B.Sc. In Software Development. Year 4. Semester I. Enterprise Development. Servlets.

LIMERICK INSTITUTE OF TECHNOLOGY
SCHOOL OF SCIENCE, ENGINEERING & I.T.
Department of Information Technology

Introduction

- Servlets technology is primarily designed for use with HTTP.
- Servlets are Java programs that run on a Web Server.
  - Used to process client requests or produce dynamic web pages.
  - A Web Server is just a program/piece of software that responds to requests from clients. Web Servers respond using HTTP.
  - HTTP is the main method of transferring information on the WWW using a request/response mechanism.

Creating a Servlet

- All Servlets that you create must extend the class `HttpServlet`.
- You need to override appropriate methods in the `HttpServlet` class to implement your servlet.
- The code listing on the upcoming slides displays a simple message on the users browser.

HttpServlet Class

- The HttpServlet class resides in the following package javax.http.servlet
- It is an abstract class.
- It must be extended/subclassed by all Servlets. A subclass of HttpServlet must override at least one method, usually one of the following:

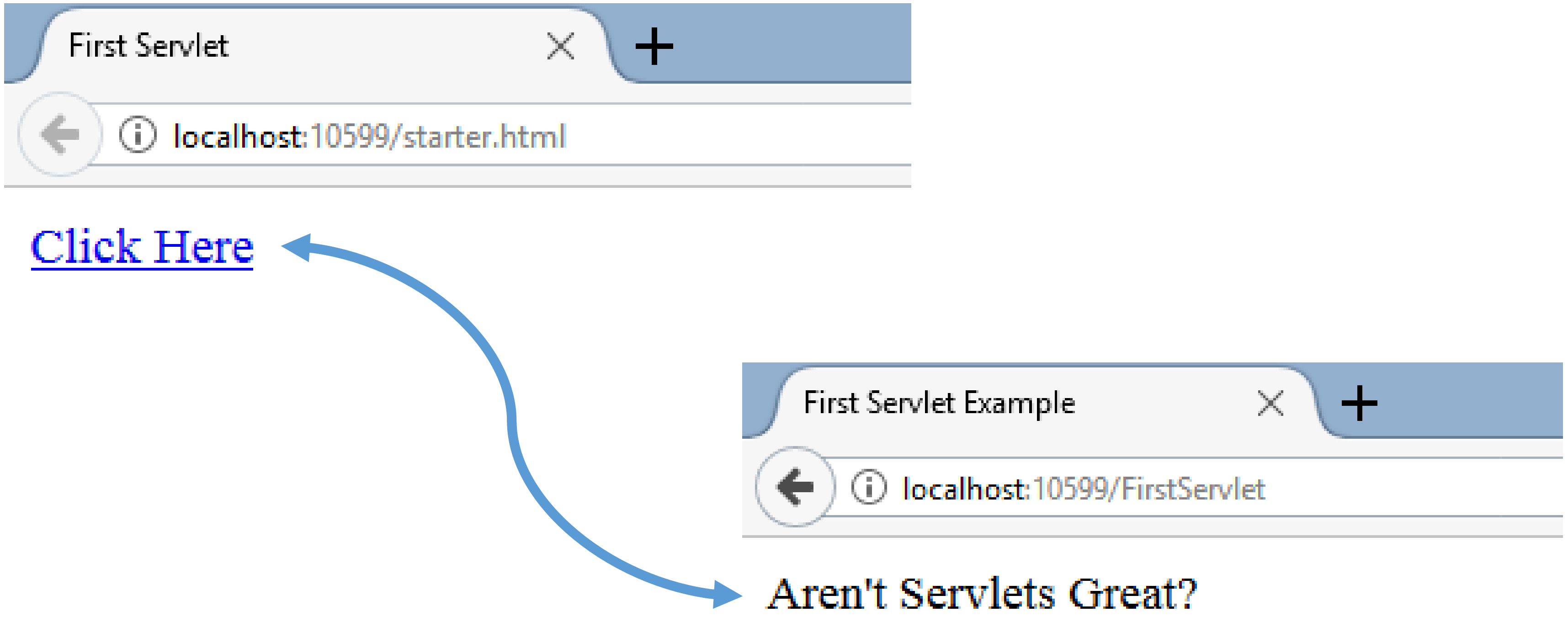
Methods:

doGet(HttpServletRequest req, HttpServletResponse resp): void

Used by your Servlet to handled get requests from clients.

doPost(HttpServletRequest req, HttpServletResponse resp): void
Used by your Servlet to handled post requests from clients.

