

Chapter 6

How to develop servlets

Objectives

Applied

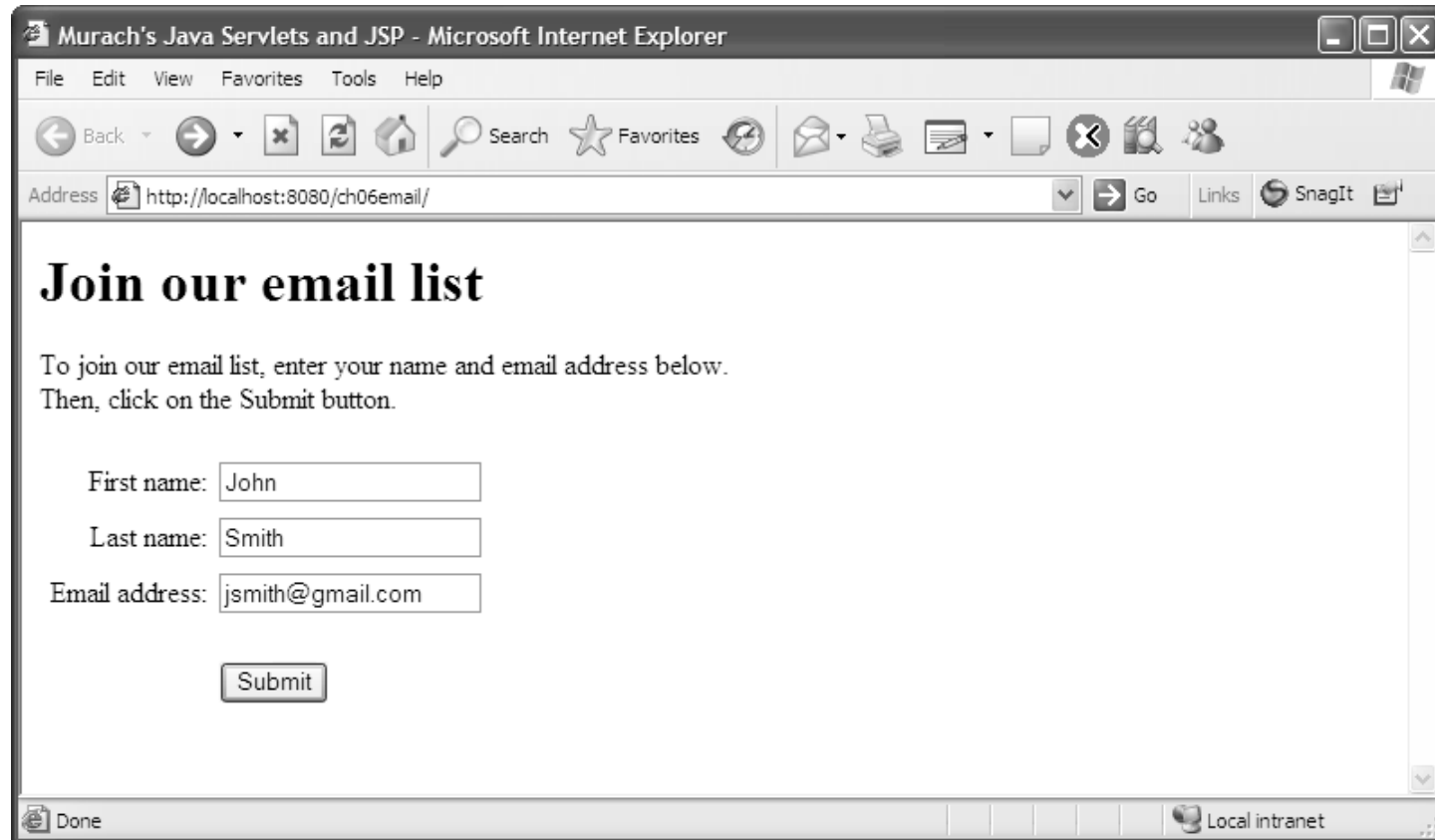
- Code and test servlets that require any of the features presented in this chapter.
- Use the web.xml file for an application to map a servlet to a URL pattern.
- Write debugging data for a servlet to either the console or a log file.

Objectives (cont.)

Knowledge

- In general terms, describe servlets and their use of request and response objects.
- Describe servlet mapping.
- Describe the use of the init, doGet, doPost, and destroy methods in a servlet.
- Describe the use of debugging data that's written to the console or a log file.

