin type = "applet"
11            code = "com.deitel.advjhtp1.jsp.applet.ShapesApplet"
12            codebase = "/advjhtp1/jsp"
13            width = "400"
14            height = "400">
15
16            <jsp:params>
17               <jsp:param name = "red" value = "255" />
18               <jsp:param name = "green" value = "255" />
19               <jsp:param name = "blue" value = "0" />
20            </jsp:params>
21
22         </jsp:plugin>
23      </body>
24   </html>
```

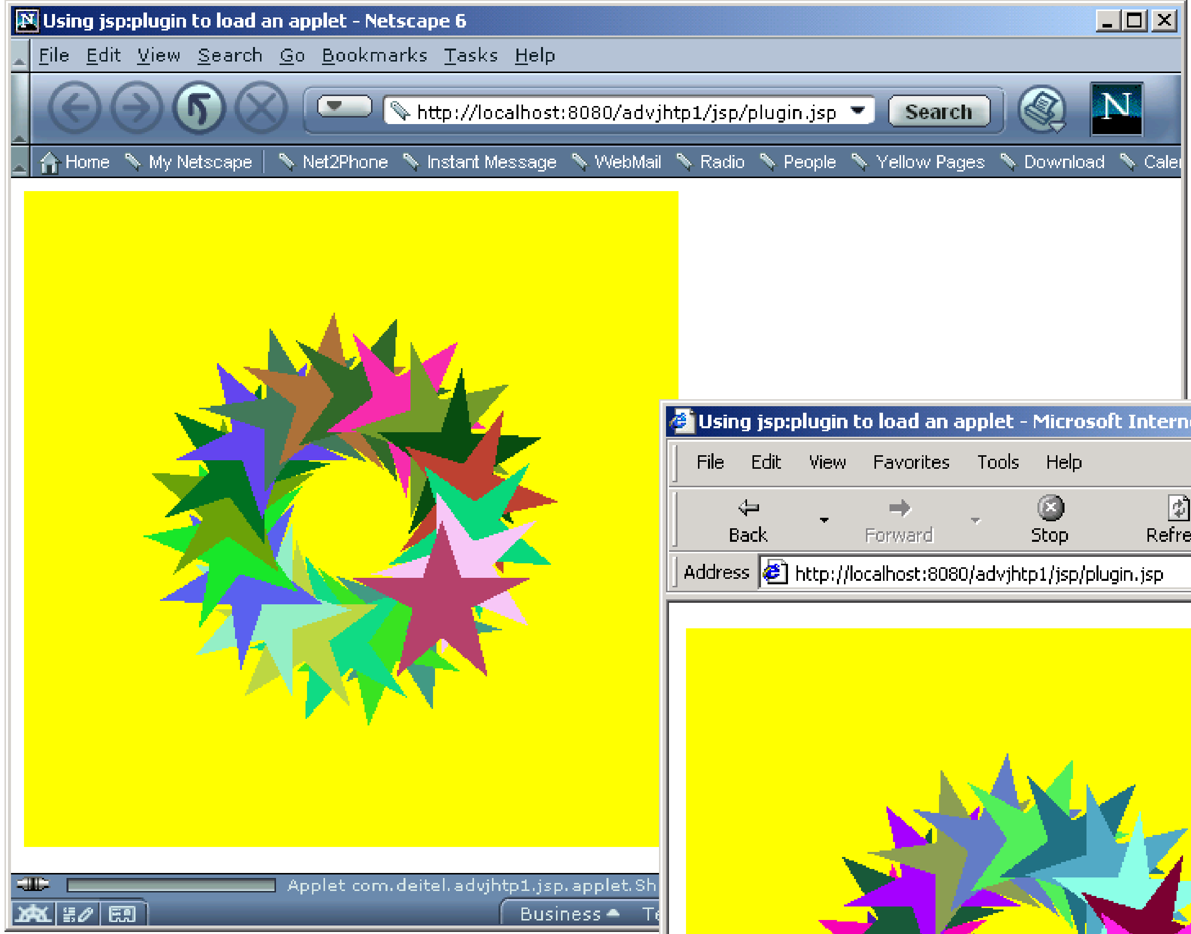
Outline

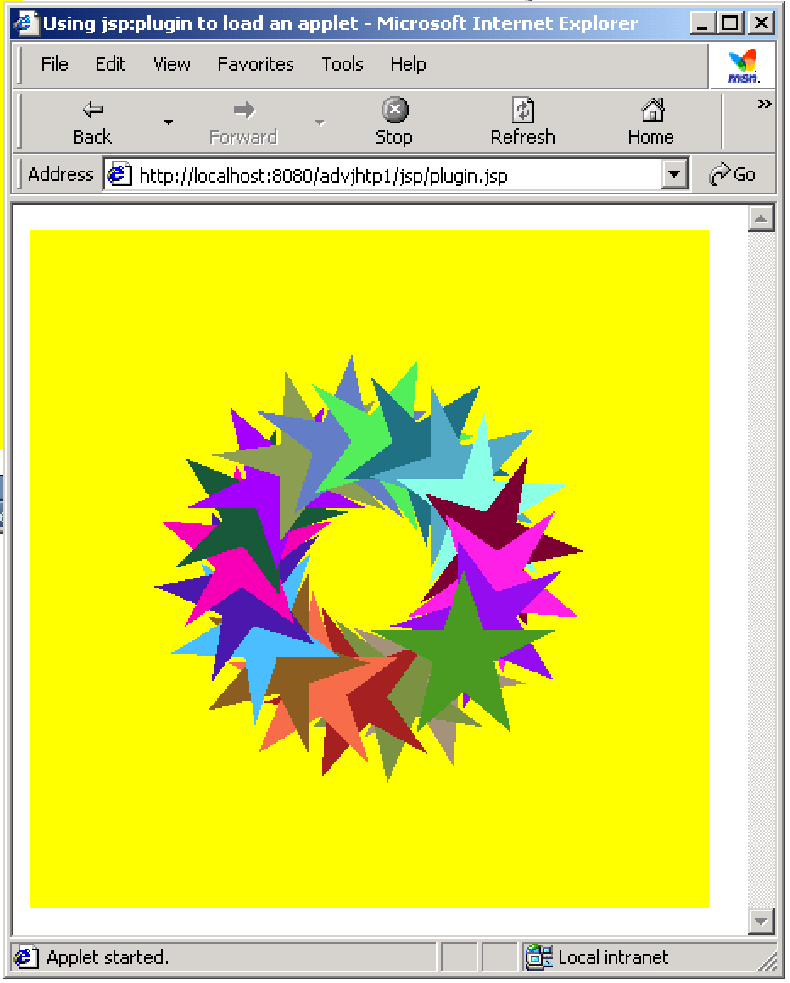
Fig. 10.15 Using
to
(Part 1).

Lines 10-22

Use **jsp:plugin** action to display **JApplet** in JSP

Use **jsp:param** action to specify **JApplet** background color


Using jsp:plugin to load an applet - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
http://localhost:8080/advjhtp1/jsp/plugin.jsp
Search
Home My Netscape Net2Phone Instant Message WebMail Radio People Yellow Pages Download
Applet com.deitel.advjhtp1.jsp.applet.Sh
Business


Using jsp:plugin to load an applet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home
Address http://localhost:8080/advjhtp1/jsp/plugin.jsp
Go
Applet started.
Local intranet

Outline

Fig. 10.15 Using **`<jsp:plugin>`** to embed a Java 2 applet in a JSP (Part 2).

10.6.4 `<jsp:useBean>` Action

- **`<jsp:useBean>`** action
  - Enables JSP to manipulate Java object
    - Creates Java object or locates an existing object for use in JSP

10.6.4 `<jsp:useBean>` Action (cont.)

Attribute	Description
`id`	The name used to manipulate the Java object with actions **`<jsp:setProperty>`** and **`<jsp:getProperty>`**. A variable of this name is also declared for use in JSP scripting elements. The name specified here is case sensitive.
`scope`	The scope in which the Java object is accessible—**`page`**, **`request`**, **`session`** or **`application`**. The default scope is **`page`**.
`class`	The fully qualified class name of the Java object.
`beanName`	The name of a bean that can be used with method **`instantiate`** of class **`java.beans.Beans`** to load a JavaBean into memory.
`type`	The type of the JavaBean. This can be the same type as the **`class`** attribute, a superclass of that type or an interface implemented by that type. The default value is the same as for attribute **`class`**. A **`ClassCastException`** occurs if the Java object is not of the type specified with attribute **`type`**.

Fig. 10.16 Attributes of the **`<jsp:useBean>`** action.