Creating Your First Servlet



First Servlet

localhost:10599/starter.html

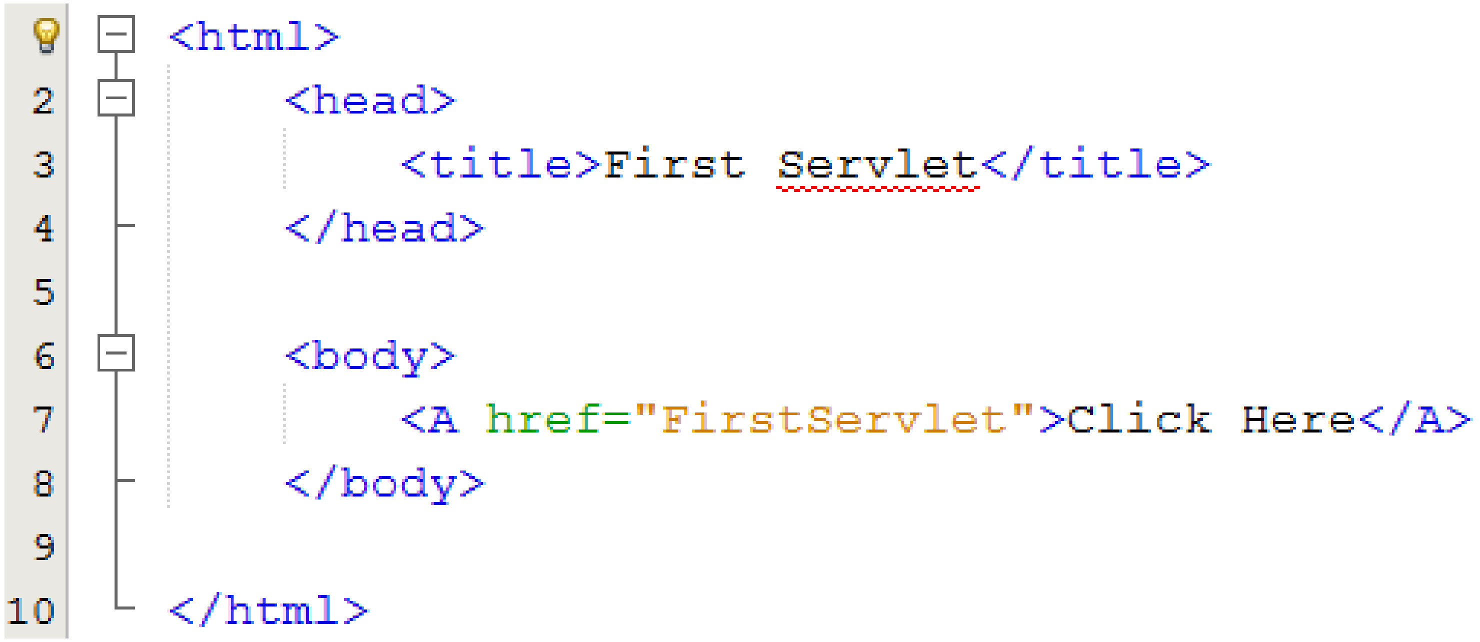
Click Here

First Servlet Example

localhost:10599/FirstServlet

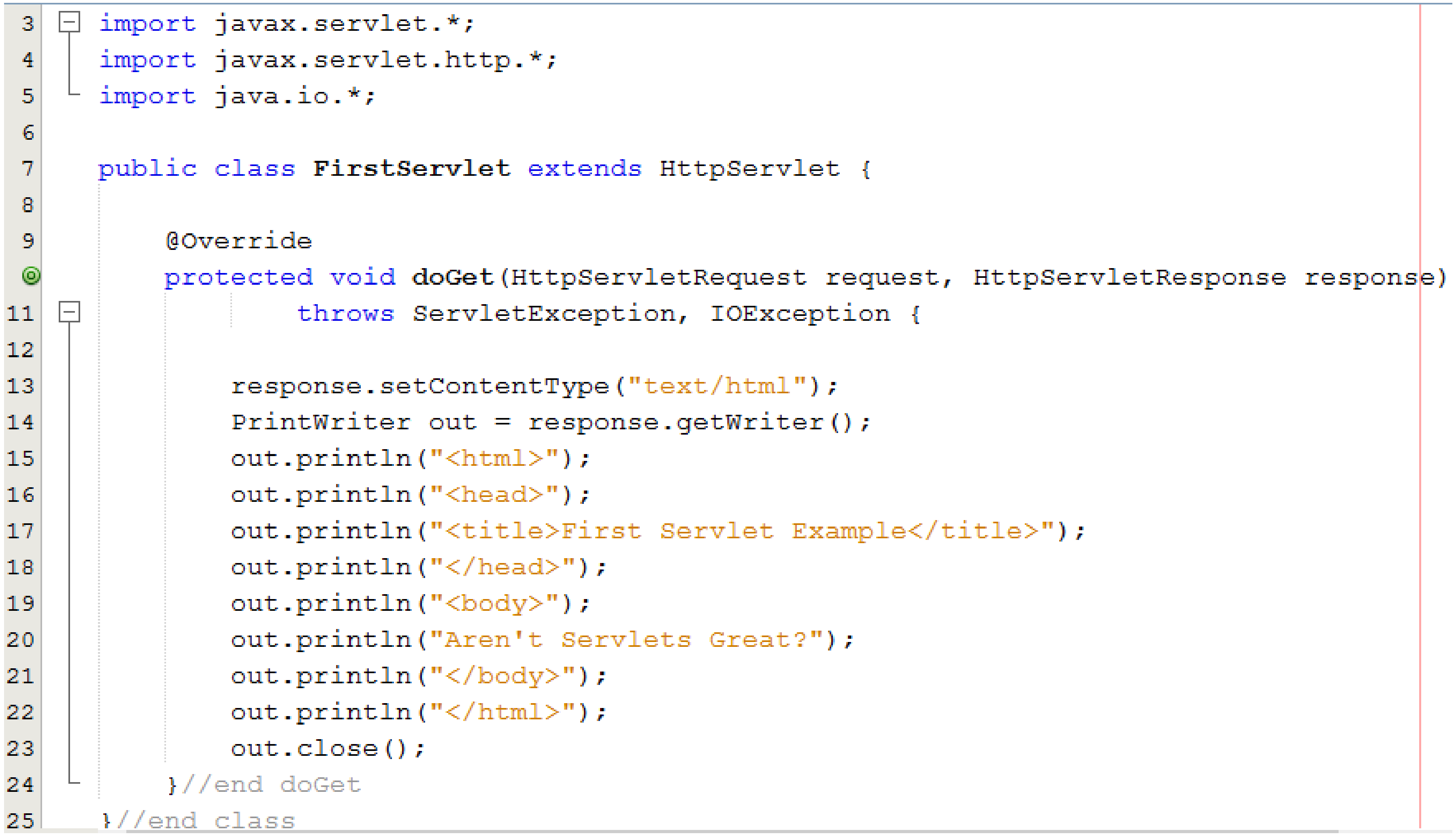
Aren't Servlets Great?