The HTML page for an Email List application



The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads 'Murach's Java Servlets and JSP - Microsoft Internet Explorer'. The menu bar includes 'File', 'Edit', 'View', 'Favorites', 'Tools', and 'Help'. The toolbar contains icons for Back, Forward, Stop, Reload, Home, Search, Favorites, and other standard browser functions. The address bar shows the URL 'http://localhost:8080/ch06email/'. The main content area displays the following HTML form:

Join our email list

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

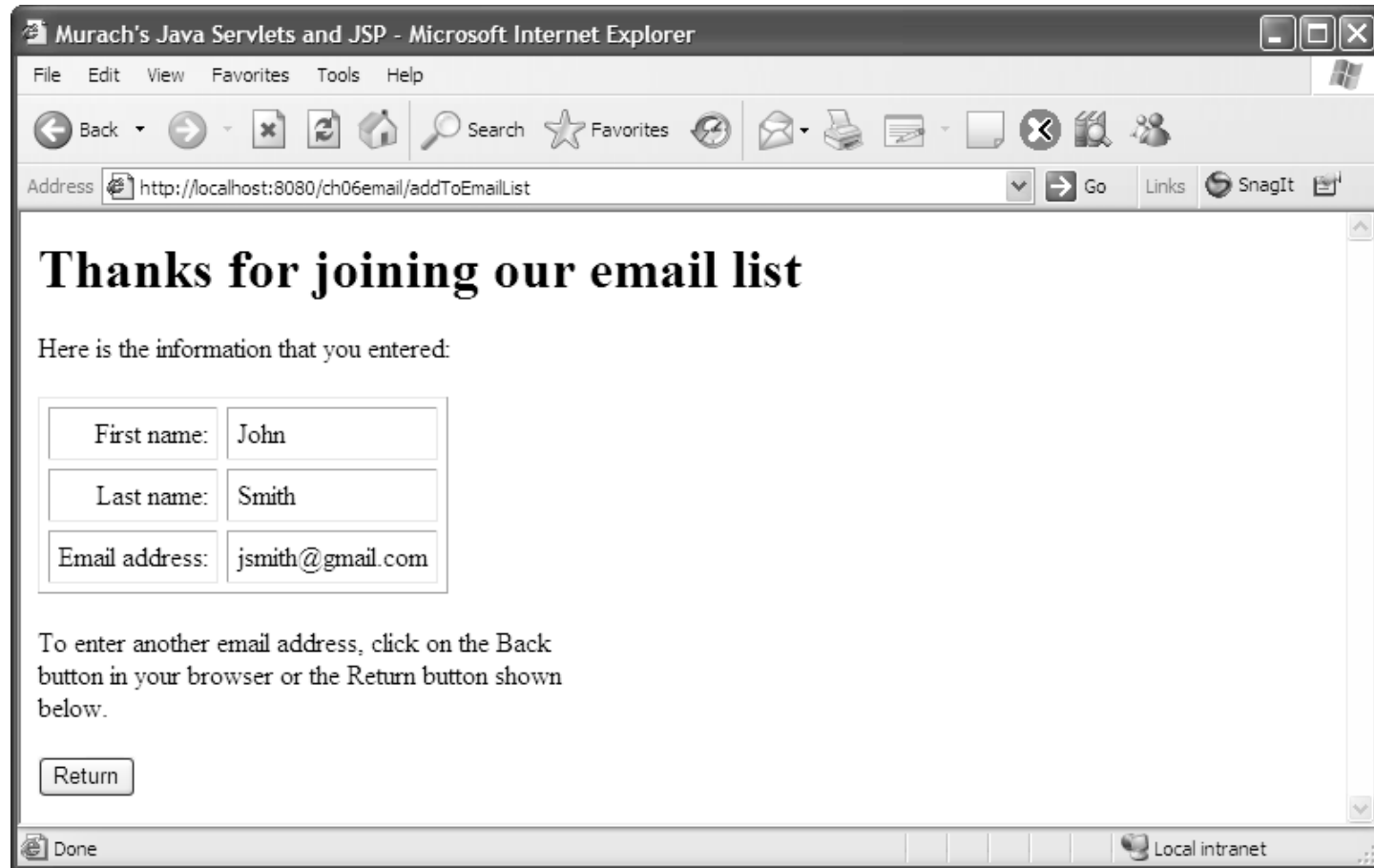
First name:

Last name:

Email address:

The status bar at the bottom shows 'Done' and 'Local intranet'.