```java
1   // Fig. 10.17: Rotator.java
2   // A JavaBean that rotates advertisements.
3   package com.deitel.advjhtp1.jsp.beans;
4
5   public class Rotator {
6      private String images[] = { "images/jhtp3.jpg",
7         "images/xmlhtp1.jpg", "images/ebechtp1.jpg",
8         "images/iw3htp1.jpg", "images/cpphtp3.jpg" };
9
10     private String links[] = {
11        "http://www.amazon.com/exec/obidos/ASIN/0130125075/" +
12           "deitelassociatin",
13        "http://www.amazon.com/exec/obidos/ASIN/0130284173/" +
14           "deitelassociatin",
15        "http://www.amazon.com/exec/obidos/ASIN/013028419X/" +
16           "deitelassociatin",
17        "http://www.amazon.com/exec/obidos/ASIN/0130161438/" +
18           "deitelassociatin",
19        "http://www.amazon.com/exec/obidos/ASIN/0130895717/" +
20           "deitelassociatin" };
21
22     private int selectedIndex = 0;
23
24     // returns image file name for current ad
25     public String getImage()
26     {
27        return images[ selectedIndex ];
28     }
29
30     // returns the URL for ad's corresponding Web site
31     public String getLink()
32     {
33        return links[ selectedIndex ];
34     }
35
```

Return image file name for book cover image

Return hyperlink to book at **Amazon.com**

Outline

Fig. 10.17 Rotator bean that maintains a set of advertisements (Part 1).

Lines 25-28

Lines 31-34

```
36      // update selectedIndex so next calls to getImage and
37      // getLink return a different advertisement
38      public void nextAd()
39      {
40         selectedIndex = ( selectedIndex + 1 ) % images.length;
41      }
42   }
```

Update **`Rotator`** so subsequent calls to **`getImage`** and **`getLink`** return information for different advertisements

(Part 2).

Lines 38-41

```
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. 10.18: adrotator.jsp -->
6
7     <jsp:useBean id = "rotator" scope = "application"
8        class = "com.deitel.advjhtp1.jsp.beans.Rotator" />
9
10    <html xmlns = "http://www.w3.org/1999/xhtml">
11
12       <head>
13          <title>AdRotator Example</title>
14
15          <style type = "text/css">
16             .big { font-family: helvetica, arial, sans-serif;
17                    font-weight: bold;
18                    font-size: 2em }
19          </style>
20
21          <%-- update advertisement --%>
22          <% rotator.nextAd(); %>
23       </head>
24
25       <body>
26          <p class = "big">AdRotator Example</p>
27
28          <p>
29             <a href = "<jsp:getProperty name = "rotator"
30                property = "link" />">
31
32                <img src = "<jsp:getProperty name = "rotator"
33                   property = "image" />" alt = "advertisement" />
34             </a>
35          </p>
```

Outline

Fig. 10.18 JSP different advertisement on each request to the page (Part 1).

Lines 7-8

Line 22

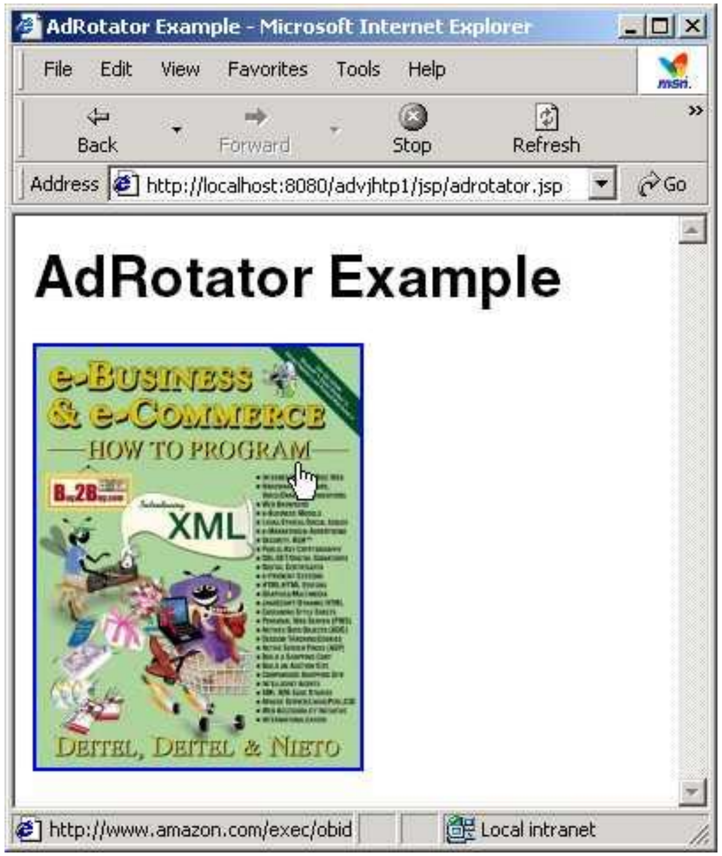
Use **jsp:useBean** action to obtain reference to **Rotator** object

Invoke **Rotator**'s **nextAd** method

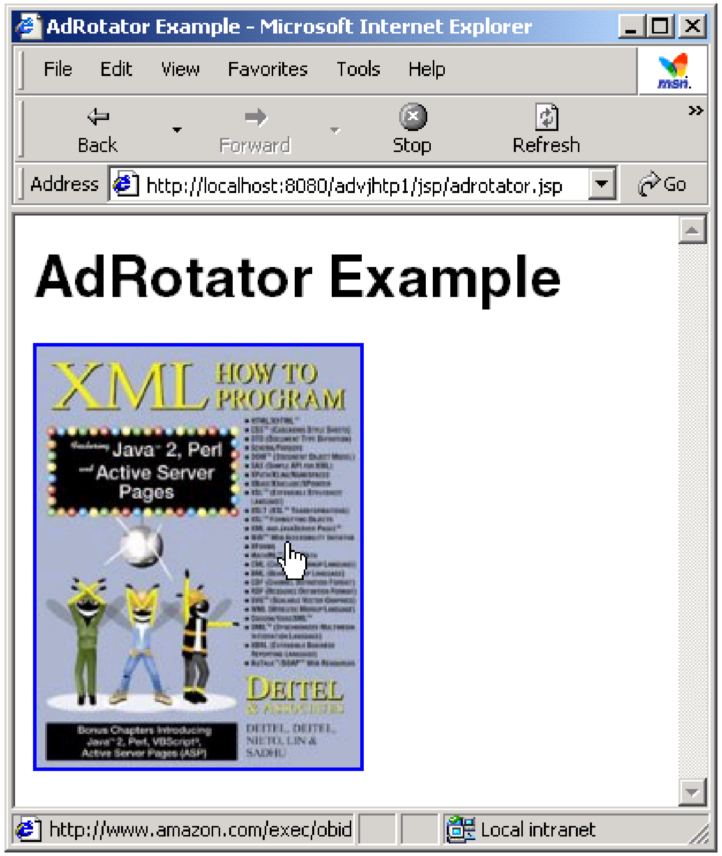
Define hyperlink to **Amazon.com** site

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

Outline


AdRotator Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address http://localhost:8080/advjhtp1/jsp/adrotator.jsp Go
http://www.amazon.com/exec/obid
Local intranet

AdRotator Example


AdRotator Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address http://localhost:8080/advjhtp1/jsp/adrotator.jsp Go
http://www.amazon.com/exec/obid
Local intranet

AdRotator Example

Fig. 10.18 JSP **`adrotator.jsp`** uses a **`Rotator`** bean to display a different advertisement on each request to the page (Part 2).

10.6.4 `<jsp:useBean>` Action (cont.)

Attribute	Description
`name`	The ID of the JavaBean for which a property (or properties) will be set.
`property`	The name of the property to set. Specifying `"*"` for this attribute causes the JSP to match the request parameters to the properties of the bean. For each request parameter that matches (i.e., the name of the request parameter is identical to the bean's property name), the corresponding property in the bean is set to the value of the parameter. If the value of the request parameter is `""`, the property value in the bean remains unchanged.
`param`	If request parameter names do not match bean property names, this attribute can be used to specify which request parameter should be used to obtain the value for a specific bean property. This attribute is optional. If this attribute is omitted, the request parameter names must match bean property names.
`value`	The value to assign to a bean property. The value typically is the result of a JSP expression. This attribute is particularly useful for setting bean properties that cannot be set using request parameters. This attribute is optional. If this attribute is omitted, the JavaBean property must be of a data type that can be set using request parameters.

Fig. 10.19 Attributes of the **`<jsp:setProperty>`** action.