Creating Your First Servlet



```
<html>
    <head>
        <title>First Servlet</title>
    </head>

    <body>
        <A href="FirstServlet">Click Here</A>
    </body>

</html>
```

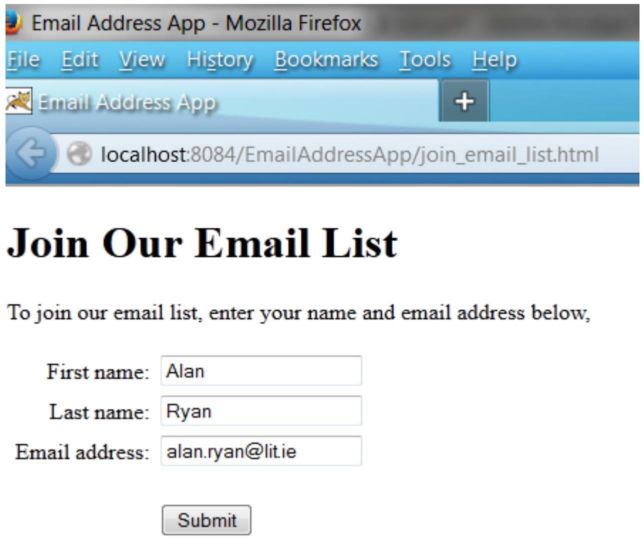
Creating Your First Servlet



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

public class FirstServlet extends HttpServlet {

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>First Servlet Example</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("Aren't Servlets Great?");
        out.println("</body>");
        out.println("</html>");
        out.close();
