# Servlet Assignment

**Name : Prashant Maurya**
**empID: 12417**
**KIIT Roll : 1828259**

1) What is a Servlet and a dynamic web page?
Sol
In Java, a **servlet** is a way to create those **dynamic web pages**. **Servlets** are nothing but the java programs. In Java, a **servlet** is a type of java class which runs on JVM(java virtual machine) on the server side. Java **servlets** works on server side.
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.
A **dynamic web page** is a **web page** that displays different content each time it's viewed. For example, the **page** may change with the time of day, the user that accesses the **webpage**, or the type of user interaction.

2) Explain the life cycle of a Servlet.
Sol

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

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.

## 1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

## 2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

## 3) init method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

```
public void init(ServletConfig config) throws ServletException
```

## 4) service method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

```
public void service(ServletRequest request, ServletResponse response)
  throws ServletException, IOException
```

## 5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

```
public void destroy()
```

3) Explain briefly the working of a Servlet Application

Sol

Each time the web server receives a request, the **servlet** container creates HttpServletRequest and HttpServletResponse objects. The HttpServletRequest object provides the access to the request information and the HttpServletResponse object allows us to format and change the http response before sending it to the client.The servlet container spawns a new thread that calls service() method for each client request. **The service() method dispatches the request to the correct handler method based on the type of request**.

4) What is a Deployment Descriptor? Give an example of a Deployment Descriptor with servlet mapping.

Sol

In java dynamic web project, we have Deployment Descriptor known as web.xml file.

Web.xml defines mappings between URL paths and the servlets that handle requests with those paths. The web server uses this configuration to identify the servlet to handle a given request and call the class method that corresponds to the request method. For example: the `doGet()` method for HTTP `GET` requests.

To map a URL to a servlet, you declare the servlet with the `<servlet>` element, then define a mapping from a URL path to a servlet declaration with the `<servlet-mapping>` element.

The `<servlet>` element declares the servlet, including a name used to refer to the servlet by other elements in the file, the class to use for the servlet, and initialization parameters. You can declare multiple servlets using the same class with different initialization parameters. The name for each servlet must be unique across the deployment descriptor

```
<servlet>
        <servlet-name>redteam</servlet-name>
        <servlet-class>mysite.server.TeamServlet</servlet-class>
        <init-param>
            <param-name>teamColor</param-name>
            <param-value>red</param-value>
        </init-param>
        <init-param>
            <param-name>bgColor</param-name>
            <param-value>#CC0000</param-value>
        </init-param>
    </servlet>

    <servlet>
        <servlet-name>blueteam</servlet-name>
        <servlet-class>mysite.server.TeamServlet</servlet-class>
        <init-param>
            <param-name>teamColor</param-name>
            <param-value>blue</param-value>
        </init-param>
        <init-param>
            <param-name>bgColor</param-name>
            <param-value>#0000CC</param-value>
        </init-param>
    </servlet>
```

```
<servlet-mapping>
        <servlet-name>redteam</servlet-name>
        <url-pattern>/red/*</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>blueteam</servlet-name>
        <url-pattern>/blue/*</url-pattern>
    </servlet-mapping>
```

With this example, a request for the URL `http://www.example.com/blue/teamProfile` is handled by the `TeamServlet` class, with the `teamColor` parameter equal to `blue` and the `bgColor` parameter equal to `#0000CC`. The servlet can get the portion of the URL path matched by the wildcard using the ServletRequest object's `getPathInfo()` method.

============================================================

5) Write the JSP, Servlet and Deployment Descriptor code for a Web application to add two numbers and show the result as a dynamic web page.

Sol

**Html file->**

```
<!DOCTYPE html>

<html>

<head>
```

```
<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

        <h3>Hello Friend</h3>

        <form action="add">

                <div>enter 1st number: <input type="text" name="num1"></div>

                <hr>

                <div>enter 2nd number: <input type="text" name="num2"></div>

                <hr>

                <input type="submit">

        </form>

</body>

</html>
```

## Servlet code->

```
package com.maurya.prashant;



import java.io.IOException;

import java.io.PrintWriter;
import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class AddServlet extends HttpServlet{

        /**

         *

         */

        private static final long serialVersionUID = 1L;

        public void service(HttpServletRequest req, HttpServletResponse res) throws IOException {



                int i1 = Integer.parseInt(req.getParameter("num1"));

                int i2 = Integer.parseInt(req.getParameter("num2"));



                int k = i1+i2;



                PrintWriter out = res.getWriter();



                out.println("sum is :"+k);
        }

}
```

**web.XML file->**

```
<servlet>


        <servlet-name>first</servlet-name>

        <servlet-class>com.maurya.prashant.AddServlet</servlet-class>


</servlet>


<servlet-mapping>


        <servlet-name>first</servlet-name>

        <url-pattern>/add</url-pattern>


</servlet-mapping>
```

============================================================

6) Write a Servlet application to print the current date and time.

Sol

## Servlet

```
package com.maurya.prashant;



import java.io.IOException;

import java.io.PrintWriter;



import javax.servlet.GenericServlet;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.HttpServlet;



//The main difference between GenericServlet and HttpServlet is that the GenericServlet

//is protocol independent and can be used with any protocol such as HTTP, SMTP, FTP, and,

//CGI while HttpServlet is protocol dependent and only used with HTTP protocol



public class CurrrentDateTime 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();

    }


}
```

## Web xml

```
<servlet>
```

```
        <servlet-name>time</servlet-name>

        <servlet-class>com.maurya.prashant.CurrentDateTime</servlet-class>


</servlet>


<servlet-mapping>


        <servlet-name>time</servlet-name>

        <url-pattern>/give-time</url-pattern>


</servlet-mapping>
```

## Html file

```
<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>


        <form action="give-time">
```

```
                    <h1>Click to get current time and date</h1>

                    <input type="submit">


          </form>


</body>

</html>
```

7) Create an html form which takes username and location and input and returns a customized message with those parameters using a servlet

Sol

Html ->

```
<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>
```

```
          <form action="inputUser">


                    <div> <h3>enter username: </h3> <input type="text" name="username"></div>

                    <div> <h3>enter location(City): </h3> <input type="text" name="location"></div>

                    <hr>


                    <input type="submit">




          </form>


</body>

</html>
```

## Web xml

```
<servlet>


          <servlet-name>user</servlet-name>

          <servlet-class>com.maurya.prashant.UserLocation</servlet-class>


</servlet>


<servlet-mapping>
```

```
        <servlet-name>user</servlet-name>

        <url-pattern>/inputUser</url-pattern>



</servlet-mapping>
```

## Servlet

```
package com.maurya.prashant;



import java.io.IOException;

import java.io.PrintWriter;



import javax.servlet.GenericServlet;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;





public class UserLocation extends GenericServlet{



        @Override

        public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {

                // TODO Auto-generated method stub
                String username = req.getParameter("username");

                String location = req.getParameter("location");



                res.setContentType("text/html");



                PrintWriter pw = res.getWriter();



                pw.println("<h2>"+username+" location is "+ location+"</h2>");



                pw.close();



        }



}
```