```java
1   // Fig. 10.20: GuestBean.java
2   // JavaBean to store data for a guest in the guest book.
3   package com.deitel.advjhtp1.jsp.beans;
4
5   public class GuestBean {
6      private String firstName, lastName, email;
7
8      // set the guest's first name
9      public void setFirstName( String name )
10     {
11        firstName = name;
12     }
13
14     // get the guest's first name
15     public String getFirstName()
16     {
17        return firstName;
18     }
19
20     // set the guest's last name
21     public void setLastName( String name )
22     {
23        lastName = name;
24     }
25
26     // get the guest's last name
27     public String getLastName()
28     {
29        return lastName;
30     }
31
```

GuestBean defines three guest properties: **firstName**, **lastName** and **email**

Outline

Fig. 10.20 **GuestBean** stores information for one guest (Part 1).

Line 6

```
   // set the guest's email address
   public void setEmail( String address )
   {
      email = address;
   }

   // get the guest's email address
   public String getEmail()
   {
      return email;
   }
}
```

Outline

Fig. 10.20 **GuestBean** stores information for one guest (Part 2).

```
1    // Fig. 10.21: GuestDataBean.java
2    // Class GuestDataBean makes a database connection and supports
3    // inserting and retrieving data from the database.
4    package com.deitel.advjhtp1.jsp.beans;
5
6    // Java core packages
7    import java.io.*;
8    import java.sql.*;
9    import java.util.*;
10
11   public class GuestDataBean {
12      private Connection connection;
13      private PreparedStatement addRecord, getRecords;
14
15      // construct TitlesBean object
16      public GuestDataBean() throws Exception
17      {
18         // load the Cloudscape driver
19         Class.forName( "COM.cloudscape.core.RmiJdbcDriver" );
20
21         // connect to the database
22         connection = DriverManager.getConnection(
23            "jdbc:rmi:jdbc:cloudscape:guestbook" );
24
25         getRecords =
26            connection.prepareStatement(
27               "SELECT firstName, lastName, email FROM guests"
28            );
29
30         addRecord =
31            connection.prepareStatement(
32               "INSERT INTO guests ( " +
33                  "firstName, lastName, email ) " +
34               "VALUES ( ?, ?, ? )"
35            );
```

GuestDataBean connects to **guestbook** database

Outline

Fig. 10.21 GuestDataBean performs database access on behalf of **guestBook-Login.jsp** (Part 1).

Lines 22-23

```
    }

    // return an ArrayList of GuestBeans
    public ArrayList getGuestList() throws SQLException
    {
       ArrayList guestList = new ArrayList();

       // obtain list of titles
       ResultSet results = getRecords.executeQuery();

       // get row data
       while ( results.next() ) {
          GuestBean guest = new GuestBean();

          guest.setFirstName( results.getString( 1 ) );
          guest.setLastName( results.getString( 2 ) );
          guest.setEmail( results.getString( 3 ) );

          guestList.add( guest );
       }

       return guestList;
    }

    // insert a guest in guestbook database
    public void addGuest( GuestBean guest ) throws SQLException
    {
       addRecord.setString( 1, guest.getFirstName() );
       addRecord.setString( 2, guest.getLastName() );
       addRecord.setString( 3, guest.getEmail() );

       addRecord.executeUpdate();
    }

```

Outline

Fig. 10.21 **`GuestDataBean`** performs database access on behalf of **`guestBook-Login.jsp`** (Part 2).

`GuestDataBean` provides methods **`getGuestList`** and **`addGuest`** to manipulate database

```
      // close statements and terminate database connection
      protected void finalize()
      {
         // attempt to close database connection
         try {
            getRecords.close();
            addRecord.close();
            connection.close();
         }

         // process SQLException on close operation
         catch ( SQLException sqlException ) {
            sqlException.printStackTrace();
         }
      }
   }
```

Outline

Fig. 10.21 **`GuestDataBean`** performs database access on behalf of **`guestBook-Login.jsp`** (Part 3).

```
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. 10.22: guestBookLogin.jsp -->
6
7    <%-- page settings --%>
8    <%@ page errorPage = "guestBookErrorPage.jsp" %>
9
10   <%-- beans used in this JSP --%>
11   <jsp:useBean id = "guest" scope = "page"
12      class = "com.deitel.advjhtp1.jsp.beans.GuestBean" />
13   <jsp:useBean id = "guestData" scope = "request"
14      class = "com.deitel.advjhtp1.jsp.beans.GuestDataBean" />
15
16   <html xmlns = "http://www.w3.org/1999/xhtml">
17
18   <head>
19      <title>Guest Book Login</title>
20
21      <style type = "text/css">
22         body {
23            font-family: tahoma, helvetica, arial, sans-serif;
24         }
25
26         table, tr, td {
27            font-size: .9em;
28            border: 3px groove;
29            padding: 5px;
30            background-color: #dddddd;
31         }
32      </style>
33   </head>
34
```

page directive defines information that is globally available in JSP

Use **jsp:useBean** actions to obtain references to **GuestBean** and **GuestDataBean** objects

Outline

ge

Login.jsp enables

d

in the guest book (Part 1).

Line 8

Lines 11-14

```
35  <body>
36     <jsp:setProperty name = "guest" property = "*" />
37
38     <% // start scriptlet
39
40        if ( guest.getFirstName() == null ||
41             guest.getLastName() == null ||
42             guest.getEmail() == null ) {
43
44     %> <%-- end scriptlet to insert fixed template data --%>
45
46           <form method = "post" action = "guestBookLogin.jsp">
47              <p>Enter your first name, last name and email
48                 address to register in our guest book.</p>
49
50              <table>
51                 <tr>
52                    <td>First name</td>
53
54                    <td>
55                       <input type = "text" name = "firstName" />
56                    </td>
57                 </tr>
58
59                 <tr>
60                    <td>Last name</td>
61
62                    <td>
63                       <input type = "text" name = "lastName" />
64                    </td>
65                 </tr>
66
67                 <tr>
68                    <td>Email</td>
69
```

Set properties of **GuestBean** with request parameter values

Outline

Login.jsp enables the user to submit a first name, a last name and an e-mail address to be placed in the guest book (Part 2).

Line 36

```
70                    <td>
71                       <input type = "text" name = "email" />
72                    </td>
73                 </tr>
74
75                 <tr>
76                    <td colspan = "2">
77                       <input type = "submit"
78                          value = "Submit" />
79                    </td>
80                 </tr>
81              </table>
82           </form>
83
84     <% // continue scriptlet
85
86        }  // end if
87        else {
88           guestData.addGuest( guest );
89
90     %> <%-- end scriptlet to insert jsp:forward action --%>
91
92           <%-- forward to display guest book contents --%>
93           <jsp:forward page = "guestBookView.jsp" />
94
95     <% // continue scriptlet
96
97        }  // end else
98
99     %> <%-- end scriptlet --%>
100 </body>
101
102 </html>
```

Forward request to **`guestBookView.jsp`**

Outline

Fig. 10.22 JavaServer page **`guestBook-Login.jsp`** enables the user to submit a first name, a last name and an e-mail address to be placed in the guest book (Part 3).

Line 93

```
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. 10.23: guestBookView.jsp -->
6
7     <%-- page settings --%>
8     <%@ page errorPage = "guestBookErrorPage.jsp" %>
9     <%@ page import = "java.util.*" %>
10    <%@ page import = "com.deitel.advjhtp1.jsp.beans.*" %>
11
12    <%-- GuestDataBean to obtain guest list --%>
13    <jsp:useBean id = "guestData" scope = "request"
14       class = "com.deitel.advjhtp1.jsp.beans.GuestDataBean" />
15
16    <html xmlns = "http://www.w3.org/1999/xhtml">
17
18       <head>
19          <title>Guest List</title>
20
21          <style type = "text/css">
22             body {
23                font-family: tahoma, helvetica, arial, sans-serif;
24             }
25
26             table, tr, td, th {
27                text-align: center;
28                font-size: .9em;
29                border: 3px groove;
30                padding: 5px;
31                background-color: #dddddd;
32             }
33          </style>
34       </head>
35
```

Outline

Use **page** directive **import** to specify Java classes and packages that are used in JSP context

ge

plays
the contents of the
guest book (Part 1).

Use **jsp:useBean** action to obtain reference to **GuestDataBean**

Lines 13-14