    }//end doGet
}//end class
```

Rewriting the address book app with a Servlet

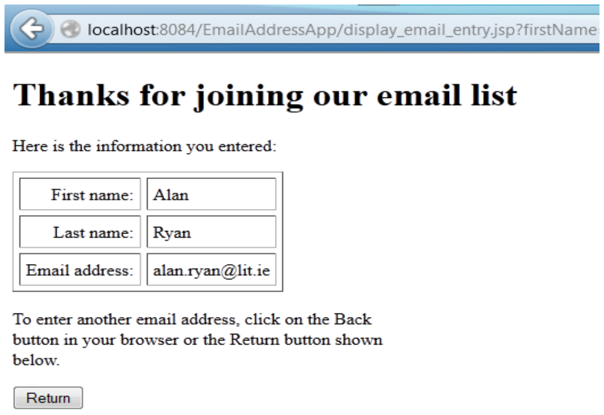
In the lecture on JSP's we built an application that allowed a user to join an email list. Recall:


Email Address App - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Email Address App
localhost:8084/EmailAddressApp/join_email_list.html

Join Our Email List

To join our email list, enter your name and email address below,

First name:	Alan
Last name:	Ryan
Email address:	alan.ryan@lit.ie


localhost:8084/EmailAddressApp/display_email_entry.jsp?firstName