The servlet page for an Email List application



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="  addToEmailList" method="post">
    <table cellpadding="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 Servlet

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

The code for the AddToEmailListServlet class

```
package email;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AddToEmailListServlet extends HttpServlet
{
    protected void doPost(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException
    {
        // get parameters from the request
        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String emailAddress = request.getParameter("emailAddress");
    }
}
```

Code for the AddToEmailListServlet class (cont.)

```
// send response to browser
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
out.println(
    "<!doctype html public \"-//W3C//DTD HTML 4.0 \"
+ \"    Transitional//EN\">\n"
+ "<html>\n"
+ "<head>\n"
+ "  <title>Murach's Java Servlets and JSP</title>\n"
+ "</head>\n"
+ "<body>\n"
+ "<h1>Thanks for joining our email list</h1>\n"
+ "<p>Here is the information that you entered:</p>\n"
+ "  <table cellpadding=\"5\" cellspacing=\"5\" \"
+ "    border=\"1\">\n"
+ "    <tr><td align=\"right\">First name:</td>\n"
+ "      <td>" + firstName + "</td>\n"
+ "    </tr>\n"
```

Code for the AddToEmailListServlet class (cont.)

```
+ "    <tr><td align=\"right\">Last name:</td>\n"
+ "        <td>" + lastName + "</td>\n"
+ "    </tr>\n"
+ "    <tr><td align=\"right\">Email address:</td>\n"
+ "        <td>" + emailAddress + "</td>\n"
+ "    </tr>\n"
+ " </table>\n"
+ "<p>To enter another email address, click on the "
+ "Back <br>\n"
+ "button in your browser or the Return button shown <br>\n"
+ "below.</p>\n"
+ "<form action=\"join_email_list.html\" "
+ "    method=\"post\">\n"
+ "    <input type=\"submit\" value=\"Return\">\n"
+ "</form>\n"
+ "</body>\n"
+ "</html>\n");

out.close();
}
```

Code for the AddToEmailListServlet class (cont.)

```
protected void doGet(  
    HttpServletRequest request,  
    HttpServletResponse response)  
    throws ServletException, IOException  
{  
    doPost(request, response);  
}  
}
```

The web.xml file with the mapping for the servlet

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

  <!-- the definitions for the servlet -->
  <servlet>
    <servlet-name>AddToEmailListServlet</servlet-name>
    <servlet-class>email.AddToEmailListServlet</servlet-class>
  </servlet>

  <!-- the mapping for the servlets -->
  <servlet-mapping>
    <servlet-name>AddToEmailListServlet</servlet-name>
    <url-pattern>/addToEmailList</url-pattern>
  </servlet-mapping>
```

The web.xml file with the mapping for the servlet (cont.)

```
<!-- other configuration settings for the application -->
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
<welcome-file-list>
    <welcome-file>join_email_list.html</welcome-file>
</welcome-file-list>
</web-app>
```

The structure for a simple servlet that returns HTML

```
package murach;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class TestServlet extends HttpServlet
{
    public void doPost(HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException
    {
        //business processing

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>HTML from servlet</h1>");
        out.close();
    }
}
```

The structure for a simple servlet that returns HTML (cont.)

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                  throws IOException, ServletException
{
    doPost(request, response);
}
}
```


How to code a servlet

- In practice, all servlets extend the `HttpServlet` class. To extend this class, the servlet must import the `java.io`, `javax.servlet`, and `javax.servlet.http` packages.
- The `doGet` method overrides the `doGet` method of the `HttpServlet` class and processes all HTTP requests that use the Get method.
- The `doPost` method overrides the `doPost` method of the `HttpServlet` class and processes all HTTP requests that use the Post method.
- The `doGet` and `doPost` methods use two objects that are passed to it by the web server: (1) the `HttpServletRequest` object, or the *request object*, and (2) the `HttpServletResponse` object, or the *response object*.

How to code a servlet (cont.)

- The `setContentType` method of the response object sets the *content type* of the response that's returned to the browser. Then, the `getWriter` method of the response object returns a `PrintWriter` object that can be used to send HTML to the browser.
- Before you can create a `PrintWriter` object, you must set the content type. This allows the `getWriter` method to return a `PrintWriter` object that uses the proper content type.