```
      <body>
         <p style = "font-size: 2em;">Guest List</p>

         <table>
            <thead>
               <tr>
                  <th style = "width: 100px;">Last name</th>
                  <th style = "width: 100px;">First name</th>
                  <th style = "width: 200px;">Email</th>
               </tr>
            </thead>

            <tbody>

            <% // start scriptlet

               List guestList = guestData.getGuestList();
               Iterator guestListIterator = guestList.iterator();
               GuestBean guest;

               while ( guestListIterator.hasNext() ) {
                  guest = ( GuestBean ) guestListIterator.next();

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

                  <tr>
                     <td><%= guest.getLastName() %></td>

                     <td><%= guest.getFirstName() %></td>

                     <td>
                        <a href = "mailto:<%= guest.getEmail() %>">
                           <%= guest.getEmail() %></a>
                     </td>
                  </tr>
```

Outline

Fig. 10.23 JavaServer page **`guestBook-View.jsp`** displays the contents of the guest book (Part 2).

Scriptlet displays last name, first name and email address for all guests

```

            <% // continue scriptlet

               } // end while

            %> <%-- end scriptlet --%>

            </tbody>
         </table>
      </body>

   </html>
```

Outline

Fig. 10.23 JavaServer page **`guestBook-View.jsp`** displays the contents of the guest book (Part 3).

```
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. 10.24: guestBookErrorPage.jsp -->
6
7     <%-- page settings --%>
8     <%@ page isErrorPage = "true" %>
9     <%@ page import = "java.util.*" %>
10    <%@ page import = "java.sql.*" %>
11
12    <html xmlns = "http://www.w3.org/1999/xhtml">
13
14       <head>
15          <title>Error!</title>
16
17          <style type = "text/css">
18             .bigRed {
19                font-size: 2em;
20                color: red;
21                font-weight: bold;
22             }
23          </style>
24       </head>
25
26       <body>
27          <p class = "bigRed">
28
29          <% // scriptlet to determine exception type
30             // and output beginning of error message
31             if ( exception instanceof SQLException )
32          %>
33
34                An SQLException
35
```

Use **page** directive **isErrorPage** to specify that **guestBookErrorPage** is an error page

Use implicit object **exception** to determine error to be displayed

Outline

or-
nds
to exceptions in **guestBook-Login.jsp** and **guestBook-View.jsp** (Part 1).

Line 8

Line 31

