Session 3

How to develop JavaServer Pages

Objectives

Applied

- Code and test JavaServer Pages that use scriptlets, expressions, implicit request objects, ServletContext objects, page directives, JSP comments, and JSP declarations.

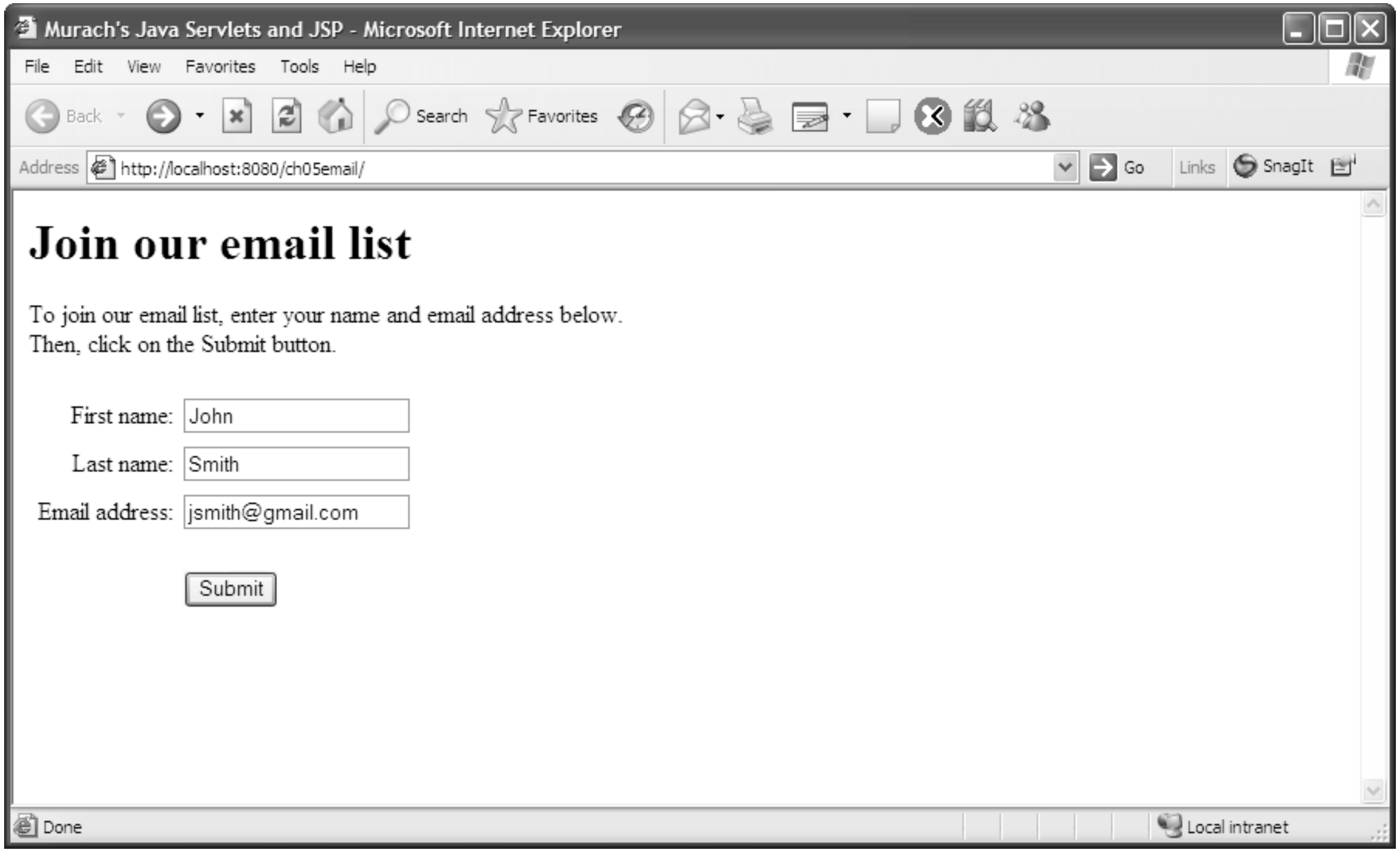
Knowledge

- Describe the use of the implicit request object.
- Describe how the ServletContext object can be used to get the path for a file.
- Describe the way parameters are passed to a JSP when you use the Get method.
- List three reasons for using the Post method instead of the Get method.

Objectives (continued)

- Explain what is meant by a thread-safe JSP.
- Explain how a JSP is executed the first time it is requested and for subsequent requests.