Thanks for joining our email list

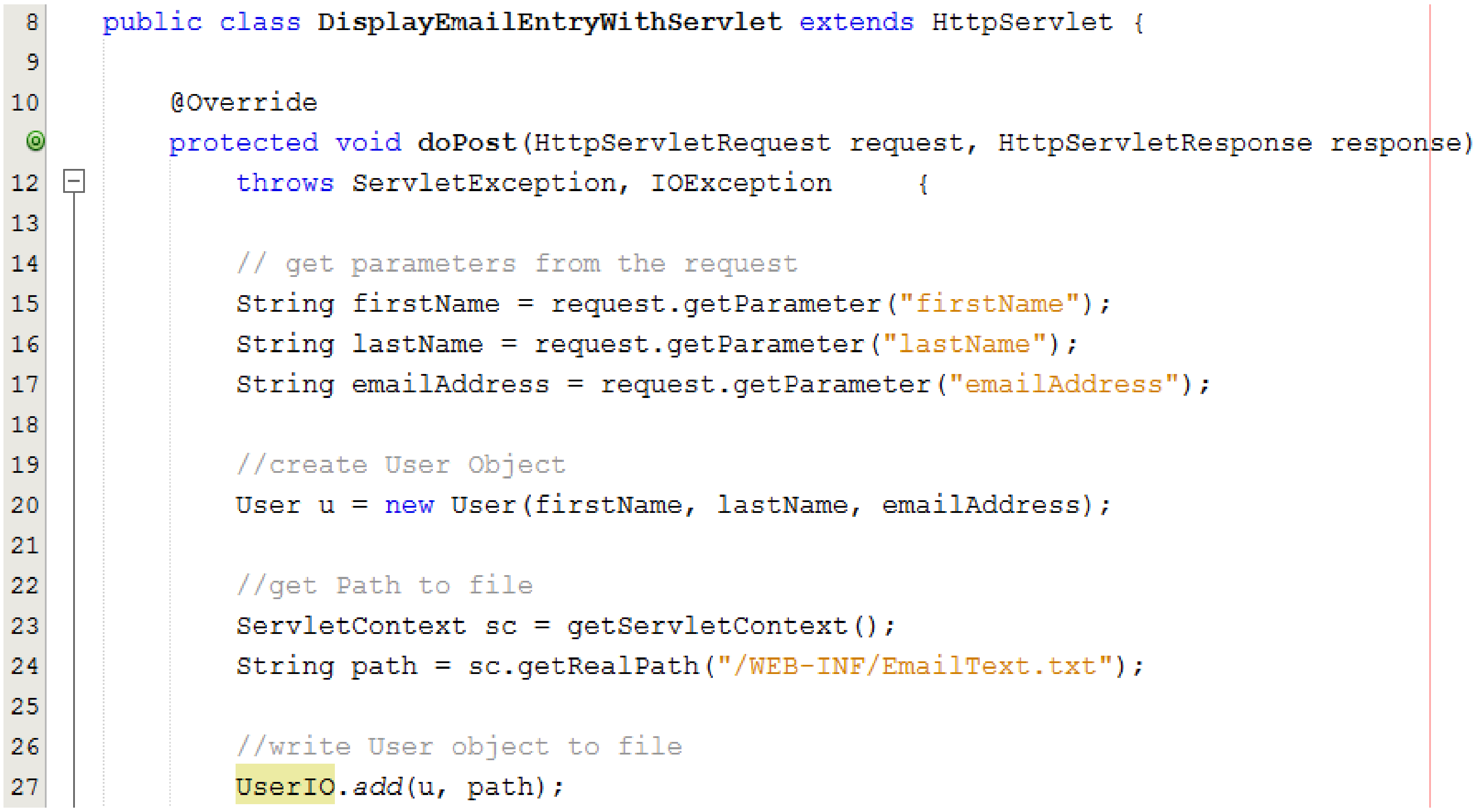
Here is the information you entered:

First name:	Alan
Last name:	Ryan
Email address:	alan.ryan@lit.ie

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

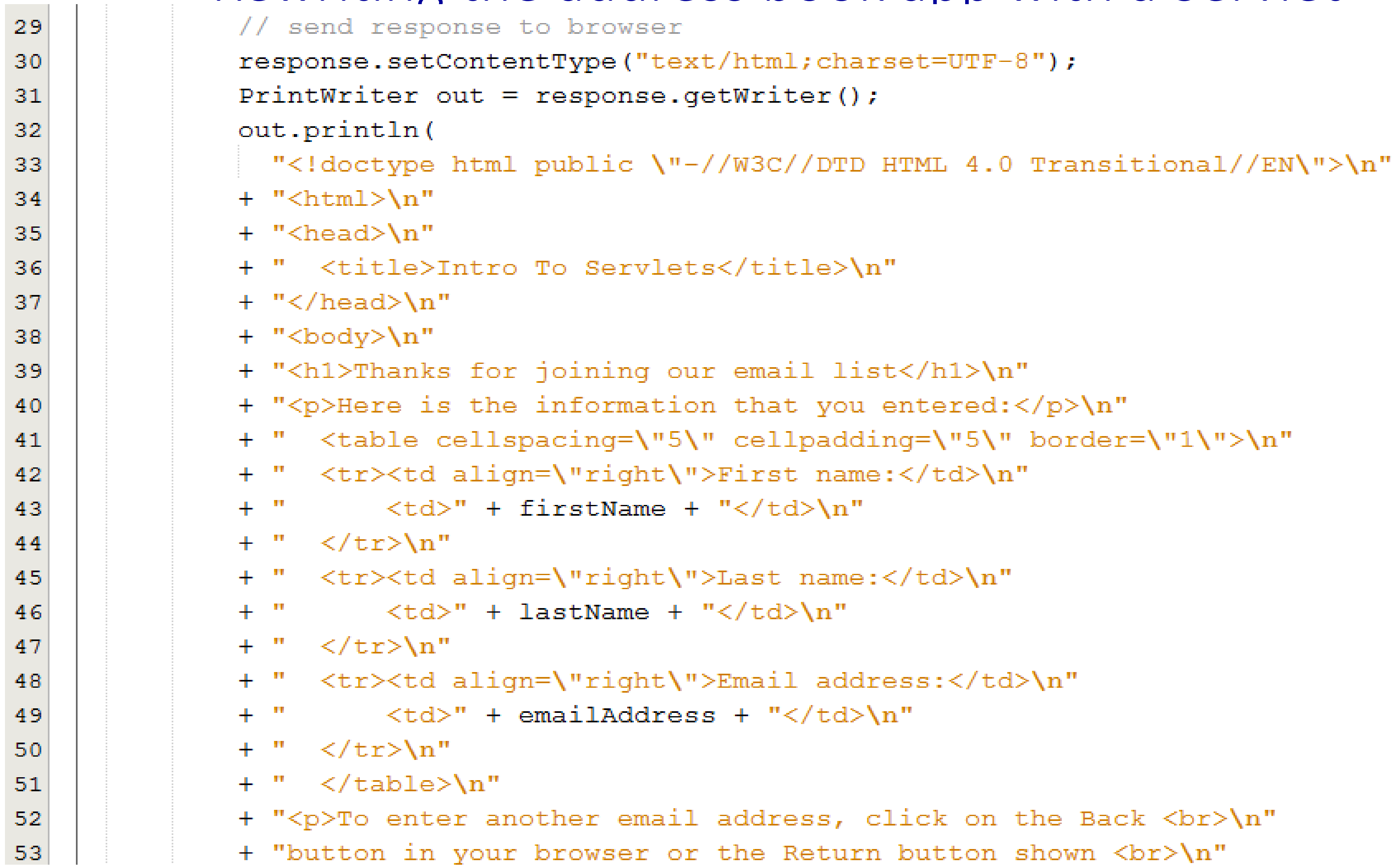
I'm now going to show how to rewrite that example using a Servlet (note: the HTML file along with the User and UserIO classes remain the same from the first example).

Rewriting the address book app with a Servlet



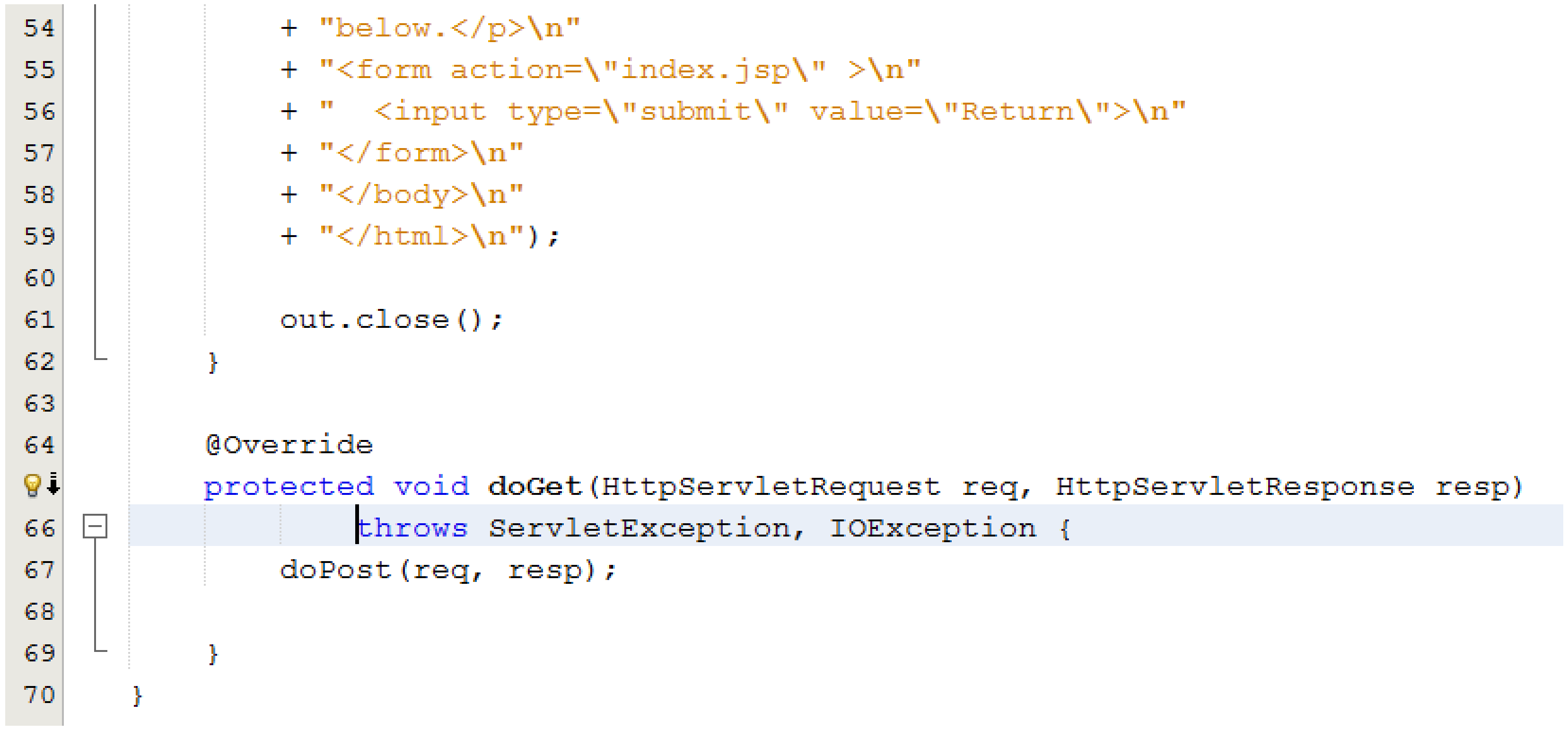
```java
public class DisplayEmailEntryWithServlet extends HttpServlet {