```
36          <%
37             else if ( exception instanceof ClassNotFoundException )
38          %>
39
40                A ClassNotFoundException
41
42          <%
43             else
44          %>
45
46                An exception
47
48          <%-- end scriptlet to insert fixed template data --%>
49
50             <%-- continue error message output --%>
51             occurred while interacting with the guestbook database.
52          </p>
53
54          <p class = "bigRed">
55             The error message was:<br />
56             <%= exception.getMessage() %>
57          </p>
58
59          <p class = "bigRed">Please try again later</p>
60       </body>
61
62    </html>
```

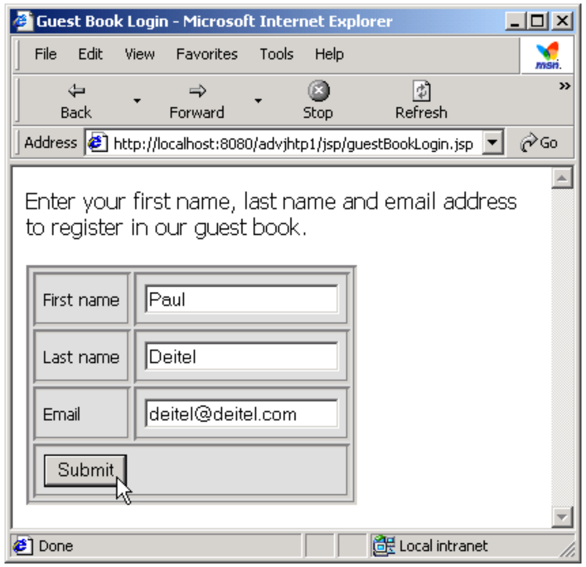
Use implicit object **`exception`** to determine error to be displayed

Outline

page **kError-** **View.jsp** responds to exceptions in **guestBook-Login.jsp** and **guestBook-View.jsp** (Part 2).

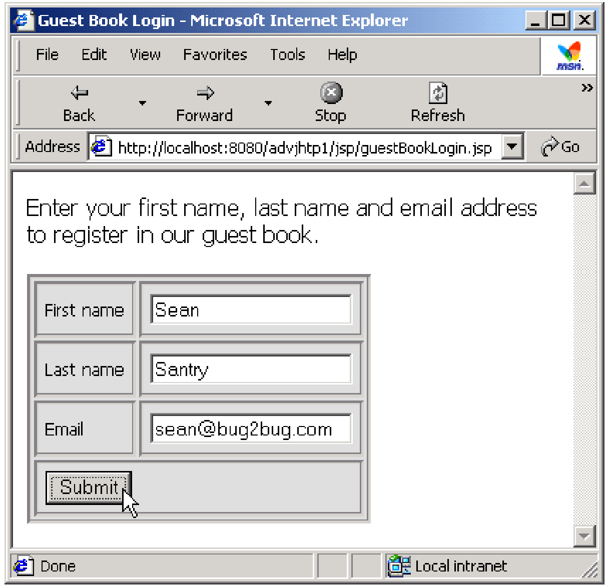
Line 37

10.6.4 `<jsp:useBean>` Action (cont.)


Guest Book Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address http://localhost:8080/advjhtp1/jsp/guestBookLogin.jsp Go
Done
Local intranet

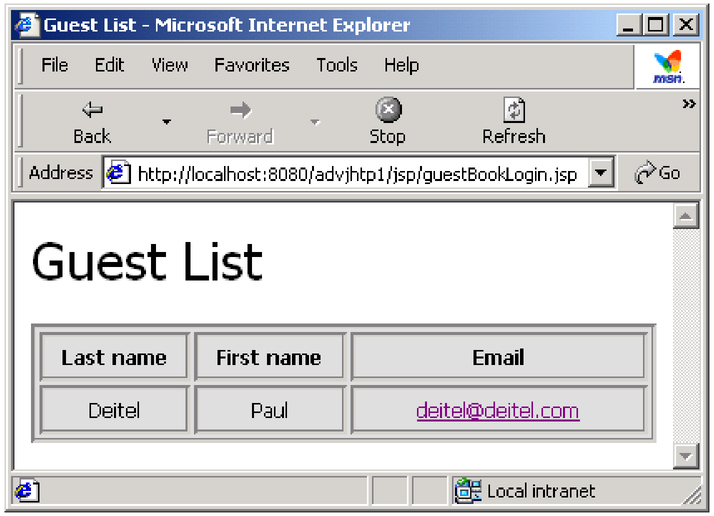
Enter your first name, last name and email address to register in our guest book.

First name	Paul
Last name	Deitel
Email	deitel@deitel.com
Submit	


Guest Book Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address http://localhost:8080/advjhtp1/jsp/guestBookLogin.jsp Go
Done
Local intranet

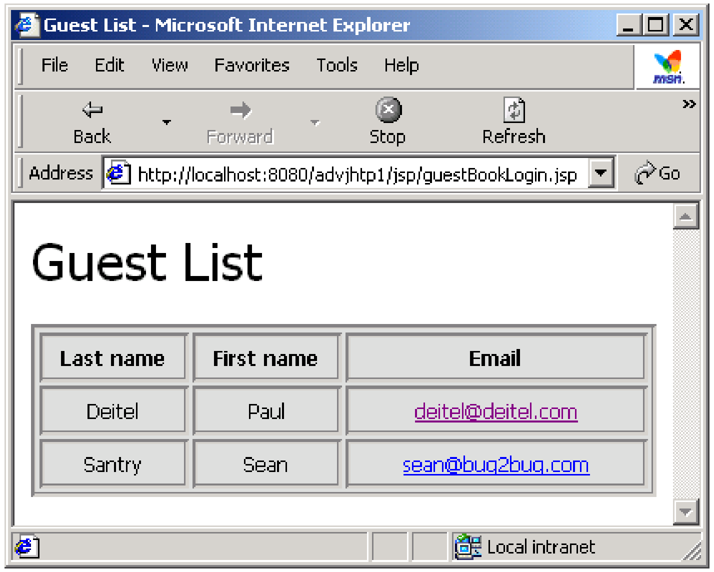
Enter your first name, last name and email address to register in our guest book.

First name	Sean
Last name	Santry
Email	sean@bug2bug.com
Submit	


Guest List - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address http://localhost:8080/advjhtp1/jsp/guestBookLogin.jsp Go
Local intranet

Guest List

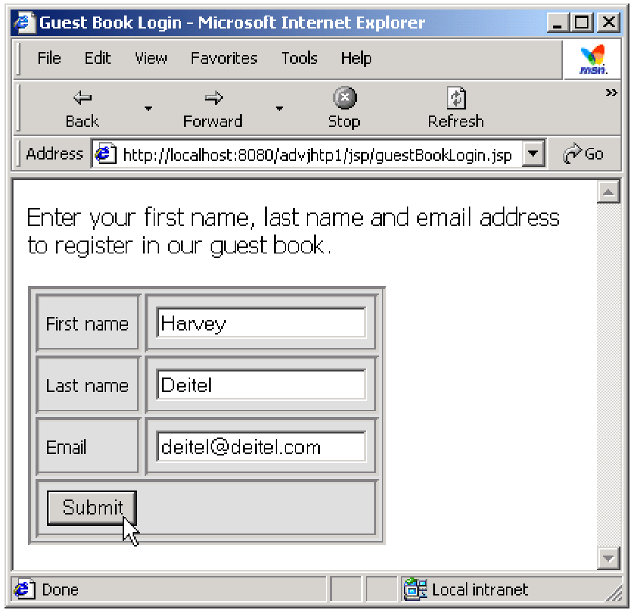
Last name	First name	Email
Deitel	Paul	deitel@deitel.com


Guest List - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address http://localhost:8080/advjhtp1/jsp/guestBookLogin.jsp Go
Local intranet

Guest List

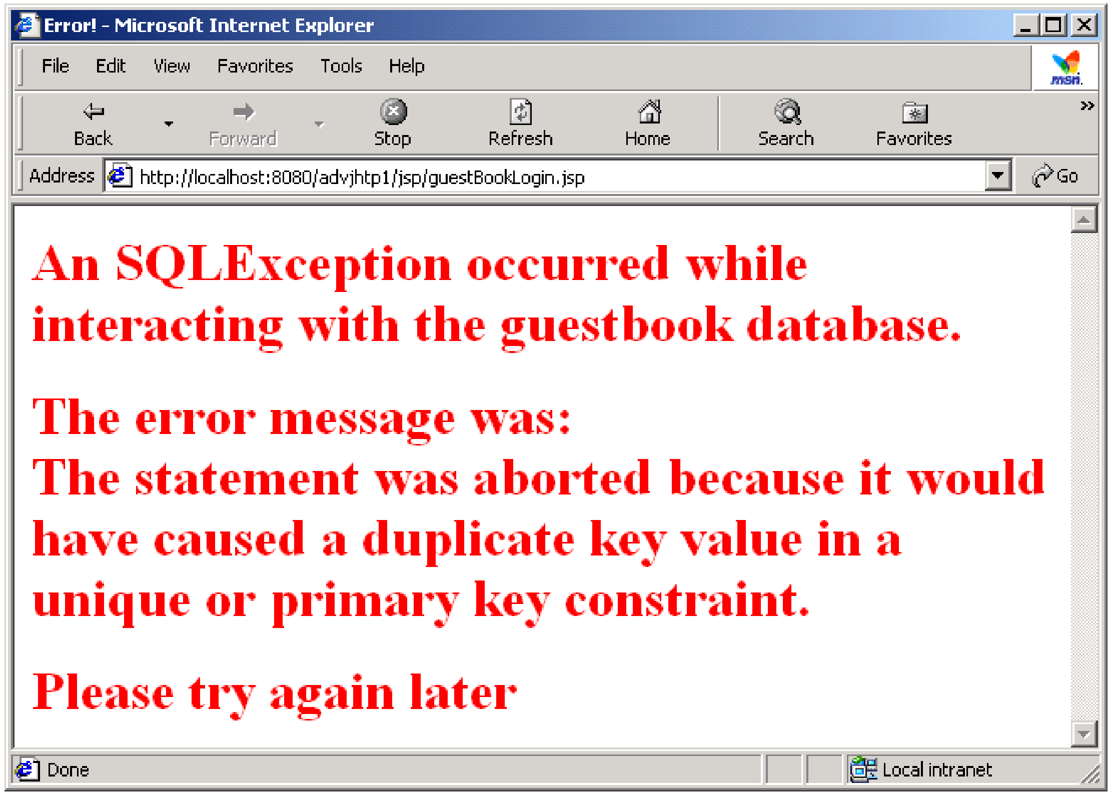
Last name	First name	Email
Deitel	Paul	deitel@deitel.com
Santry	Sean	sean@bug2bug.com

10.6.4 `<jsp:useBean>` Action (cont.)


Guest Book Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address http://localhost:8080/advjhtp1/jsp/guestBookLogin.jsp Go
Submit
Done
Local intranet

Enter your first name, last name and email address to register in our guest book.

First name	Harvey
Last name	Deitel
Email	deitel@deitel.com


Error! - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/advjhtp1/jsp/guestBookLogin.jsp Go
Done
Local intranet

An SQLException occurred while interacting with the guestbook database.

The error message was:
The statement was aborted because it would have caused a duplicate key value in a unique or primary key constraint.

Please try again later

10.7 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 **`%>`**

10.7 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 include their own new tags in the form of *tag libraries*. These libraries can be used to encapsulate functionality and simplify the coding of a JSP.

Fig. 10.26 JSP directives.

10.7.1 page Directive

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

10.7.1 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 package and class name.
`import`	Specifies a comma-separated list of fully qualified class 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, the JSP implicit object **`session`** is available for use in the page. Otherwise, **`session`** is not available. In the latter case, using **`session`** in the scripting code results in a translation-time error.
`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.

Fig. 10.27 Attributes of the **`page`** directive (part 1 of 3).

10.7.1 page Directive (cont.)

Attribute	Description
`autoFlush`	When set to **`true`** (the default value), 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. 10.27 Attributes of the **`page`** directive (part 2 of 3).

10.7.1 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. 10.27 Attributes of the **`page`** directive (part 3 of 3).

10.7.2 `include` Directive

- JSP **`include`** directive
  - Includes content of another resource at JSP translation time
    - Not as flexible as **`<jsp:include>`** action

```
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. 10.28: includeDirective.jsp -->
6
7     <html xmlns = "http://www.w3.org/1999/xhtml">
8
9        <head>
10          <title>Using the include directive</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
```

Reimplement **include.jsp** using **include** directives

Outline

ve **.jsp** demonstrates including content at translation-time with directive include (Part 1).

Line 5

