LAB - 12

Java Programming Lab

Topics Covered

Servlet

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 12.1

Aim:

Write hello world program in servlet.

Theory:

Servlets are Java classes which service HTTP requests and implement the **javax.servlet.Servlet** interface. Web application developers typically write servlets that extend javax.servlet.http.HttpServlet, an abstract class that implements the Servlet interface and is specially designed to handle HTTP requests.

Assuming your environment is setup properly, go in **ServletDevel** directory and compile HelloWorld.java as follows −

$ javac HelloWorld.java

If the servlet depends on any other libraries, you have to include those JAR files on your CLASSPATH as well. I have included only servlet-api.jar JAR file because I'm not using any other library in Hello World program.

This command line uses the built-in javac compiler that comes with the Sun Microsystems Java Software Development Kit (JDK). For this command to work properly, you have to include the location of the Java SDK that you are using in the PATH environment variable.

If everything goes fine, above compilation would produce **HelloWorld.class** file in the same directory. Next section would explain how a compiled servlet would be deployed in production.

Servlet Deployment

By default, a servlet application is located at the path <Tomcat-installationdirectory>/webapps/ROOT and the class file would reside in <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes.

If you have a fully qualified class name of **com.myorg.MyServlet**, then this servlet class must be located in WEB-INF/classes/com/myorg/MyServlet.class.

For now, let us copy HelloWorld.class into <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes and create following entries in **web.xml** file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

Source Code:

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloWorld extends HttpServlet {

    private String message;

    public void init() throws ServletException {
        // Do required initialization
        message = "Hello World";
    }

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

        // Set response content type
        response.setContentType("text/html");

        // Actual logic goes here.
        PrintWriter out = response.getWriter();
        out.println("<h1>" + message + "</h1>");
    }

    public void destroy() {
        // do nothing.
    }
```

```
}
```

Output:


http://localhost:8080/HelloWorld - Windows Internet Explorer
http://localhost:8080/HelloWorld
File Edit View Favorites Tools Help
http://localhost:8080/HelloWorld
Hello World

EXPERIMENT – 12.2

Aim:

Write a servlet which displays current system date and time.

Theory:

The most important advantage of using Servlet is that we can use all the methods available in core java. The Date class is available in java.util package.

A **servlet** is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.

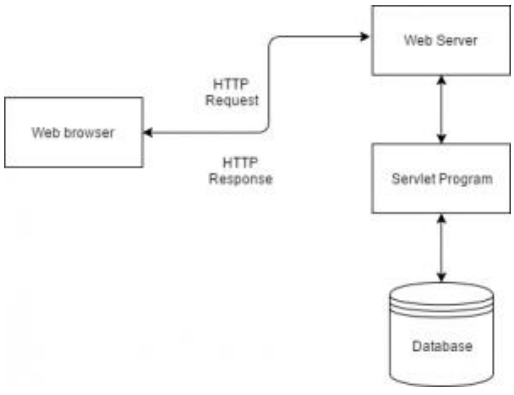
The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines life-cycle methods. When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.

This chapter focuses on writing servlets that generate responses to HTTP requests.

Properties of Servlets are as follows:

- Servlets work on the server-side.
- Servlets are capable of handling complex requests obtained from the webserver.

Servlet Architecture is can be depicted from the image itself as provided below as follows:



Execution of Servlets basically involves six basic steps:

1. The clients send the request to the webserver.
2. The web server receives the request.
3. The web server passes the request to the corresponding servlet.
4. The servlet processes the request and generates the response in the form of output.
5. The servlet sends the response back to the webserver.
6. The web server sends the response back to the client and the client browser displays it on the screen.

Source Code:

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

public class DateSrv extends GenericServlet
{
     //implement service()
     public void service(ServletRequest req, ServletResponse res) throws
IOException, ServletException
     {
          //set response content type
          res.setContentType("text/html");
          //get stream obj
          PrintWriter pw = res.getWriter();
          //write req processing logic
          java.util.Date date = new java.util.Date();
          pw.println("<h2>"+"Current Date & Time: "
+date.toString()+"</h2>");
          //close stream object
          pw.close();
     }
}
```

Output:


http://localhost:8080/HelloWorld - Windows Internet Explorer
http://localhost:8080/HelloWorld
File Edit View Favorites Tools Help
http://localhost:8080/HelloWorld
Current Date & Time: Mon Dec 12 18:01:39 IST 2016

EXPERIMENT – 12.3

Aim:

Write a program that handle HTTP request.