    @Override
    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");

        //create User Object
        User u = new User(firstName, lastName, emailAddress);

        //get Path to file
        ServletContext sc = getServletContext();
        String path = sc.getRealPath("/WEB-INF/EmailText.txt");

        //write User object to file
        UserIO.add(u, path);
```

Rewriting the address book app with a Servlet



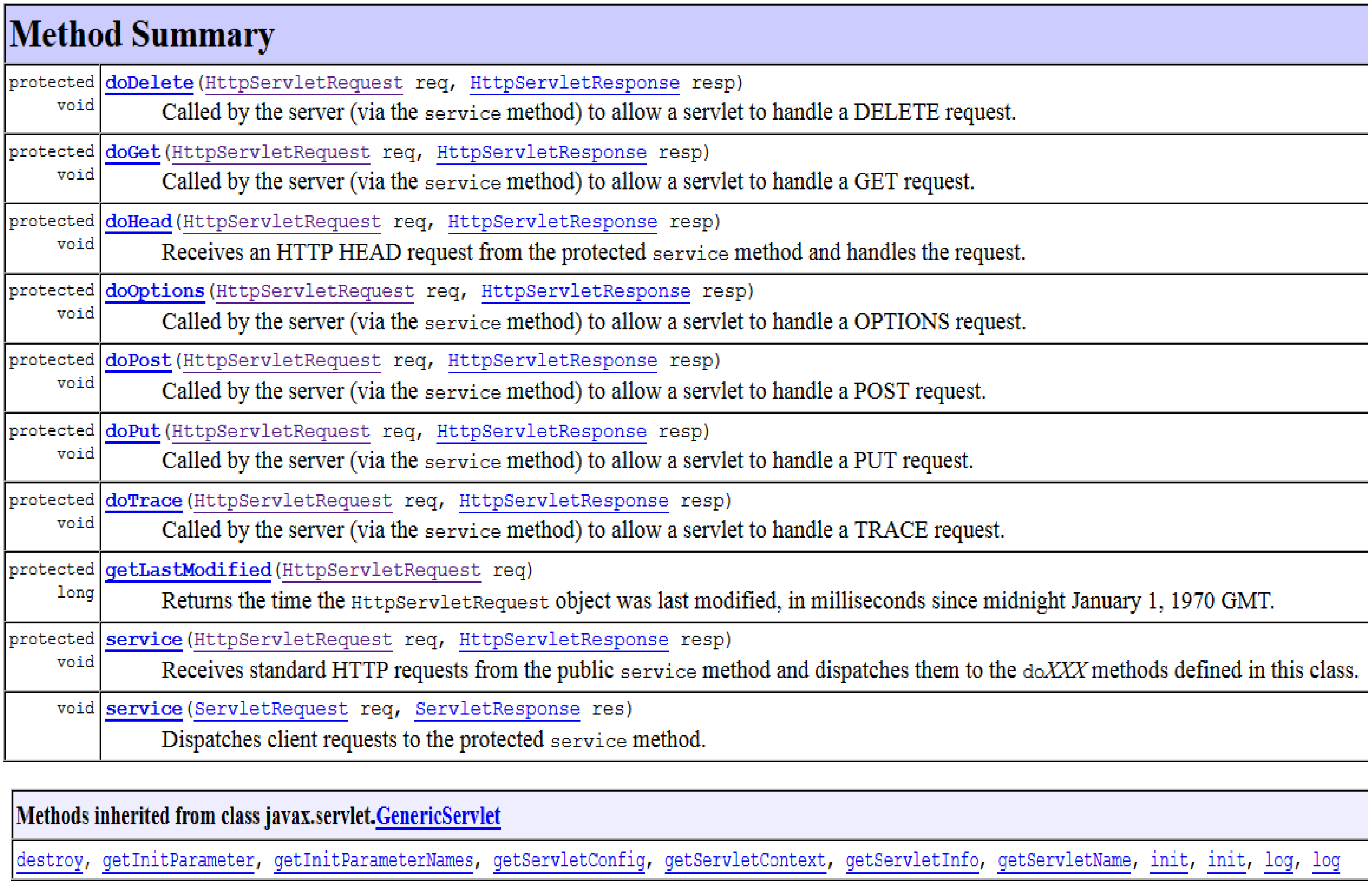
```java
// send response to browser
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
out.println(
  "<!doctype html public \"-//W3C//DTD HTML 4.0 Transitional//EN\">\n"
+ "<html>\n"
+ "<head>\n"
+ "  <title>Intro To Servlets</title>\n"
+ "</head>\n"
+ "<body>\n"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

Rewriting the address book app with a Servlet



```
        + "below.</p>\n"
        + "<form action=\"index.jsp\" >\n"
        + "   <input type=\"submit\" value=\"Return\">\n"
        + "</form>\n"
        + "</body>\n"
        + "</html>\n");

        out.close();
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        doPost(req, resp);

    }
}
```