The HTML page for an Email List application


Murach's Java Servlets and JSP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/ch05email/
Go Links SnagIt
Done
Local intranet

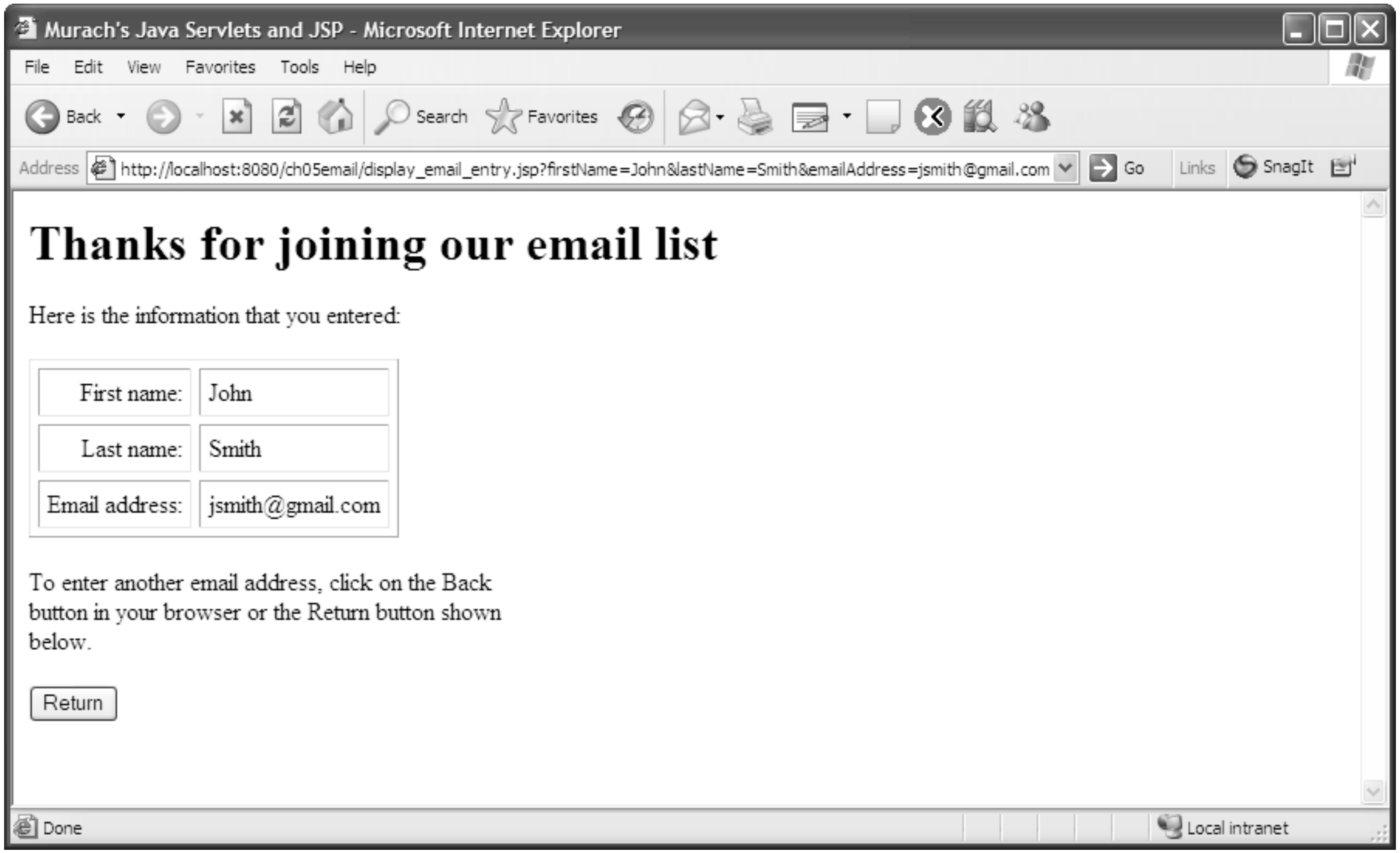
Join our email list

To join our email list, enter your name and email address below.
Then, click on the Submit button.

First name:	John
Last name:	Smith
Email address:	jsmith@gmail.com

Submit

The JSP for an Email List application


Murach's Java Servlets and JSP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links
SnagIt
Done
Local intranet

http://localhost:8080/ch05email/display_email_entry.jsp?firstName=John&lastName=Smith&emailAddress=jsmith@gmail.com

Thanks for joining our email list

Here is the information that you entered:

First name:	John
Last name:	Smith
Email address:	jsmith@gmail.com

To enter another email address, click on the Back button in your browser or the Return button shown below.