Theory:

The most important advantage of using Servlet is that we can use all the methods available in core java. The Date class is available in java.util package.

A **servlet** is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines life-cycle methods. When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.

The doGet () method is invoked by server through service () method to handle a HTTP GET request. This method also handles HTTP HEAD request automatically as HEAD request is nothing but a GET request having no body in the code for response and only includes request header fields. To understand the working of doGet () method, let us consider a sample program to define a servlet for handling the HTTP GET request.

The HTTP client sends the request to the server in the form of request message which includes following information:

- The Request-line
- The analysis of source IP address, proxy and port
- The analysis of destination IP address, protocol, port and host
- The Requested URI (Uniform Resource Identifier)
- The Request method and Content
- The User-Agent header
- The Connection control header

- The Cache control header

Source Code:

```
import java.net.*;
import java.io.*;

public class URLConnectionReader {
    public static void main(String[] args) throws Exception {
        URL yahoo = new URL("http://www.yahoo.com/");
        URLConnection yc = yahoo.openConnection();
        BufferedReader in = new BufferedReader(
                new InputStreamReader(
                        yc.getInputStream()));
        String inputLine;

        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();
    }
}
```

Output:

```
PS E:\sem 5\java> cd "e:\sem 5\java\" ; if ($?) { javac URLConnectionReader.java } ; if ($?) { java URLConne
ctionReader }
redirect
PS E:\sem 5\java>
```

EXPERIMENT – 12.4

Aim:

Write a program that handle HTTP response.

Theory:

Methods to Set HTTP Response Header

There are following methods which can be used to set HTTP response header in your servlet program. These methods are available with *HttpServletResponse* object.

Sr.No.	Method & Description
1	**String encodeRedirectURL(String url)** Encodes the specified URL for use in the sendRedirect method or, if encoding is not needed, returns the URL unchanged.
2	**String encodeURL(String url)** Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged.
3	**boolean containsHeader(String name)** Returns a Boolean indicating whether the named response header has already been set.
4	**boolean isCommitted()** Returns a Boolean indicating if the response has been committed.
5	**void addCookie(Cookie cookie)** Adds the specified cookie to the response.
6	**void addDateHeader(String name, long date)**

	Adds a response header with the given name and date-value.
7	**void addHeader(String name, String value)** Adds a response header with the given name and value.
8	**void addIntHeader(String name, int value)** Adds a response header with the given name and integer value.
9	**void flushBuffer()** Forces any content in the buffer to be written to the client.
10	**void reset()** Clears any data that exists in the buffer as well as the status code and headers.
11	**void resetBuffer()** Clears the content of the underlying buffer in the response without clearing headers or status code.
12	**void sendError(int sc)** Sends an error response to the client using the specified status code and clearing the buffer.
13	**void sendError(int sc, String msg)** Sends an error response to the client using the specified status.
14	**void sendRedirect(String location)** Sends a temporary redirect response to the client using the specified redirect location URL.
15	**void setBufferSize(int size)** Sets the preferred buffer size for the body of the response.

16	**void setCharacterEncoding(String charset)** Sets the character encoding (MIME charset) of the response being sent to the client, for example, to UTF-8.
17	**void setContentLength(int len)** Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.
18	**void setContentType(String type)** Sets the content type of the response being sent to the client, if the response has not been committed yet.
19	**void setDateHeader(String name, long date)** Sets a response header with the given name and date-value.
20	**void setHeader(String name, String value)** Sets a response header with the given name and value.
21	**void setIntHeader(String name, int value)** Sets a response header with the given name and integer value
22	**void setLocale(Locale loc)** Sets the locale of the response, if the response has not been committed yet.
23	**void setStatus(int sc)** Sets the status code for this response

we would use **setIntHeader()** method to set **Refresh** header.