Common Methods of the HttpServlet Class



Method Summary	
protected void	**doDelete**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the `service` method) to allow a servlet to handle a DELETE request.
protected void	**doGet**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the `service` method) to allow a servlet to handle a GET request.
protected void	**doHead**(HttpServletRequest req, HttpServletResponse resp) Receives an HTTP HEAD request from the protected `service` method and handles the request.
protected void	**doOptions**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the `service` method) to allow a servlet to handle a OPTIONS request.
protected void	**doPost**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the `service` method) to allow a servlet to handle a POST request.
protected void	**doPut**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the `service` method) to allow a servlet to handle a PUT request.
protected void	**doTrace**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the `service` method) to allow a servlet to handle a TRACE request.
protected long	**getLastModified**(HttpServletRequest req) Returns the time the `HttpServletRequest` object was last modified, in milliseconds since midnight January 1, 1970 GMT.
protected void	**service**(HttpServletRequest req, HttpServletResponse resp) Receives standard HTTP requests from the public `service` method and dispatches them to the `do`*XXX* methods defined in this class.
void	**service**(ServletRequest req, ServletResponse res) Dispatches client requests to the protected `service` method.

Methods inherited from class javax.servlet.GenericServlet
destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, init, log, log

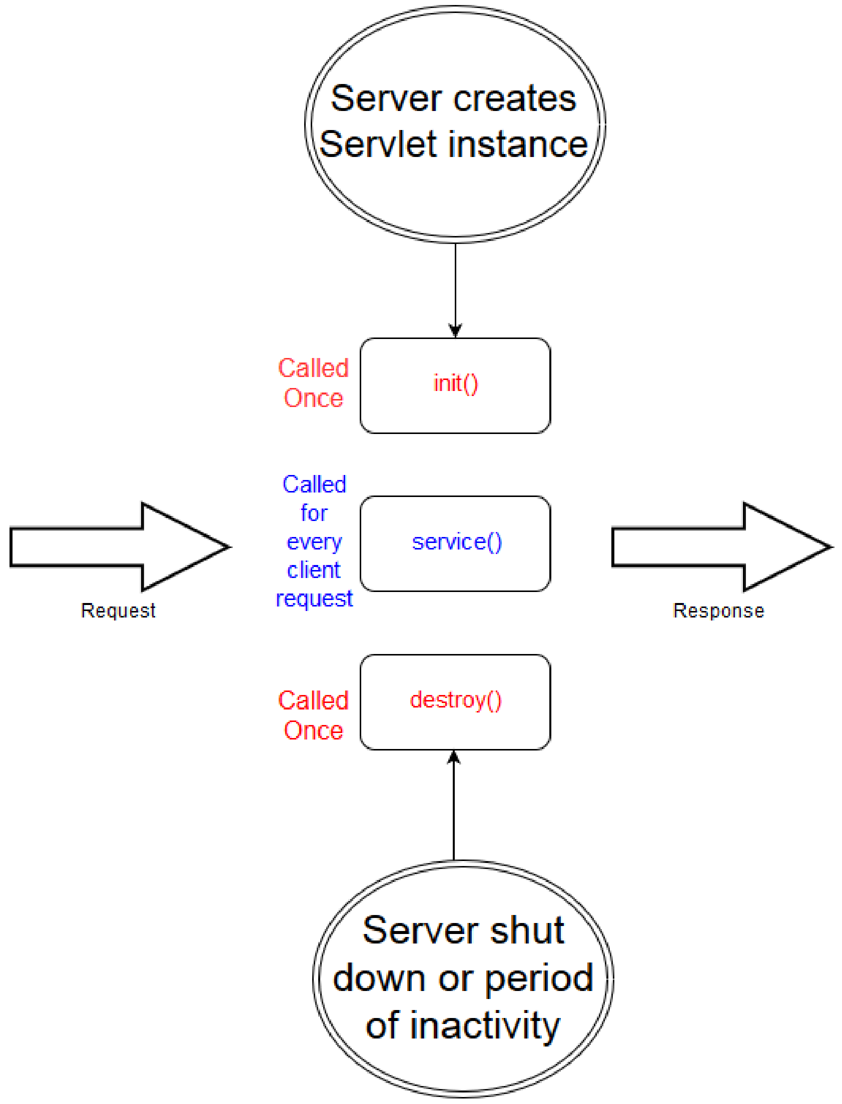
Common Methods of the HttpServlet Class

- The server only creates one instance of a Servlet.
  - Usually occurs when the server starts or when the Servlet is first requested.
  - Each request for the Servlet starts (or spawns) a thread that can access that one instance of the Servlet.
- When the server creates the instance of the Servlet, it calls the **init** method.
  - You can override it to supply any initialization code.
- After the server has created the one instance of the Servlet, each request for that Servlet spawns a thread that calls the service method of the Servlet.
  - This method checks the method that specified in the HTTP request and calls the appropriate **doGet** or **doPost** method.

