

Employee Id : 11949

Employee Name : Muthuraj Periyasamy

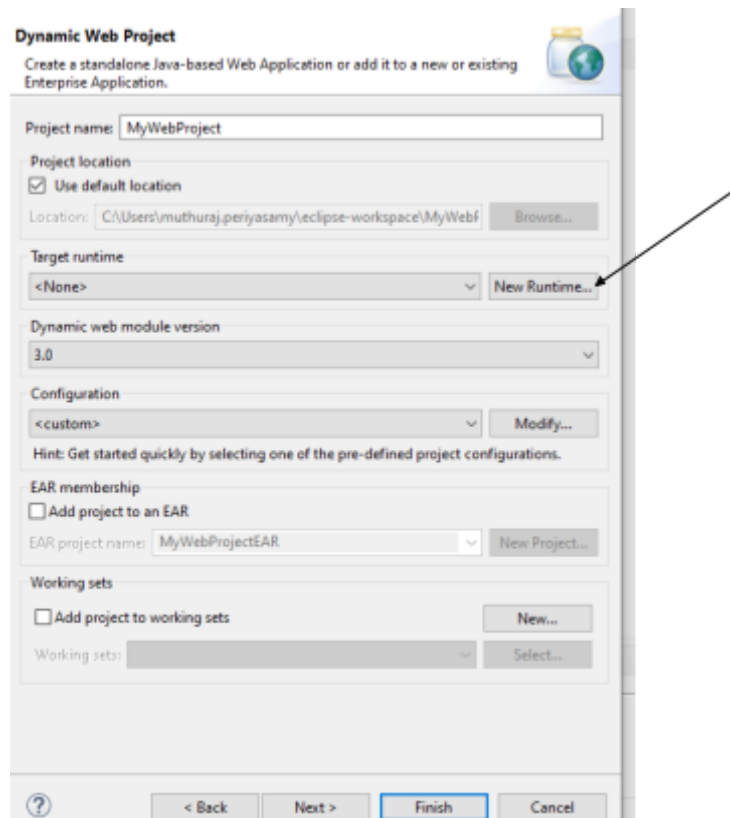
Three Important :

1. Where did write Java Code — Src/main/java
2. Where did write html.jsp —src -main -webapp-html and jsp
3. Deployment Describe—Now call address Server

1. Servlet:

New – Click Others Option - Click and -Web - DWP

1. First - RightClick -File - Open -Dynamic Web Project



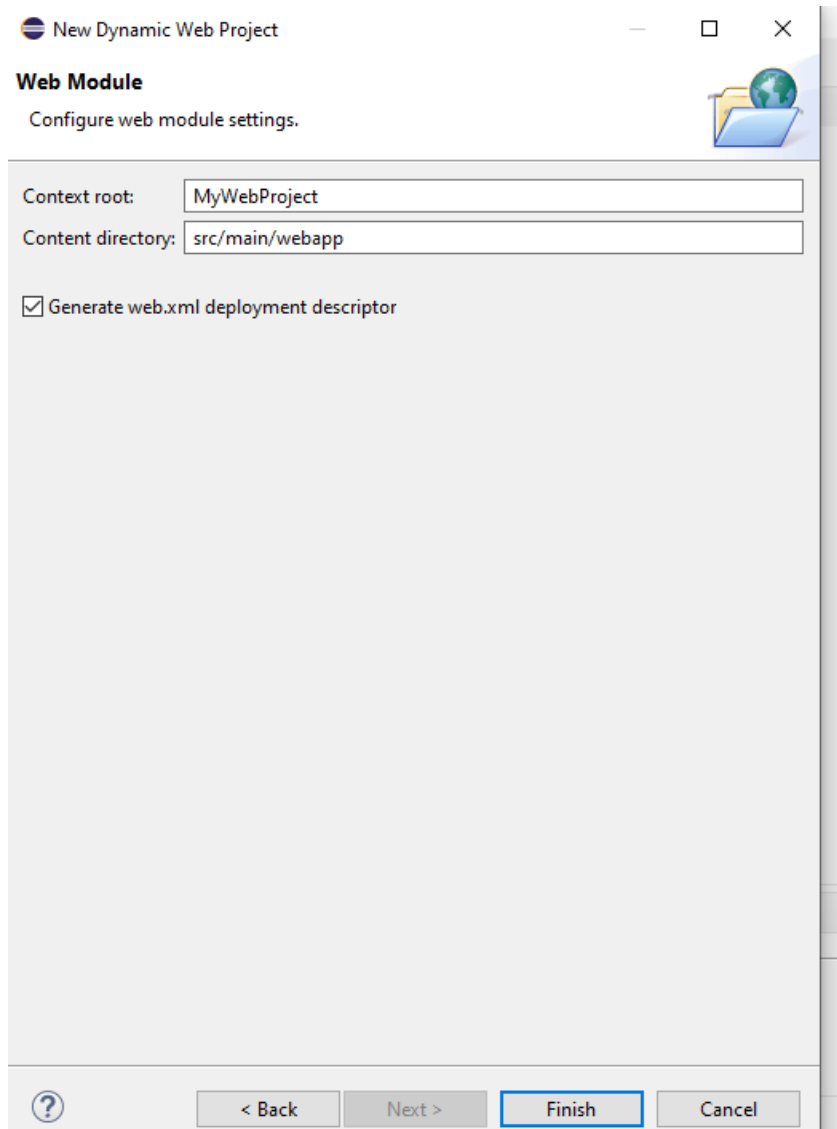
2. Next Option : NEW RUNTIME – Apache –Apache TomCat v9.0

☒ Create a new local Server - NEXT

Which Folder Extract TOMCAT FILE:

Click –Browser -Documents -Select the Tomcat File -Path –**FINISH**

3.