Return

The code for the HTML page

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>
    <title>Murach's Java Servlets and JSP</title>
</head>

<body>
    <h1>Join our email list</h1>
    <p>To join our email list, enter your name and
     email address below. <br>
     Then, click on the Submit button.</p>

    <form action="display_email_entry.jsp" method="get">
    <table cellspacing="5" border="0">
        <tr>
            <td align="right">First name:</td>
            <td><input type="text" name="firstName"></td>
        </tr>
```

The code for the HTML page (continued)

```
        <tr>
            <td align="right">Last name:</td>
            <td><input type="text" name="lastName"></td>
        </tr>
        <tr>
            <td align="right">Email address:</td>
            <td><input type="text" name="emailAddress"></td>
        </tr>
        <tr>
            <td></td>
            <td><br>
                <input type="submit" value="Submit"></td>
        </tr>
    </table>
    </form>
</body>

</html>
```

The code for the HTML page that calls the JSP

- The Action and Method attributes for the Form tag set up a request for a JSP that will be executed when the user clicks on the Submit button.
- The three text boxes represent *parameters* that will be passed to the JSP when the user clicks the Submit button.

The code for the JSP

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
    <title>Murach's Java Servlets and JSP</title>
</head>
<body>
    <%
        // get parameters from the request
        String firstName =
            request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String emailAddress =
            request.getParameter("emailAddress");
    %>

    <h1>Thanks for joining our email list</h1>

    <p>Here is the information that you entered:</p>
```

The code for the JSP (continued)

```
<table cellspacing="5" cellpadding="5" border="1">
    <tr>
        <td align="right">First name:</td>
        <td><%= firstName %></td>
    </tr>
    <tr>
        <td align="right">Last name:</td>
        <td><%= lastName %></td>
    </tr>
    <tr>
        <td align="right">Email address:</td>
        <td><%= emailAddress %></td>
    </tr>
</table>

<p>To enter another email address, click on the Back <br>
button in your browser or the Return button shown <br>
below.</p>
```

The code for the JSP (continued)

```
    <form action="join_email_list.html" method="post">
        <input type="submit" value="Return">
    </form>

</body>
</html>
```

The code for a JSP

- A JSP contains HTML tags and embedded Java code.
- To code a *scriptlet* that contains one or more Java statements, you use the <% and %> tags.
- To code an *expression* that can be converted to a string, you use the <%= and %> tags.
- To get the values of the parameters that are passed to the JSP, you can use the getParameter method of *implicit request object* named request.

The syntax for a JSP scriptlet

```
<% Java statements %>
```

The syntax for a JSP expression

```
<%= any Java expression that can be converted to a string %>
```

The syntax for getting a parameter from the implicit request object

```
request.getParameter(parameterName);
```

A scriptlet and expression that display the value of the firstName parameter

```
<%
    String firstName = request.getParameter("firstName");
%>
The first name is <%= firstName %>.
```

An expression that displays the value of the firstName parameter

```
The first name is <%= request.getParameter("firstName") %>.
```

Two scriptlets and an expression that display an HTML line 5 times

```
<%
    int numOfTimes = 1;
    while (numOfTimes <= 5)
    {
%>
    <h1>This line is shown <%= numOfTimes %>
        of 5 times in a JSP.</h1>
<%
        numOfTimes++;
    }
%>
```

How to code scriplets and expressions

- Within a scriptlet, you can code one or more Java statements. You must end each Java statement with a semicolon.
- Within a JSP expression, you can code any Java expression that evaluates to a Java object or to a primitive type. Since an expression isn’t a statement, you don’t end it with a semicolon.

Three methods available from the request object

Method	Description
`getParameter(String param)`	Returns the value of the specified parameter as a string if it exists or null if it doesn’t.
`getParameterValues(String param)`	Returns an array of String objects containing all of the values that the given request parameter has or null if the parameter doesn’t have any values.
`getParameterNames()`	Returns an Enumeration object that contains the names of all the parameters contained in the request. If the request has no parameters, the method returns an empty Enumeration object.

A scriptlet that determines if a checkbox is checked

```
<%
    // returns the value or "on" if checked, null otherwise.
    String rockCheckBox = request.getParameter("Rock");
    if (rockCheckBox != null)
    {
%>
        You checked Rock music!
<%
    }
%>
```

A scriptlet that reads and displays multiple values from a list box

```
<%
    // returns the values of items selected in a list box.
    String[] selectedCountries =
        request.getParameterValues("country");
    for (int i = 0; i < selectedCountries.length; i++)
    {
%>
        <%= selectedCountries[i] %> <br>
<%
    }
%>
```

A scriptlet that reads and displays all request parameters and values

```
<%
    Enumeration parameterNames =
        request.getParameterNames();
    while (parameterNames.hasMoreElements())
    {
        String parameterName = (String)
            parameterNames.nextElement();
        String parameterValue =
            request.getParameter(parameterName);
%>
        <%= parameterName %> has value
            <%= parameterValue %>. <br>
<%
     }
%>
```

A method of the GenericServlet class

Method	Description
`getServletContext()`	Returns a ServletContext object that contains information about the application’s context.