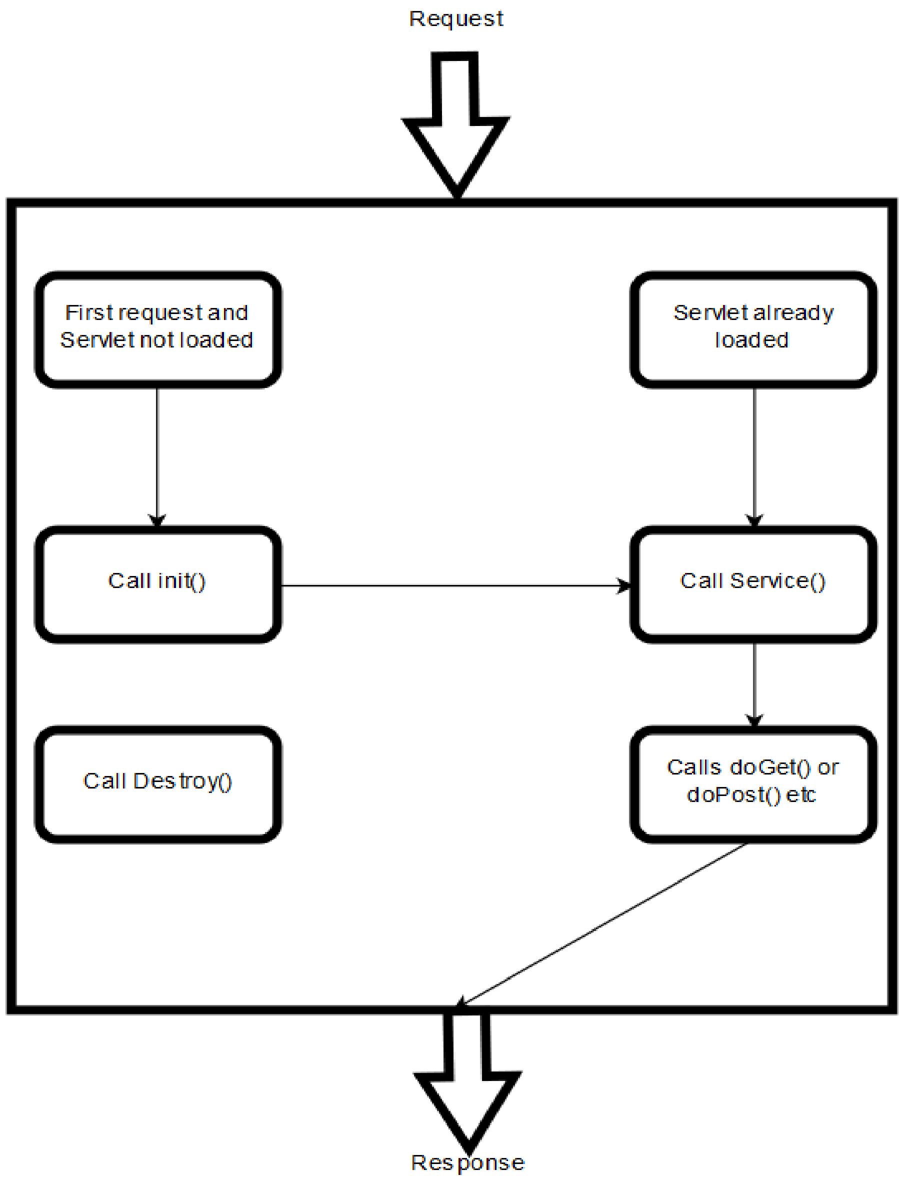
Common Methods of the HttpServlet Class

- When you code Servlets, you shouldn’t code the **service** method.
  - Instead you should override the appropriate **doGet** or **doPost** method.
- If a Servlet has been idle for some time or if the server has been shut down it (the server) will call the **destroy** method.
- If you want to provide some cleanup code like writing a value to a file or closing a DB connection you can place this code in the **destroy** method.
  - Note, the **destroy** method will not be called if the server crashes so it can’t be relied on to execute critical code.

Lifecycle of a Servlet


Server creates Servlet instance
Called Once
init()
Called for every client request
service()
Request
Response
Called Once
destroy()
Server shut down or period of inactivity

How the server handles a request for a Servlet



The lifecycle of a Servlet

- A server loads and initialises a Servlet by calling the **init** method.
- The Servlet handles each request by calling the service method. This method then calls another method (**doGet** or **doPost** for example) to handle the specific HTTP request type.
- The server removes the Servlet by calling the **destroy** method. This occurs when either the servlet has been idle for some time or the server has been shut down.

What’s next with Servlets

- You should now be able to develop simple but practical Servlets of your own that return HTML to the browser.
- However, you usually don’t use Servlets in this manner (to generate HTML).
- Instead you structure your web applications so that Servlets do the business processing that’s required and that JSP’s send the HTML code back to the browser.
  - In this way you combine what is good about both JSP’s and Servlets.

References

Murach, J., (2014) *Murachs Java Servlets JSP*, 3rd edn. Mike Murach and Associates, Inc.

Jendrock E, Cervera-Navarro R, Evans I, Hasse K, Markito W (2014) *The Java EE 7 Tutorial*, 5th edn. Addison-Wesley Professional.

http://docs.oracle.com/javaee/6/tutorial/doc/