XML tags that add servlet mapping to the web.xml file

```
<!-- the definitions for the servlets -->
<servlet>
    <servlet-name>AddToEmailListServlet</servlet-name>
    <servlet-class>
        email.AddToEmailListServlet</servlet-class>
</servlet>
<servlet>
    <servlet-name>TestServlet</servlet-name>
    <servlet-class>email.TestServlet</servlet-class>
</servlet>

<!-- the mapping for the servlets -->
<servlet-mapping>
    <servlet-name>AddToEmailListServlet</servlet-name>
    <url-pattern>/addToEmailList</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>AddToEmailListServlet</servlet-name>
    <url-pattern>/email/*</url-pattern>
</servlet-mapping>
```

XML tags that add servlet mapping to the web.xml file (cont.)

```
<servlet-mapping>  
    <servlet-name>TestServlet</servlet-name>  
    <url-pattern>/test</url-pattern>  
</servlet-mapping>
```

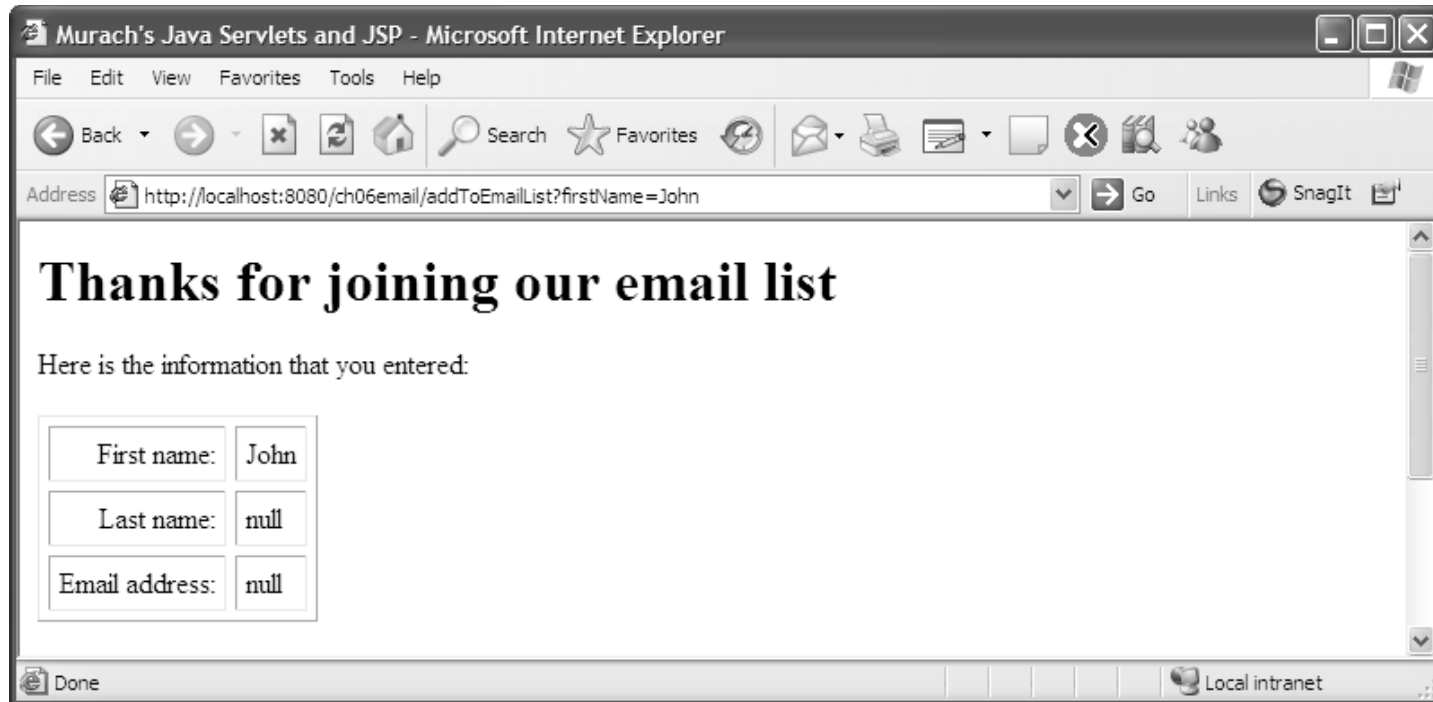
XML elements for working with servlet mapping

Element	Description
<code><servlet-class></code>	Specifies the class for the servlet.
<code><servlet-name></code>	Specifies a unique name for the servlet that's used to identify the servlet within the web.xml file. This element is required for both the servlet element and the servlet-mapping element and maps each servlet-mapping element to a servlet element.
<code><url-pattern></code>	Specifies the URL or URLs that are mapped to the specified servlet. This pattern must begin with a front slash, but the URL pattern can specify a virtual directory or file that doesn't actually exist.

