Chapter 2,5
How to develop servlets
A servlet that returns HTML

```java
package murach.email;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestServlet extends HttpServlet {
    
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        try {
            out.println("<h1>HTML from servlet</h1>"); 
        } finally {
            out.close();
        }
    }
}
```
A servlet that returns HTML (continued)

```java
@Override
protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        doPost(request, response);
    }
``` 

This example corresponds to our Servlet1 example
Next example: email list servlet

First look in Chap. 2 for basics on the email list servlet

The Chap2 email servlet is almost identical to the Chap5 ch05email example (online)
The HTML page that gets data from the user

This is just index.html in webapps/ch05email/index.html
The JSP that displays the data

This is displayed after the servlet runs.
Again: The HTML page that gets data from the user

This is just index.html in webapps/ch05email/index.html
Now let’s look at that HTML…
The index.html file (continued)

```html
<form action="emailList" method="post">
  <input type="hidden" name="action" value="add">

  <label>Email:</label>
  <input type="email" name="email" required><br>

  <label>First Name:</label>
  <input type="text" name="firstName" required><br>

  <label>Last Name:</label>
  <input type="text" name="lastName" required><br>

  <label>&nbsp;</label>
  <input type="submit" value="Join Now" id="submit">
</form>
</body>
</html>
```

- This is from Chap. 2, pg. 37, but relevant to Chap. 5 ch05email (its first page)
- Then POST body carries parameters action=add&email=js@foo.com &firstName=John&lastName=Smith for example
- The POST request will be handled by the servlet…
The EmailListServlet class

```java
package murach.email;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import murach.business.User;
import murach.data.UserDB;

public class EmailListServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
                         throws ServletException, IOException {

        String url = "/index.html";

        // get current action
        String action = request.getParameter("action");
        if (action == null) {
            action = "join";  // default action
        }
```
The EmailListServlet class (continued)

    // perform action and set URL to appropriate page
    if (action.equals("join")) {
        url = "/index.html";    // the "join" page
    }
    else if (action.equals("add")) {
        // get parameters from the request
        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String email = request.getParameter("email");

        // store data in User object and save User object in db
        User user = new User(firstName, lastName, email);
        UserDB.insert(user);

        // set User object in request object and set URL
        request.setAttribute("user", user);
        url = "/thanks.jsp";   // the "thanks" page
    }

    // forward request and response objects to specified URL
    getServletContext()
        .getRequestDispatcher(url)
        .forward(request, response);
The EmailListServlet class (continued)

```java
@Override
protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
                         throws ServletException, IOException {
        doPost(request, response);
    }
```
The web.xml file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">

    <servlet>
        <servlet-name>EmailListServlet</servlet-name>
        <servlet-class>murach.email.EmailListServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>EmailListServlet</servlet-name>
        <url-pattern>/emailList</url-pattern>
    </servlet-mapping>

    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>

</web-app>
```
The web.xml file (continued)

```xml
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>

</web-app>
```
The User class

```java
package murach.business;

import java.io.Serializable;

public class User implements Serializable {
    private String firstName;
    private String lastName;
    private String email;

    public User() {
        firstName = "";
        lastName = "";
        email = "";
    }

    public User(String firstName, String lastName, String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }
}
```
The User class (continued)

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

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

    public String getLastName() { 
        return lastName; 
    }

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

    public String getEmail() { 
        return email; 
    }

    public void setEmail(String email) { 
        this.email = email; 
    }
```
The thanks.jsp file

```html
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Murach's Java Servlets and JSP</title>
    <link rel="stylesheet" href="styles/main.css" type="text/css"/>
</head>

<body>
    <h1>Thanks for joining our email list</h1>
    <p>Here is the information that you entered:

    <label>Email:</label>
    <span>${user.email}</span>
    <label>First Name:</label>
    <span>${user.firstName}</span>
    <label>Last Name:</label>
    <span>${user.lastName}</span>
```
The thanks.jsp file (continued)

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

<form action="" method="get">
    <input type="hidden" name="action" value="join">
    <input type="submit" value="Return">
</form>

</body>
</html>
Ch05email Example: second page returned by JSP

Thanks for joining our email list

Here is the information that you entered:

Email:
First Name: John
Last Name: Smith

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

Return
Servlet concepts

- The doGet method processes all HTTP requests that use the GET method.
- The doPost method processes all HTTP requests that use the POST method.
- The doGet and doPost methods both accept (1) the HttpServletRequest object, or the request object, and (2) the HttpServletResponse object, or the response object.
**Two methods of the HttpServletRequest object**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getParameter(String param)</code></td>
<td>Returns the value of the specified parameter as a string if it exists or null if it doesn’t. Often, this is the value defined in the value attribute of the control in the HTML or JSP file.</td>
</tr>
<tr>
<td><code>getParameterValues(String param)</code></td>
<td>Returns an array of String objects containing all values that the given request parameter has or null if the parameter doesn’t have any values.</td>
</tr>
</tbody>
</table>
Servlet code that gets text from a text box

```java
String firstName = request.getParameter("firstName");
```

Servlet code that determines if a box is checked

```java
// returns the value or "on" if checked, null otherwise.
String rockCheckBox = request.getParameter("rock");
if (rockCheckBox != null) {
    // rock music was checked
}
```

Servlet code that reads and processes multiple values from a list box

```java
// returns the values of the items selected in a list box.
String[] selectedCountries = request.getParameterValues("country");
for (String country : selectedCountries) {
    // code that processes each country
}
```
# A method of the GenericServlet class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getServletContext()</code></td>
<td>Returns a ServletContext object that contains information about the application’s context.</td>
</tr>
</tbody>
</table>

# A method of the ServletContext class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getRealPath(String path)</code></td>
<td>Returns a String object for the absolute path of the specified relative path.</td>
</tr>
</tbody>
</table>
Code that gets the absolute path for a file

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

A more concise way to write the same code

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

A possible value for the real path variable

```
C:\murach\servlet_and_jsp\netbeans\book_apps\ch05email\build\web\WEB-INF\EmailList.txt
```
How to work with the ServletContext object

- Because servlets inherit the GenericServlet class, the `getServletContext` method is available to all servlets.

- You can use the ServletContext object to read global initialization parameters, to work with global variables, and to write data to log files.
## Two methods of the HttpServletRequest object

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setAttribute(String name, Object o)</code></td>
<td>Stores any object in the request as an attribute and specifies a name for the attribute. Attributes reset between requests.</td>
</tr>
<tr>
<td><code>getAttribute(String name)</code></td>
<td>Returns value of the specified attribute as an Object type. If no attribute exists for specified name, method returns a null value.</td>
</tr>
</tbody>
</table>
How to set a request attribute

User user = new User(firstName, lastName, email);
request.setAttribute("user", user);

How to get a request attribute

User user = (User) request.getAttribute("user");

How to set a request attribute for a primitive type

int id = 1;
request.setAttribute("id", new Integer(id));

How to get a request attribute for a primitive type

int id = (Integer) request.getAttribute("id");
### A method of the ServletContext object

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getRequestDispatcher(String path)</code></td>
<td>Returns a RequestDispatcher object for specified path.</td>
</tr>
</tbody>
</table>

### A method of the RequestDispatcher object

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>forward(ServletRequest request, ServletResponse response)</code></td>
<td>Forwards the request and response objects to another resource on the server (usually a JSP or servlet).</td>
</tr>
</tbody>
</table>
How to forward the request to an HTML page

```java
String url = "/index.html";
getServletContext().getRequestDispatcher(url)
    .forward(request, response);
```

How to forward the request to a JSP

```java
String url = "/thanks.jsp";
getServletContext().getRequestDispatcher(url)
    .forward(request, response);
```

How to forward the request to a servlet

```java
String url = "/cart/displayInvoice";
getServletContext().getRequestDispatcher(url)
    .forward(request, response);
```

Note the URL always starts with / and specifies a URL within the servlet context--it’s “context-relative”
Example: /thanks.jsp of ch05email is thanks.jsp in directory webapps/ch05email
Five common methods of a servlet

```java
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()
```
The lifecycle of a servlet

Note

- It’s generally considered a bad practice to override the service method. Instead, you should override a method like doGet or doPost to handle a specific type of HTTP request.
A servlet with an instance variable

package murach.email;

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

public class EmailListServlet extends HttpServlet {

    // declare an instance variable for the page
    private int globalCount; // not thread-safe

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

    We will only use immutable variables as servlet fields (instance variables), such as the service API object.
A servlet with an instance variable (continued)

```java
@Override
protected void doPost(
    HttpServletRequest request,
    HttpServletResponse response)
throws ServletException, IOException
{
    // update global count variable
    globalCount++;   // this is not thread-safe

    // the rest of the code goes here
}
```
Why you shouldn’t use instance variables in servlets

- 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*. 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.