```
35                <td>
36
37                   <%-- include banner.html in this JSP --%>
38                   <%@ include file = "banner.html" %>
39
40                </td>
41             </tr>
42
43             <tr>
44                <td style = "width: 160px">
45
46                   <%-- include toc.html in this JSP --%>
47                   <%@ include file = "toc.html" %>
48
49                </td>
50
51                <td style = "vertical-align: top">
52
53                   <%-- include clock2.jsp in this JSP --%>
54                   <%@ include file = "clock2.jsp" %>
55
56                </td>
57             </tr>
58          </table>
59       </body>
60    </html>
```

Use **include** directive to include **banner.html**

Use **include** directive to include **toc.html**

Use **include** directive to include **clock2.jsp**

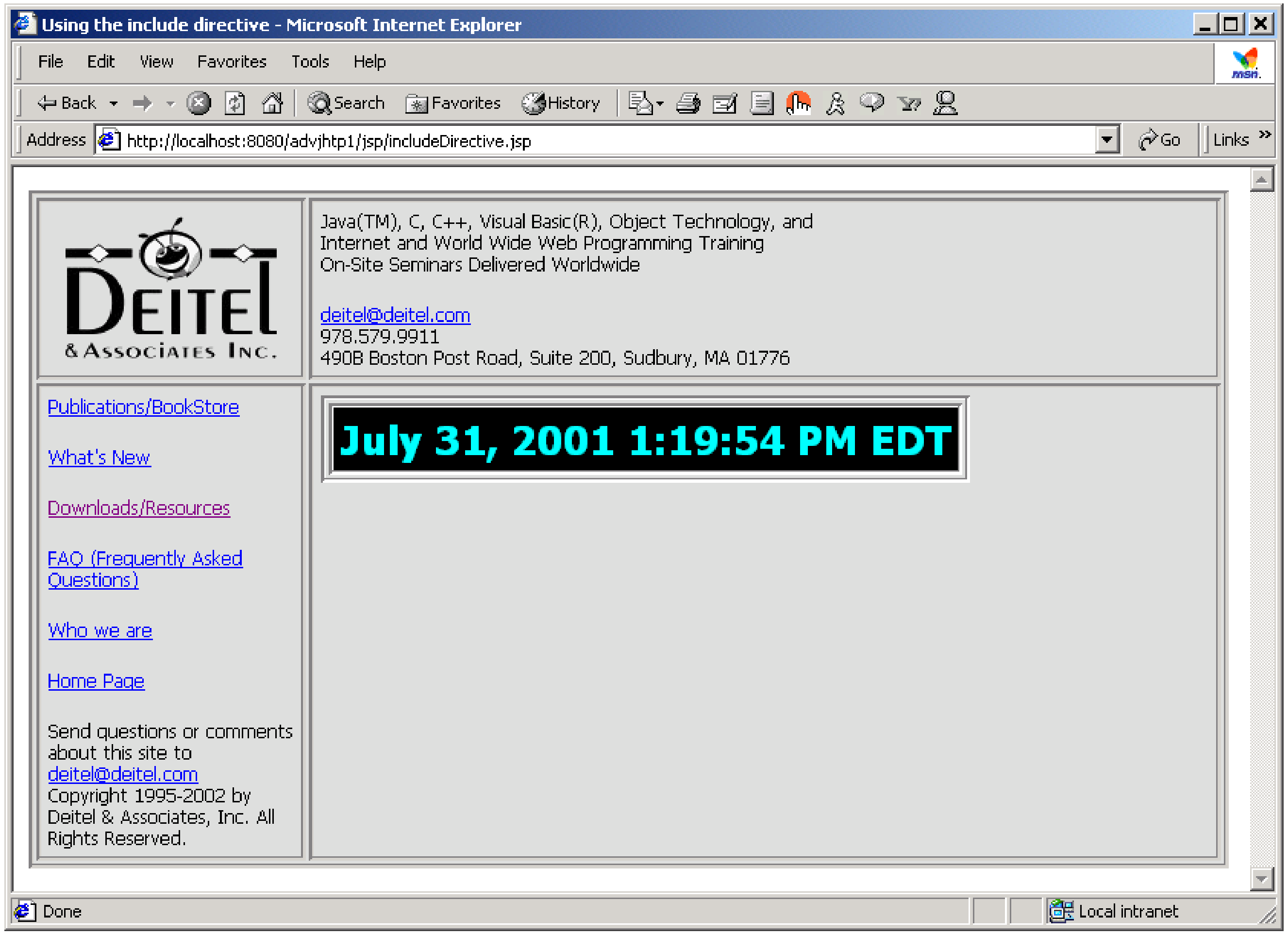
Outline

includeDirective.jsp demonstrates including content at translation-time with directive include

Line 38

Line 47

Outline


Using the include directive - Microsoft Internet Explorer

Address http://localhost:8080/advjhtp1/jsp/includeDirective.jsp

Java(TM), C, C++, Visual Basic(R), Object Technology, and
Internet and World Wide Web Programming Training
On-Site Seminars Delivered Worldwide

deitel@deitel.com
978.579.9911
490B Boston Post Road, Suite 200, Sudbury, MA 01776

Publications/BookStore

What's New

Downloads/Resources

FAQ (Frequently Asked Questions)

Who we are

Home Page

Send questions or comments about this site to deitel@deitel.com
Copyright 1995-2002 by Deitel & Associates, Inc. All Rights Reserved.

July 31, 2001 1:19:54 PM EDT

Fig. 10.28 JSP **`includeDirective.jsp`** demonstrates including content at translation-time with directive include (Part 3).

10.8 Custom Tag Libraries

- Custom-tag libraries
  - Encapsulates complex functionality for use in JSPs
  - Define custom tags
    - Used for creating dynamic content
    - Classes that implement interface **`Tag`**
      - Pacakge **`javax.servlet.jsp.tagext`**

10.8 Custom Tag Libraries (cont.)

Attribute	Description
uri	Specifies the relative or absolute URI of the tag library descriptor.
tagPrefix	Specifies the required prefix that distinguishes custom tags from built-in tags. The prefix names **jsp**, **jspx**, **java**, **javax**, **servlet**, **sun** and **sunw** are reserved.

Fig. 10.29 Attributes of the **taglib** directive.

```
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. 10.30: customTagWelcome.jsp               -->
6     <!-- JSP that uses a custom tag to output content. -->
7
8     <%-- taglib directive --%>
9     <%@ taglib uri = "advjhtp1-taglib.tld" prefix = "advjhtp1" %>
10
11    <html xmlns = "http://www.w3.org/1999/xhtml">
12
13       <head>
14          <title>Simple Custom Tag Example</title>
15       </head>
16
17       <body>
18          <p>The following text demonstrates a custom tag:</p>
19          <h1>
20             <advjhtp1:welcome />
21          </h1>
22       </body>
23
24    </html>
```

Use **taglib** directive to include use tags in tag library

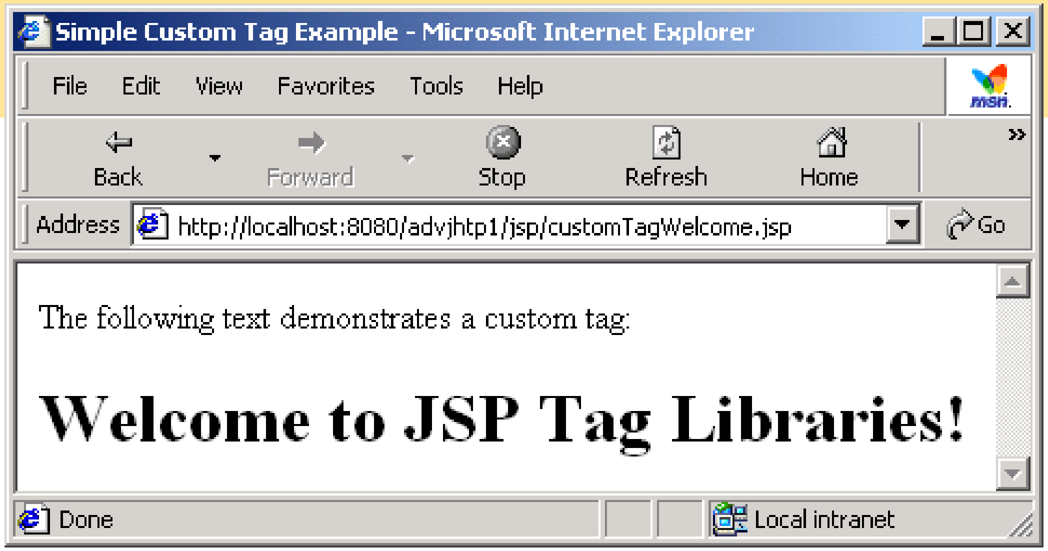
Use custom tag **welcome** to insert text in the JSP

Outline

customTagWelcome.jsp uses a simple custom tag.

Line 9

Line 20


Simple Custom Tag Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home
Address http://localhost:8080/advjhtp1/jsp/customTagWelcome.jsp Go
Done
Local intranet

The following text demonstrates a custom tag:

Welcome to JSP Tag Libraries!