Some URL pattern examples

URL pattern	Description
<code>/addToEmailList</code>	Specifies the addToEmailList URL in the root directory of the application.
<code>/email/*</code>	Specifies any URL in the email directory.
<code>/email/add</code>	Specifies the add URL in the email directory.
<code>/email/add.jsp</code>	Specifies the add.jsp URL in the email directory.

How to request the mapped servlet



A Form tag that requests the servlet

```
<form action="addToEmailList" method="post">
```

The URL displayed in the browser

```
http://localhost:8080/ch06email/addToEmailList
```

Another Form tag that requests the servlet

```
<form action="email/addToList" method="post">
```

The URL displayed in the browser

```
http://localhost:8080/ch06email/email/addToList
```

An A tag that requests a servlet

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

A URL that requests a servlet

```
http://localhost:8080/ch06email/addToEmailList?firstName=John
```


Five common methods of a servlet

```
public void init() throws ServletException{}
```

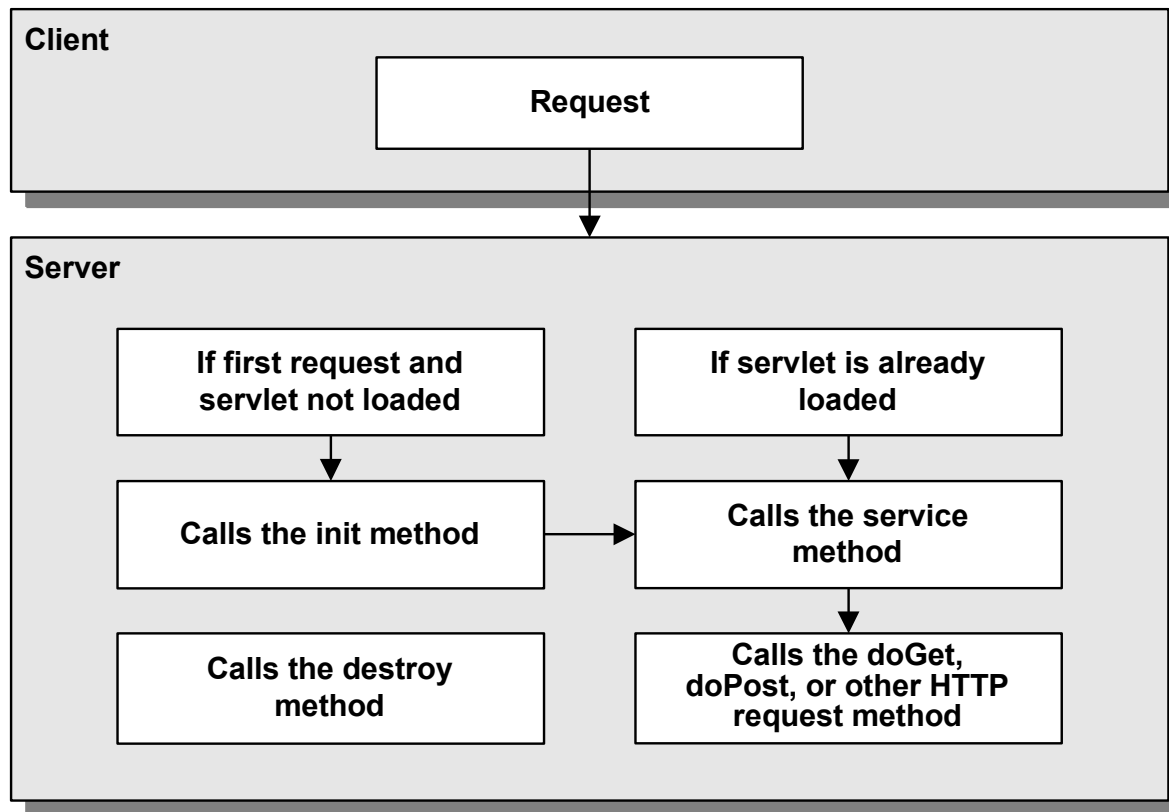
```
public void service(HttpServletRequest request,  
                    HttpServletResponse response)  
    throws IOException, ServletException{}
```

```
public void doGet(HttpServletRequest request,  
                  HttpServletResponse response)  
    throws IOException, ServletException{}
```

```
public void doPost(HttpServletRequest request,  
                   HttpServletResponse response)  
    throws IOException, ServletException{}
```

```
public void destroy(){}
```