Source Code:

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

// Extend HttpServlet class
public class refresh extends HttpServlet {

    // Method to handle GET method request.
    public void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        // Set refresh, autoload time as 5 seconds
        response.setIntHeader("refresh", 5);

        // Set response content type
        response.setContentType("text/html");

        // Get current time
        Calendar calendar = new GregorianCalendar();
        String am_pm;
        int hour = calendar.get(Calendar.HOUR);
        int minute = calendar.get(Calendar.MINUTE);
        int second = calendar.get(Calendar.SECOND);

        if (calendar.get(Calendar.AM_PM) == 0)
            am_pm = "AM";
        else
            am_pm = "PM";

        String CT = hour + ":" + minute + ":" + second + " " + am_pm;

        PrintWriter out = response.getWriter();
        String title = "Auto refresh Header Setting";
        String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
"transitional//en\">\n";

        out.println(docType +
                "<html>\n" +
```

```
                "<head><title>" + title + "</title></head>\n" +
                "<body bgcolor = \"#f0f0f0\">\n" +
                "<h1 align = \"center\">" + title + "</h1>\n" +
                "<p>Current Time is: " + CT + "</p>\n");
    }

    // Method to handle POST method request.
    public void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        doGet(request, response);
    }
}
```

Output:

EXPERIMENT – 12.5

Aim:

Create a servlet that uses Cookies to store the number of times a user has visited your servlet.

Theory:

Many times you would be interested in knowing total number of hits on a particular page of your website. It is very simple to count these hits using a servlet because the life cycle of a servlet is controlled by the container in which it runs.

Following are the steps to be taken to implement a simple page hit counter which is based on Servlet Life Cycle −

- Initialize a global variable in init() method.
- Increase global variable every time either doGet() or doPost() method is called.
- If required, you can use a database table to store the value of global variable in destroy() method. This value can be read inside init() method when servlet would be initialized next time. This step is optional.
- If you want to count only unique page hits with-in a session then you can use isNew() method to check if same page already have been hit with-in that session. This step is optional.
- You can display value of the global counter to show total number of hits on your web site. This step is also optional.

Here I'm assuming that the web container will not be restarted. If it is restarted or servlet destroyed, the hit counter will be reset.

```
<servlet>
   <servlet-name>PageHitCounter</servlet-name>
   <servlet-class>PageHitCounter</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>PageHitCounter</servlet-name>
   <url-pattern>/PageHitCounter</url-pattern>
</servlet-mapping>
....
```

Source Code:

```
import java.io.*;
import java.sql.Date;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class PageHitCounter extends HttpServlet {

   private int hitCount;

   public void init() {
      // Reset hit counter.
      hitCount = 0;
   }

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

      // Set response content type
      response.setContentType("text/html");

      // This method executes whenever the servlet is hit
      // increment hitCount
      hitCount++;
      PrintWriter out = response.getWriter();
      String title = "Total Number of Hits";
      String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
"transitional//en\">\n";

      out.println(docType +
         "<html>\n" +
            "<head><title>" + title + "</title></head>\n" +
            "<body bgcolor = \"#f0f0f0\">\n" +
               "<h1 align = \"center\">" + title + "</h1>\n" +
               "<h2 align = \"center\">" + hitCount + "</h2>\n" +
            "</body>
         </html>"
      );
   }

   public void destroy() {
      // This is optional step but if you like you
```

```
      // can write hitCount value in your database.
   }
}
```

Output:

Viva Questions

1. How many objects of a servlet is created?

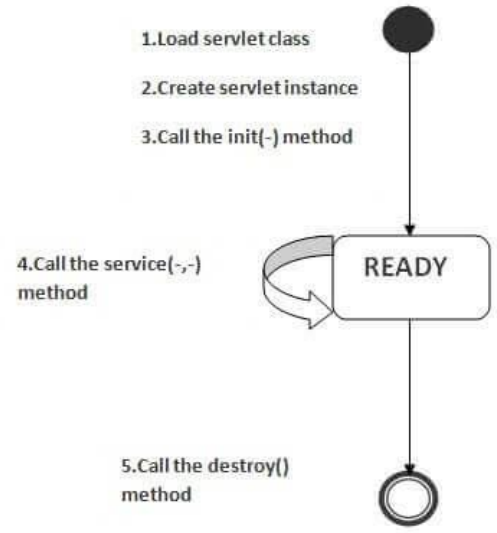
Ans.

Only one object at the time of first request by servlet or web container.

2. What is the life-cycle of a servlet?

Ans.

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



3. What are the life-cycle methods for a servlet?

Ans.

Method	Description
public void init(ServletConfig config)	It is invoked only once when first request comes for the servlet. It is used to initialize the servlet.
public void service(ServletRequest request,ServletResponse)throws ServletException,IOException	It is invoked at each request.The service() method is used to service the request.
public void destroy()	It is invoked only once when servlet is unloaded.

4. Who is responsible to create the object of servlet?

Ans.

The web container or servlet container.

5. When servlet object is created?

Ans.

At the time of first request.