A method of the ServletContext class for working with paths

Method	Description
`getRealPath(String path)`	Returns a String object for the real path of the specified relative path.

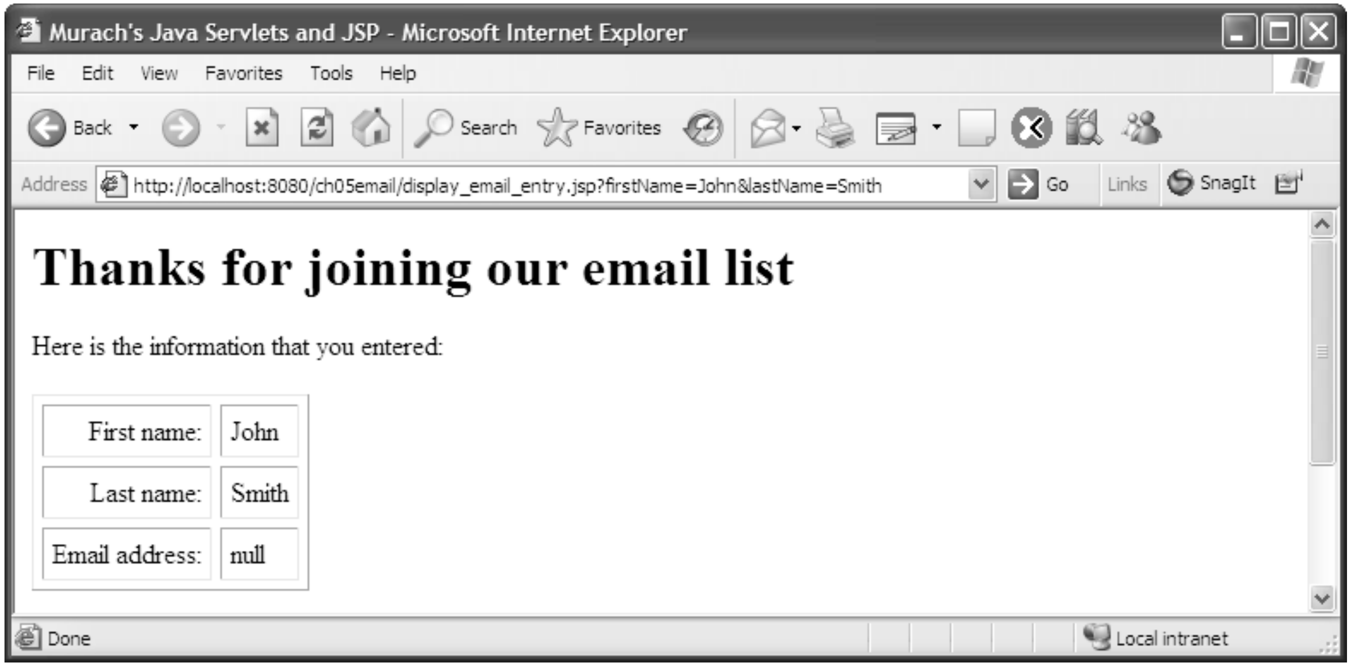
Code that gets the real path for a file

```
ServletContext sc = this.getServletContext();
String path = sc.getRealPath("/WEB-INF/EmailList.txt");
```

The value for the real path variable if the application is ch05email in the netbeans directory

```
C:\murach\servlet_jsp\netbeans\book_apps\ch05email\build\web\
WEB-INF\EmailList.txt
```

A JSP that's requested with the HTTP Get method


Murach's Java Servlets and JSP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go Links SnagIt
Done
Local intranet

http://localhost:8080/ch05email/display_email_entry.jsp?firstName=John&lastName=Smith

Thanks for joining our email list

Here is the information that you entered:

First name:	John
Last name:	Smith
Email address:	null

Two Form tags that use the Get method

```
<form action="display_email_entry.jsp">
<form action="display_email_entry.jsp" method="get">
```

How to append parameters to a request

```
display_email_entry.jsp?firstName=John
display_email_entry.jsp?firstName=John&lastName=Smith
```

An Anchor tag that requests a JSP with the Get method

```
<a href="display_email_entry.jsp?firstName=John&lastName=Smith">
    Display Email Entry Test
</a>
```