How the server handles a request for a servlet



The life cycle of a servlet

- A server loads and initializes the servlet by calling the init method.
- The servlet handles each browser request by calling the service method. This method then calls another method to handle the specific HTTP request type.
- The server removes the servlet by calling the destroy method. This occurs either when the servlet has been idle for some time or when the server is shut down.

A method of the **GenericServlet** class

Method	Description
<code>getServletContext()</code>	Returns a <code>ServletContext</code> object that contains information about the application's context.

A method of the **ServletContext** class for working with paths

Method	Description
<code>getRealPath(String path)</code>	Returns a <code>String</code> 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
```

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 the AddToEmailListServlet class using User and UserIO classes

```
package email;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

import business.User;
import data.UserIO;

public class AddToEmailListServlet extends HttpServlet
{
    protected void doPost(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException
    {
        // 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 = getServletContext();
        String path = sc.getRealPath("/WEB-INF/EmailList.txt");
```

Code for the AddToEmailListServlet class (cont.)

```
// use regular Java objects to write the data to a file
User user = new User(firstName, lastName, emailAddress);
UserIO.add(user, path);
```

```
// send response to browser
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
out.println(
    "<!doctype html public \"-//W3C//DTD HTML 4.0 \"
+ \"    Transitional//EN\">\n"
+ "<html>\n"
+ "<head>\n"
+ "  <title>Murach's Java Servlets and JSP</title>\n"
+ "</head>\n"
+ "<body>\n"
+ "<h1>Thanks for joining our email list</h1>\n"
+ "<p>Here is the information that you entered:</p>\n"
+ "  <table cellpadding=\"5\" cellspacing=\"5\" \"
+ "    border=\"1\">\n"
+ "    <tr><td align=\"right\">First name:</td>\n"
+ "      <td>" + firstName + "</td>\n"
+ "    </tr>\n"
```

Code for the AddToEmailListServlet class (cont.)

```
+ "    <tr><td align=\"right\">Last name:</td>\n"
+ "        <td>" + lastName + "</td>\n"
+ "    </tr>\n"
+ "    <tr><td align=\"right\">Email address:</td>\n"
+ "        <td>" + emailAddress + "</td>\n"
+ "    </tr>\n"
+ " </table>\n"
+ "<p>To enter another email address, click on the "
+ "Back <br>\n"
+ "button in your browser or the Return button shown <br>\n"
+ "below.</p>\n"
+ "<form action=\"join_email_list.html\" "
+ "    method=\"post\">\n"
+ "    <input type=\"submit\" value=\"Return\">\n"
+ "</form>\n"
+ "</body>\n"
+ "</html>\n");

out.close();
}
```

Code for the AddToEmailListServlet class (cont.)

```
protected void doGet(  
    HttpServletRequest request,  
    HttpServletResponse response)  
    throws ServletException, IOException  
{  
    doPost(request, response);  
}  
}
```

Code that adds an instance variable to the EmailServlet class

```
package email;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AddToEmailListServlet2 extends HttpServlet
{
    // declare an instance variable for the page
    int globalCount;
    // instance variables are not thread-safe

    public void init() throws ServletException
    {
        globalCount = 0; // initialize the instance variable
    }
}
```

Code that adds an instance variable to the EmailServlet class (cont.)

```
protected void doPost(  
    HttpServletRequest request,  
    HttpServletResponse response)  
    throws ServletException, IOException  
{  
    // update global count variable  
    globalCount++;    // this is not thread-safe  
  
    // send response to browser  
    response.setContentType("text/html;charset=UTF-8");  
    PrintWriter out = response.getWriter();  
    out.println(  
        "<!doctype html public "  
        + "\"-//W3C//DTD HTML 4.0 Transitional//EN\">\n"  
        + "<html>\n"  
        + "<head>\n"  
        + "    <title>Murach's Java Servlets and "  
        + "JSP</title>\n"  
        + "</head>\n"  
        + "<body>\n"
```

Code that adds an instance variable to the EmailServlet class (cont.)