```java
1     // Fig. 10.31: WelcomeTagHandler.java
2     // Custom tag handler that handles a simple tag.
3     package com.deitel.advjhtp1.jsp.taglibrary;
4
5     // Java core packages
6     import java.io.*;
7
8     // Java extension packages
9     import javax.servlet.jsp.*;
10    import javax.servlet.jsp.tagext.*;
11
12    public class WelcomeTagHandler extends TagSupport {
13
14       // Method called to begin tag processing
15       public int doStartTag() throws JspException
16       {
17          // attempt tag processing
18          try {
19             // obtain JspWriter to output content
20             JspWriter out = pageContext.getOut();
21
22             // output content
23             out.print( "Welcome to JSP Tag Libraries!" );
24          }
25
26          // rethrow IOException to JSP container as JspException
27          catch( IOException ioException ) {
28             throw new JspException( ioException.getMessage() );
29          }
30
31          return SKIP_BODY;  // ignore the tag's body
32       }
33    }
```

Class **WelcomeTagHandler** implements interface **Tag** by extending class **TagSupport**

JSP container calls method **doStartTag** when it encounters the starting custom tag

Use custom tag handler's **pageContext** to obtain JSP's **JspWriter** object for outputting text

Outline

Fig. 10.31

om tag

Line 12

```
1    <?xml version = "1.0" encoding = "ISO-8859-1" ?>
2    <!DOCTYPE taglib PUBLIC
3       "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
4       "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
5
6    <!-- a tag library descriptor -->
7
8    <taglib>
9       <tlibversion>1.0</tlibversion>
10      <jspversion>1.1</jspversion>
11      <shortname>advjhtp1</shortname>
12
13      <info>
14         A simple tab library for the examples
15      </info>
16
17      <!-- A simple tag that outputs content -->
18      <tag>
19         <name>welcome</name>
20
21         <tagclass>
22            com.deitel.advjhtp1.jsp.taglibrary.WelcomeTagHandler
23         </tagclass>
24
25         <bodycontent>empty</bodycontent>
26
27         <info>
28            Inserts content welcoming user to tag libraries
29         </info>
30      </tag>
31   </taglib>
```

Outline

Fig. 10.32 Custom tag library descriptor

Define custom-tag library descriptor file

Line 8

`tag` element describes **`welcome`** custom tag

Lines 19-23

Specify custom tag name and class

10.8.2 Custom Tag with Attributes

- XHTML and JSP elements
  - Use attributes to customize functionality
    - Specify attributes for custom tags

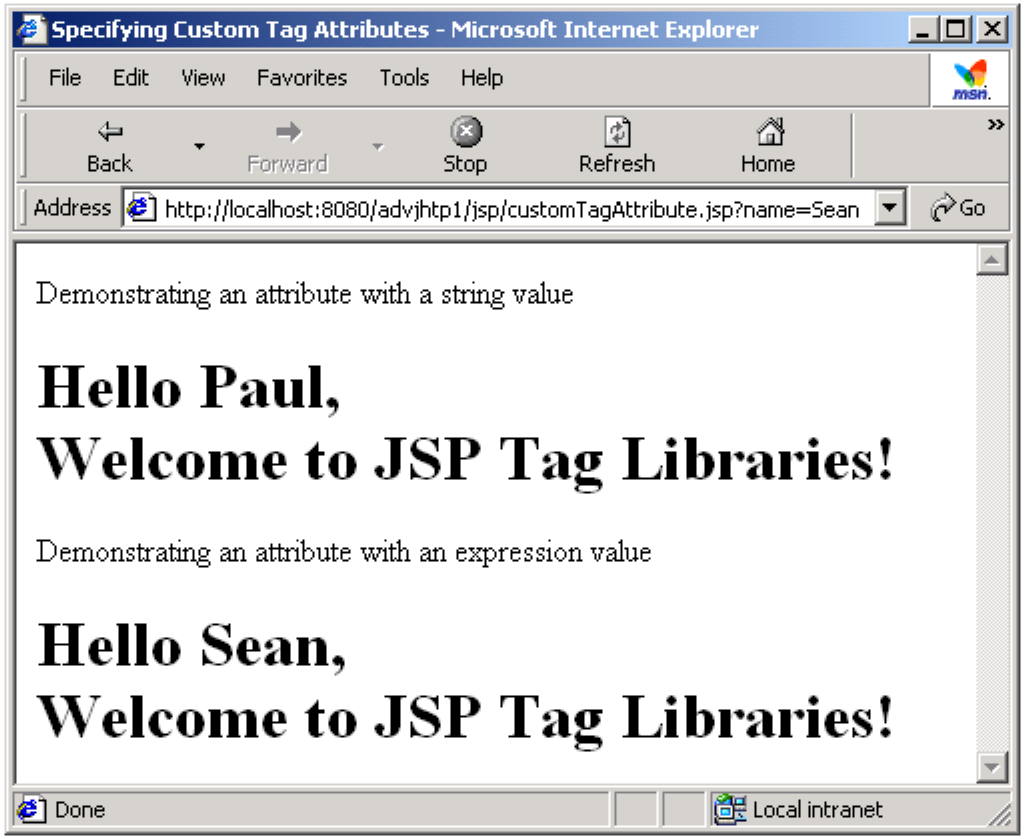
```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 10.33: customTagAttribute.jsp             -->
6   <!-- JSP that uses a custom tag to output content. -->
7
8   <%-- taglib directive --%>
9   <%@ taglib uri = "advjhtp1-taglib.tld" prefix = "advjhtp1" %>
10
11  <html xmlns = "http://www.w3.org/1999/xhtml">
12
13     <head>
14        <title>Specifying Custom Tag Attributes</title>
15     </head>
16
17     <body>
18        <p>Demonstrating an attribute with a string value</p>
19        <h1>
20           <advjhtp1:welcome2 firstName = "Paul" />
21        </h1>
22
23        <p>Demonstrating an attribute with an expression value</p>
24        <h1>
25           <%-- scriptlet to obtain "name" request parameter --%>
26           <%
27              String name = request.getParameter( "name" );
28           %>
29
30           <advjhtp1:welcome2 firstName = "<%= name %>" />
31        </h1>
32     </body>
33
34  </html>
```

Fig. 10.33 Specifying attributes for a custom tag (Part 1).

Lines 20 and 30

Use **welcome2** tag to insert text in JSP that is customized based on **firstName** attribute value

Outline


Specifying Custom Tag Attributes - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home
Address http://localhost:8080/advjhtp1/jsp/customTagAttribute.jsp?name=Sean
Go
Done
Local intranet

Demonstrating an attribute with a string value

Hello Paul,
Welcome to JSP Tag Libraries!

Demonstrating an attribute with an expression value

Hello Sean,
Welcome to JSP Tag Libraries!

Fig. 10.33 Specifying attributes for a custom tag (Part 2).

```
1    // Fig. 10.34: Welcome2TagHandler.java
2    // Custom tag handler that handles a simple tag.
3    package com.deitel.advjhtp1.jsp.taglibrary;
4
5    // Java core packages
6    import java.io.*;
7
8    // Java extension packages
9    import javax.servlet.jsp.*;
10   import javax.servlet.jsp.tagext.*;
11
12   public class Welcome2TagHandler extends TagSupport {
13      private String firstName = "";
14
15      // Method called to begin tag processing
16      public int doStartTag() throws JspException
17      {
18         // attempt tag processing
19         try {
20            // obtain JspWriter to output content
21            JspWriter out = pageContext.getOut();
22
23            // output content
24            out.print( "Hello " + firstName +
25               ", <br />Welcome to JSP Tag Libraries!" );
26         }
27
28         // rethrow IOException to JSP container as JspException
29         catch( IOException ioException ) {
30            throw new JspException( ioException.getMessage() );
31         }
32
33         return SKIP_BODY;  // ignore the tag's body
34      }
35
```

Define **`firstName`** attribute

Use **`firstName`** attribute value as part of text output by custom tag

Outline

Fig. 10.34 `Welcome2TagHandler` custom tag handler for a tag with an attribute (Part 1).

Lines 24-25

```
36     // set firstName attribute to the users first name
37     public void setFirstName( String username )
38     {
39        firstName = username;
40     }
41  }
```

Corresponding set accessor method for **firstName** attribute value

Outline

Fig. 10.34
2Tag-
custom tag handler for a tag with an attribute (Part 2).

Lines 37-40

```
1   <!-- A tag with an attribute -->
2   <tag>
3      <name>welcome2</name>
4
5      <tagclass>
6         com.deitel.advjhtp1.jsp.taglibrary.Welcome2TagHandler
7      </tagclass>
8
9      <bodycontent>empty</bodycontent>
10
11     <info>
12        Inserts content welcoming user to tag libraries. Uses
13        attribute "name" to insert the user's name.
14     </info>
15
16     <attribute>
17        <name>firstName</name>
18        <required>true</required>
19        <rtexprvalue>true</rtexprvalue>
20     </attribute>
21  </tag>
```

Introduce element **`attribute`** for specifying the characteristics of a tag's attributes

Fig. 10.35 Element **`tag`** for the **`welcome2`** custom tag.

Lines 16-20

10.8.3 Evaluating the Body of a Custom Tag

- Custom tags
  - Particularly useful for processing element body

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

<!-- customTagBody.jsp -->

<%-- taglib directive --%>
<%@ taglib uri = "advjhtp1-taglib.tld" prefix = "advjhtp1" %>

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

   <head>
      <title>Guest List</title>

      <style type = "text/css">
         body {
            font-family: tahoma, helvetica, arial, sans-serif
         }

         table, tr, td, th {
            text-align: center;
            font-size: .9em;
            border: 3px groove;
            padding: 5px;
            background-color: #dddddd
         }
      </style>
   </head>

   <body>
      <p style = "font-size: 2em">Guest List</p>