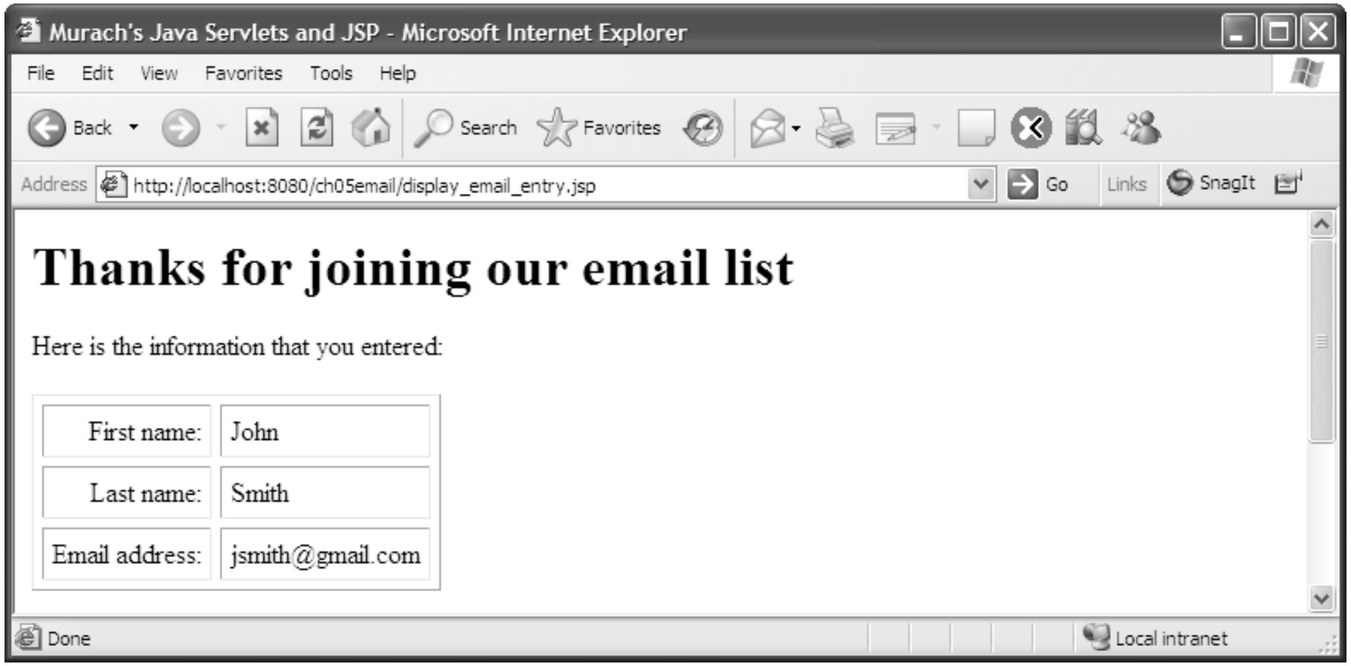
Two URLs that request a JSP with the Get method

```
http://localhost:8080/ch05email/display_email_entry.jsp?firstName=Jo
http://www.murach.com/email/display_email_entry.jsp?firstName=John
```

How to request a JSP with the HTTP Get method

- When you use the HTTP Get method to request a JSP from an HTML form, the parameters are automatically appended to the URL.
- When you code or enter a URL that requests a JSP, you can add a parameter list to it starting with a question mark and with no intervening spaces. Then, each parameter consists of its name, an equals sign, and its value.
- To code multiple parameters, use ampersands (&) to separate the parameters.
- The A tag always uses the HTTP Get method.

A JSP that's requested with the HTTP Post method


Murach's Java Servlets and JSP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost:8080/ch05email/display_email_entry.jsp
Go
Links
SnagIt
Done
Local intranet

Thanks for joining our email list

Here is the information that you entered:

First name:	John
Last name:	Smith
Email address:	jsmith@gmail.com

A Form tag that uses the Post method

```
<form action="display_email_entry.jsp" method="post">
```

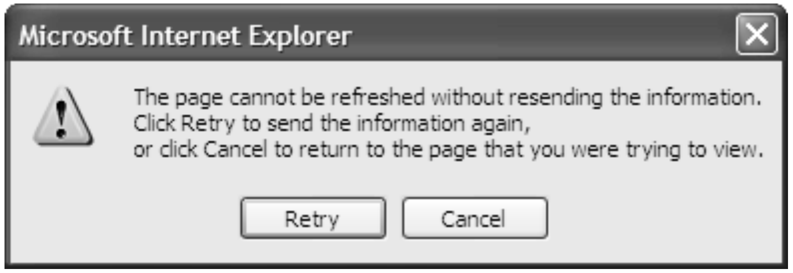
When to use the Get method

- When the request reads data from the server.
- When the request can be executed multiple times without causing any problems.

When to use the Post method

- When the request writes data to the server.
- When executing the request multiple times may cause problems.
- When you don’t want to include the parameters in the URL for security reasons.
- When you don’t want users to be able to include parameters when they bookmark a page.
- When you need to transfer more than 4 KB of data.