```
        + "<h1>Thanks for joining our email list</h1>\n"
        + "<p>This page has been accessed "
        +   globalCount + " times.</p>"
        + "</body>\n"
        + "</html>\n");

    out.close();
}
}
```


How to code instance variables

- An *instance variable* of a servlet belongs to the one instance of the servlet and is shared by any threads that request the servlet.
- Instance variables are not *thread-safe*. In other words, two threads may conflict when they try to read, modify, and update the same instance variable at the same time, which can result in lost updates or other problems.

Common servlet problems

Problem	Possible solutions
The servlet won't compile	<p>Make sure the compiler has access to the JAR files for all necessary APIs.</p> <p>Make sure the Java classes that you code are stored in the correct directories with the correct package statements.</p>
The servlet won't run	<p>Make sure the web server is running.</p> <p>Make sure you're using the correct URL.</p>
Changes to the servlet aren't showing up	<p>Make sure servlet reloading is on.</p> <p>Redeploy the application.</p> <p>Restart the server so it reloads all applications.</p>
The page doesn't display correctly	<p>Select the Source or Page Source command from your browser's View menu to view the HTML code and identify the problem. Then, you can fix the problem in the servlet.</p>

Code that prints debugging data to the console

```
public void doPost(HttpServletRequest request,
                  HttpServletResponse response)
                  throws IOException, ServletException
{
    //
    // code
    //
    String emailAddress =
        request.getParameter("emailAddress");
    System.out.println("AddToEmailListServlet emailAddress: "
        + emailAddress);
    //
    // code
    //
}
```

The output that's printed to the Tomcat console

```
AddToEmailListServlet emailAddress: jsmith@gmail.com
```

How to print debugging data to the console

- When you're testing an application on your local system, you can use the `println` method of the `System.out` or `System.err` objects to display debugging messages on the console for the servlet engine.
- When you use debugging messages to display variable values, it's a good practice to include the servlet name and the variable name so the messages are easy to interpret.
- When you use an IDE like NetBeans, debugging data that is written to a server console is also available in an output window.

Two methods of the HttpServlet class used to log errors

Method	Description
<code>log(String message)</code>	Writes the specified message to the server's log file.
<code>log(String message, Throwable t)</code>	Writes the specified message to the server's log file, followed by the stack trace for the exception.

Servlet code that prints the value of a variable to a log file

```
log("emailAddress=" + emailAddress);
```

The data that's written to the log file

```
Jun 29, 2007 6:26:04 PM
```

```
org.apache.catalina.core.ApplicationContext log
```

```
INFO: AddToEmailListServlet: emailAddress=jsmith@gmail.com
```

Servlet code that prints a stack trace to a log file

```
try
{
    UserIO.add(user, path);
}
catch(IOException e)
{
    log("An IOException occurred.", e);
}
```

The data that's written to the log file

```
Jun 29, 2007 6:30:38 PM org.apache.catalina.core.StandardWrapperValve invoke
WARNING: Servlet.service() for servlet AddToEmailListServlet threw exception
java.io.FileNotFoundException:
C:\murach\servlet_jsp\netbeans\ch06email\build\web\WEB-INF\EmailList.txt
(Access is denied)
    at java.io.FileOutputStream.openAppend(Native Method)
    at java.io.FileOutputStream.<init>(FileOutputStream.java:177)
    at java.io.FileWriter.<init>(FileWriter.java:90)
    at data.UserIO.add(UserIO.java:11)
    at email.AddToEmailListServlet.doPost(AddToEmailListServlet.java:38)
...
...
```

The location of a typical Tomcat log file

`C:\tomcat\logs\localhost.2007-06-29.log`

How to print debugging data to a log file

- You can use the log methods of the `HttpServlet` class to write debugging data to a web server's *log file*.
- A *stack trace* is the chain of method calls for any statement that calls a method.
- The data that's written by the log methods will vary from one server to another. The name and location of the log files will also vary from one server to another.
- When you use an IDE like NetBeans, the server log file is also available in an output window.