```

Fig. 10.36 Using a custom tag that interacts with its body (Part 1).

```
33          <table>
34             <thead>
35                <th style = "width: 100px">Last name</th>
36                <th style = "width: 100px">First name</th>
37                <th style = "width: 200px">Email</th>
38             </thead>
39
40             <%-- guestlist custom tag --%>
41             <advjhtp1:guestlist>
42                <tr>
43                   <td><%= lastName %></td>
44
45                   <td><%= firstName %></td>
46
47                   <td>
48                      <a href = "mailto:<%= email %>">
49                         <%= email %></a>
50                   </td>
51                </tr>
52             </advjhtp1:guestlist>
53          </table>
54       </body>
55
56    </html>
```

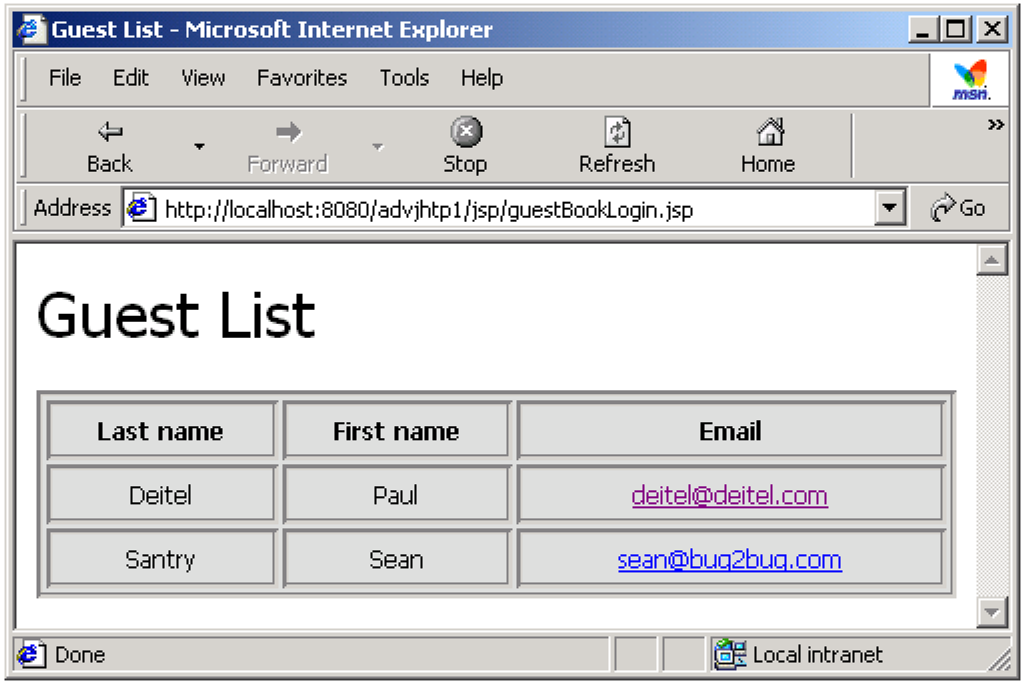
Outline

Fig. 10.36 Using a custom tag that interacts with its body (Part 2).

Lines 41-52

Use custom **`guestlist`** tag to display last name, first name and email

Outline


Guest List - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home
Address http://localhost:8080/advjhtp1/jsp/guestBookLogin.jsp
Go
Done
Local intranet

Guest List

Last name	First name	Email
Deitel	Paul	deitel@deitel.com
Santry	Sean	sean@bug2bug.com

Fig. 10.36 Using a custom tag that interacts with its body (Part 3).

```java
// Fig. 10.37: GuestBookTag.java
// Custom tag handler that reads information from the guestbook
// database and makes that data available in a JSP.
package com.deitel.advjhtp1.jsp.taglibrary;

// Java core packages
import java.io.*;
import java.util.*;

// Java extension packages
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

// Deitel packages
import com.deitel.advjhtp1.jsp.beans.*;

public class GuestBookTag extends BodyTagSupport {
   private String firstName;
   private String lastName;
   private String email;

   private GuestDataBean guestData;
   private GuestBean guest;
   private Iterator iterator;

   // Method called to begin tag processing
   public int doStartTag() throws JspException
   {
      // attempt tag processing
      try {
         guestData = new GuestDataBean();

         List list = guestData.getGuestList();
         iterator = list.iterator();

```

Outline

Fig. 10.37 **`GuestBookTag`** custom tag handler (Part 1).

```
36               if ( iterator.hasNext() ) {
37                  processNextGuest();
38
39                  return EVAL_BODY_TAG;    // continue body proces
40               }
41               else
42                  return SKIP_BODY;        // terminate body processing
43            }
44
45            // if any exceptions occur, do not continue processing
46            // tag's body
47            catch( Exception exception ) {
48               exception.printStackTrace();
49               return SKIP_BODY;   // ignore the tag's body
50            }
51         }
52
53         // process body and determine if body processing
54         // should continue
55         public int doAfterBody()
56         {
57            // attempt to output body data
58            try {
59               bodyContent.writeOut( getPreviousOut() );
60            }
61
62            // if exception occurs, terminate body processing
63            catch ( IOException ioException ) {
64               ioException.printStackTrace();
65               return SKIP_BODY;         // terminate body processing
66            }
67
68            bodyContent.clearBody();
69
```

Extract information for first guest and create variable containing that information in the JSP's **PageContext**

custom tag handler (Part 2).

Lines 36-27

Line 55

Line 59

Method **doAfterBody** can be called many times to process the body of a custom tag

Process the first client's data

Ensure that the outputted body content is not processed in subsequent call to method **doAfterBody**

```
70           if ( iterator.hasNext() ) {
71              processNextGuest();
72 
73              return EVAL_BODY_TAG;    // continue body processing
74           }
75           else
76              return SKIP_BODY;        // terminate body processing
77        }
78 
79        // obtains the next GuestBean and extracts its data
80        private void processNextGuest()
81        {
82           // get next guest
83           guest = ( GuestBean ) iterator.next();
84 
85           pageContext.setAttribute(
86              "firstName", guest.getFirstName() );
87 
88           pageContext.setAttribute(
89              "lastName", guest.getLastName() );
90 
91           pageContext.setAttribute(
92              "email", guest.getEmail() );
93        }
94     }
```

Extract information for next guest (if available)

Fig. 10.37
`GuestBookTag`
custom tag handler (Part 3).

Lines 70-71

```java
// Fig. 10.38: GuestBookTagExtraInfo.java
// Class that defines the variable names and types created by
// custom tag handler GuestBookTag.
package com.deitel.advjhtp1.jsp.taglibrary;

// Java core packages
import javax.servlet.jsp.tagext.*;

public class GuestBookTagExtraInfo extends TagExtraInfo {

   // method that returns information about the variables
   // GuestBookTag creates for use in a JSP
   public VariableInfo [] getVariableInfo( TagData tagData )
   {
      VariableInfo firstName = new VariableInfo( "firstName",
         "String", true, VariableInfo.NESTED );

      VariableInfo lastName = new VariableInfo( "lastName",
         "String", true, VariableInfo.NESTED );

      VariableInfo email = new VariableInfo( "email",
         "String", true, VariableInfo.NESTED );

      VariableInfo varableInfo [] =
         { firstName, lastName, email };

      return varableInfo;
   }
}
```

Outline

Fig. 10.38 `GuestBookTag-ExtraInfo` used by the container to define scripting variables in a JSP that uses the `guestlist` custom tag.

```
<!-- A tag that iterates over an ArrayList of GuestBean -->
<!-- objects, so they can be output in a JSP            -->
<tag>
   <name>guestlist</name>

   <tagclass>
      com.deitel.advjhtp1.jsp.taglibrary.GuestBookTag
   </tagclass>

   <teiclass>
      com.deitel.advjhtp1.jsp.taglibrary.GuestBookTagExtraInfo
   </teiclass>

   <bodycontent>JSP</bodycontent>

   <info>
      Iterates over a list of GuestBean objects
   </info>
</tag>
```

Specify custom tag's **ExtraInfo** class

Outline

Fig. 10.39 Element **tag** for the **guest-**