A typical browser dialog that's displayed if the user tries to refresh a post


Microsoft Internet Explorer

The page cannot be refreshed without resending the information.
Click Retry to send the information again,
or click Cancel to return to the page that you were trying to view.

The code for the User class

```
package business;

public class User
{
    private String firstName;
    private String lastName;
    private String emailAddress;

    public User()
    {
        firstName = "";
        lastName = "";
        emailAddress = "";
    }
```

The code for the User class (continued)

```
public User(String firstName, String lastName,
String emailAddress)
{
    this.firstName = firstName;
    this.lastName = lastName;
    this.emailAddress = emailAddress;
}

public void setFirstName(String firstName)
{
    this.firstName = firstName;
}

public String getFirstName()
{
    return firstName;
}
```

The code for the User class (continued)

```
    public void setLastName(String lastName)
    {
        this.lastName = lastName;
    }

    public String getLastName()
    {
        return lastName;
    }

    public void setEmailAddress(String emailAddress)
    {
        this.emailAddress = emailAddress;
    }

    public String getEmailAddress()
    {
        return emailAddress;
    }
}
```

The code for the UserIO class

```
package data;

import java.io.*;
import business.User;

public class UserIO
{
    public static void add(User user, String filepath)
    throws IOException
    {
        File file = new File(filepath);
        PrintWriter out = new PrintWriter(
                new FileWriter(file, true));
        out.println(user.getEmailAddress()+ "|"
                + user.getFirstName() + "|"
                + user.getLastName());
        out.close();
    }
}
```

The code for a JSP that uses the User and UserIO classes

```
<!DOCTYPE HTML PUBLIC
   "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
    <title>Murach's Java Servlets and JSP</title>
</head>
<body>
    <!-- import packages and classes needed by scripts -->
    <%@ page import="business.*, data.*" %>

    <%
        // get parameters from the request
        String firstName =
            request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String emailAddress =
            request.getParameter("emailAddress");
```

The code for the JSP (continued)

```
        // get the real path for the EmailList.txt file
        ServletContext sc = this.getServletContext();
        String path =
            sc.getRealPath("/WEB-INF/EmailList.txt");

        // use regular Java objects
        User user = new User(firstName, lastName,
            emailAddress);
        UserIO.add(user, path);
    %>

    <h1>Thanks for joining our email list</h1>

    <p>Here is the information that you entered:</p>

    <table cellspacing="5" cellpadding="5" border="1">
        <tr>
            <td align="right">First name:</td>
            <td><%= user.getFirstName() %></td>
        </tr>
```

The code for the JSP (continued)

```
        <tr>
            <td align="right">Last name:</td>
            <td><%= user.getLastName() %></td>
        </tr>
        <tr>
            <td align="right">Email address:</td>
            <td><%= user.getEmailAddress() %></td>
        </tr>
    </table>

    <p>To enter another email address, click on the Back <br
    button in your browser or the Return button shown <br>
    below.</p>

    <form action="join_email_list.html" method="post">
        <input type="submit" value="Return">
    </form>

</body>
</html>
```

The five types of JSP tags

Tag	Name	Purpose
`<% %>`	JSP scriptlet	To insert a block of Java statements.
`<%= %>`	JSP expression	To display the string value of an expression.
`<%@ %>`	JSP directive	To set conditions that apply to the entire JSP.
`<%-- --%>`	JSP comment	To tell the JSP engine to ignore code.
`<%! %>`	JSP declaration	To declare instance variables and methods for a JSP.

JSP code that imports Java classes

```
<%@ page import="business.*, data.*, java.util.Date" %>
<%
    // get parameters from the request
    String firstName = request.getParameter("firstName");
    String lastName = request.getParameter("lastName");
    String emailAddress =
        request.getParameter("emailAddress");

    // get a relative file name
    ServletContext sc = this.getServletContext();
    String path =
        sc.getRealPath("/WEB-INF/EmailList.txt");

    // use regular Java objects
    User user =
        new User(firstName, lastName, emailAddress);
    UserIO.add(user, path);
%>

<p>This email address was added to our list on
    <%= new Date() %></p>
```

How to import classes

- To define the conditions that the JSP engine should follow when converting a JSP into a servlet, you can use a *JSP directive*.
- To import classes in a JSP, you use the import attribute of the *page directive*. This makes the imported classes available to the entire page.

An HTML comment in a JSP

```
<!--
<p>This email address was added to our list on
    <%= new Date() %></p>
-->
```

A JSP comment

```
<%--
<p>This email address was added to our list on
    <%= new Date() %></p>
--%>
```

Java comments in a JSP scriptlet

```
<%
    // get parameters from the request
    String firstName = request.getParameter("firstName");
    String lastName = request.getParameter("lastName");
    String emailAddress =
        request.getParameter("emailAddress");

    /*
    User user =
        new User(firstName, lastName, emailAddress);
    UserIO.add(user, path);
    */
%>
```

How to code comments in a JSP

- When you code HTML comments, the comments are compiled and executed, but the browser doesn’t display them.
- When you code *JSP comments*, the comments aren’t compiled or executed.
- When you code Java comments within a scriptlet, the comments aren’t compiled or executed.

JSP code that declares an instance variable and a method

```
<%-- import any packages needed by the page --%>
<%@ page import="business.*, data.*, java.util.Date,
    java.io.*" %>
<%!
    // declare an instance variable for the page
    int globalCount = 0;   // this is not thread-safe
%>
<%!
    // declare a method for the page
    public void add(User user, String filename)
            throws IOException
    {
        PrintWriter out = new PrintWriter(
                new FileWriter(filename, true));
        out.println(user.getEmailAddress()+ "|"
                + user.getFirstName() + "|"
                + user.getLastName());
        out.close();
    }
%>
```

JSP code that declares an instance variable and a method (continued)

```
<%
    String firstName = request.getParameter("firstName");
    String lastName = request.getParameter("lastName");
    String emailAddress =
        request.getParameter("emailAddress");

    ServletContext sc = getServletContext();
    String path = sc.getRealPath("/WEB-INF/EmailList.txt");

    User user = new User(firstName, lastName, emailAddress);

    // use the declared method
    this.add(user, path);

    // update the instance variable
    globalCount++;  // this is not thread-safe
%>
.
.
.
```

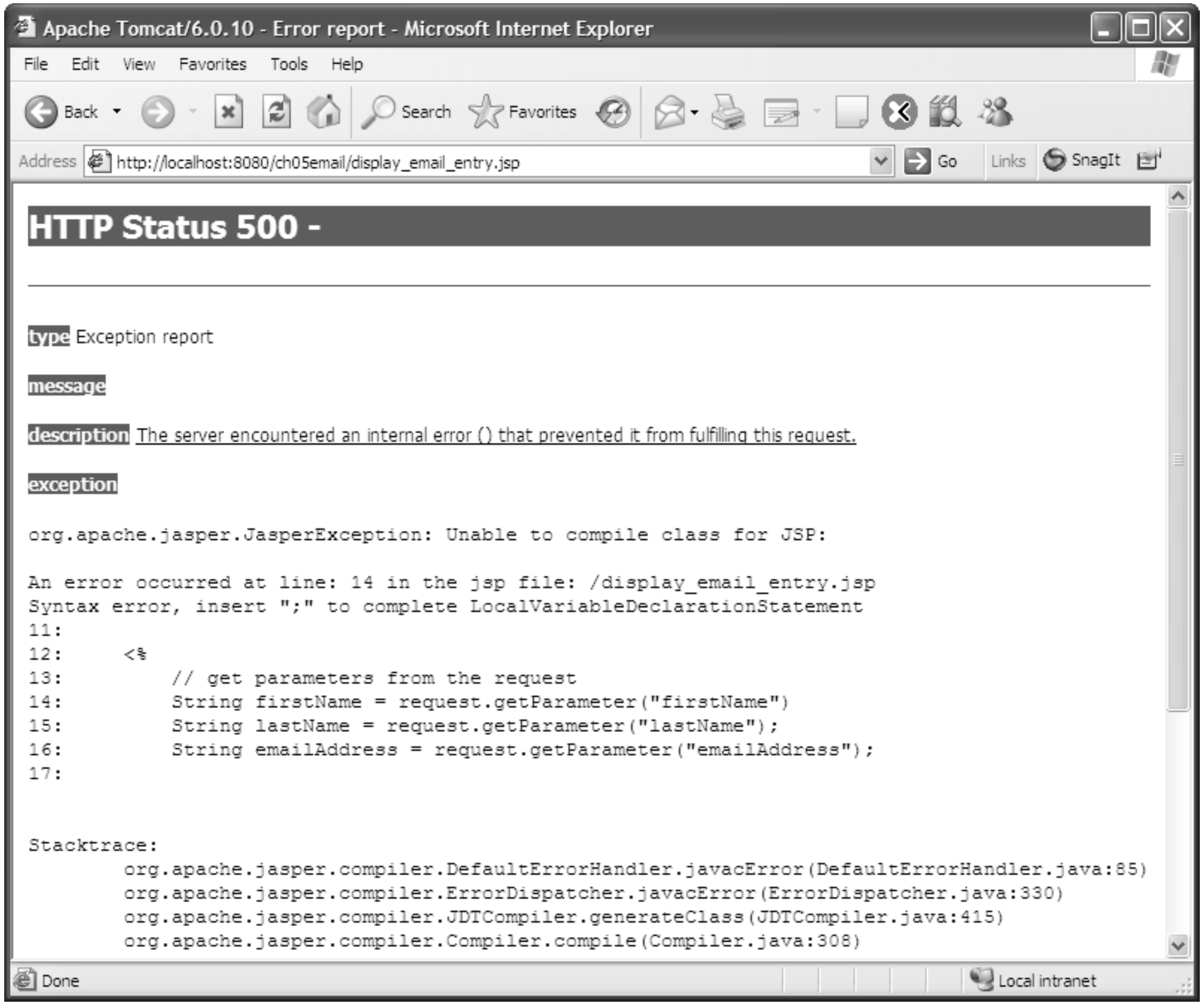
JSP code that declares an instance variable and a method (continued)

```
<p>
    This email address was added to our list on
        <%= new Date() %><br>
    This page has been accessed <%= globalCount %> times.
</p>
```

How to declare instance variables and methods

- You can use *JSP declarations* to declare instance variables and methods.
- Since instance variables aren't *thread-safe*, two threads may conflict when they try to read, modify, and update the same instance variable at the same time.
- In general, you should avoid coding instance variables for JSPs and servlets. Instead, you should use other thread-safe techniques for working with global variables.
- In general, you should avoid coding methods within JSPs. Instead, you should use some combination of JSPs, servlets, and regular Java classes.

An error page for a common JSP error


Apache Tomcat/6.0.10 - Error report - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/ch05email/display_email_entry.jsp
Go Links SnagIt
Done
Local intranet

HTTP Status 500 -

type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

```
org.apache.jasper.JasperException: Unable to compile class for JSP:

An error occurred at line: 14 in the jsp file: /display_email_entry.jsp
Syntax error, insert ";" to complete LocalVariableDeclarationStatement
11:
12:     <%
13:         // get parameters from the request
14:         String firstName = request.getParameter("firstName")
15:         String lastName = request.getParameter("lastName");
16:         String emailAddress = request.getParameter("emailAddress");
17:


Stacktrace:
        org.apache.jasper.compiler.DefaultErrorHandler.javacError(DefaultErrorHandler.java:85)
        org.apache.jasper.compiler.ErrorDispatcher.javacError(ErrorDispatcher.java:330)
        org.apache.jasper.compiler.JDTCompiler.generateClass(JDTCompiler.java:415)
        org.apache.jasper.compiler.Compiler.compile(Compiler.java:308)
```

Common JSP errors

- HTTP Status 404 – File Not Found Error
- HTTP Status 500 – Internal Server Error

Tips for fixing JSP errors

- Make sure the Tomcat server is running.
- Make sure that the URL is valid and that it points to the right location for the requested page.
- Make sure all of the HTML, JSP, and Java class files are in the correct locations.
- Read the error page carefully to get all available information about the error.

The JSP work directory for ch05email application

```
C:\tomcat\work\Catalina\localhost\ch05email\org\apache\jsp
```

Part of the servlet class that's generated from the JSP for the Email List application

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import business.*;
import data.*;
import java.util.Date;

public final class display_005femail_005fentry_jsp extends
org.apache.jasper.runtime.HttpJspBase
implements org.apache.jasper.runtime.JspSourceDependent {
      ...
```

Part of the servlet class (continued)

```
public void _jspService(HttpServletRequest request,
HttpServletResponse response)
throws java.io.IOException, ServletException {
    ...
    response.setContentType("text/html");
    ...
    out.write("<html>\n");
    out.write("<head>\n");
    out.write("    <title>Murach's Java Servlets and
        JSP</title>\n");
    out.write("</head>\n");
    out.write("<body>\n");
    ...
      // get parameters from the request
      String firstName =
          request.getParameter("firstName");
      String lastName = request.getParameter("lastName");
      String emailAddress =
          request.getParameter("emailAddress");
```

Part of the servlet class (continued)

```
        // get the real path for the emaillist file
        ServletContext sc = this.getServletContext();
        String path =
            sc.getRealPath("/WEB-INF/EmailList.txt");

        // use regular Java objects
        User user =
            new User(firstName, lastName, emailAddress);
        UserIO.add(user, path);
      ...
      out.write("    <table cellspacing=\"5\"
                     cellpadding=\"5\"
                     border=\"1\">\n");
      out.write("        <tr>\n");
      out.write("            <td align=\"right\">
                             First name:</td>\n");
      out.write("            <td>");
      out.print( user.getFirstName() );
      out.write("</td>\n");
      out.write("        </tr>\n");
      ...
```

Part of the servlet class (continued)

```
        out.write("</body>\n");
        out.write("</html>");
        ...
  }
}
```