New Dynamic Web Project

Web Module
Configure web module settings.

Context root:

Content directory:

☒ Generate web.xml deployment descriptor

? < Back Next > Finish Cancel

Click two times Next

Create Servlet Class:

Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:

Which method stubs would you like to create?

☐ Constructors from superclass
☒ Inherited abstract methods

<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input checked="" type="checkbox"/> service	<input type="checkbox"/> doGet
<input type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

? < Back Next > Finish Cancel

Login And User:

localhost:8080/MyWebProject/LoginServlet?uname=Admin&pwd=123 QueryString:

DOGET:

Service is default doGet Calling:

DOPOST:

It can mention (**doPost**) LoginServlet.java.

Jsp: <form action="LoginServlet **method="post">**

</form>

Java Servlet:

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).

Servlet technology is robust and scalable because of the Java language. Before Servlet, CGI (**Common Gateway Interface**) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below.

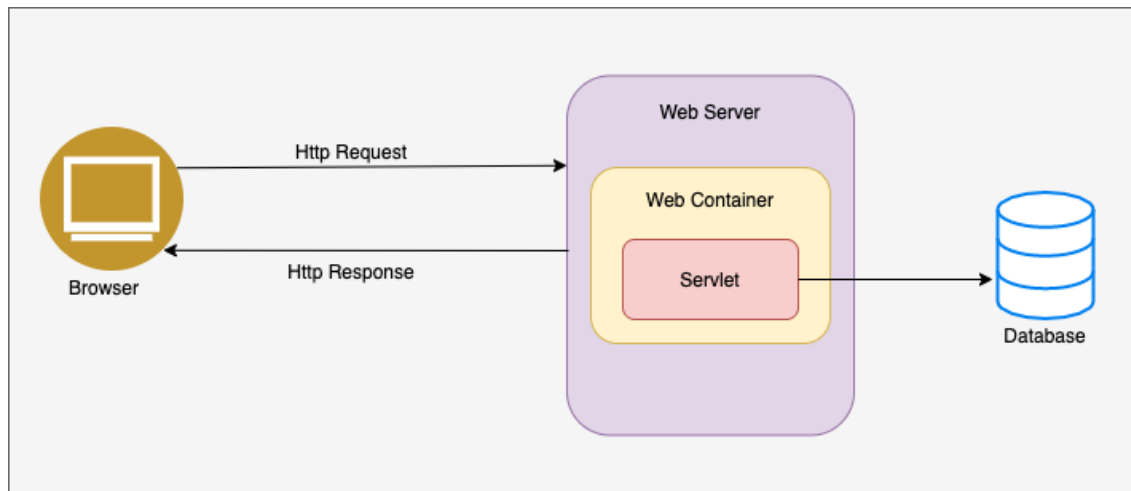
There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

What is a Servlet?

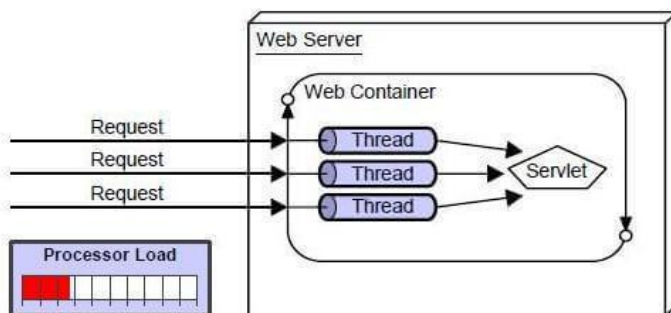
Servlet can be described in many ways, depending on the context.

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.

- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.



Advantages of Servlet:



There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, are lightweight, and cost of communication between the threads are low. The advantages of Servlet are as follows:

1. **Better performance:** Because it creates a thread for each request, not process.
2. **Portability:** Because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** Because it uses java language.

There are two types of servlets in Java:

generic servlets and HTTP servlets. Generic servlets are protocol-independent and can handle any type of request, while HTTP servlets are specific to the HTTP protocol and are used to handle HTTP requests and responses.

Servlet containers are usually a component of Web and application servers, such as **BEA WebLogic Application Server, IBM WebSphere, Sun Java System Web Server, Sun Java System Application Server**, and others. You might want to check out the latest information on JavaServer Pages (JSP) technology.

Servlet Terminology	Description
Website: static vs dynamic	It is a collection of related web pages that may contain text, images, audio and video.
HTTP	It is the data communication protocol used to establish communication between client and server.
HTTP Requests	It is the request send by the computer to a web server that contains all sorts of potentially interesting information.
Get vs Post	It gives the difference between GET and POST request.
Container	It is used in java for dynamically generating the web pages on the server side.
Server: Web Application	It is used to manage the network resources and for running the program or software that provides services.
Content Type	It is HTTP header that provides the description about what are you sending to the browser.

Website

Website is a collection of related web pages that may contain text, images, audio and video. The first page of a website is called the home page. Each website has a specific internet address (URL) that you need to enter in your browser to access a website.

Website is hosted on one or more servers and can be accessed by visiting its homepage using a computer network. A website is managed by its owner that can be an individual, company or an organization.

A website can be of two types:

- Static Website
- Dynamic Website

Static website

Static website is the basic type of website that is easy to create. You don't need the knowledge of web programming and database design to create a static website. Its web pages are coded in HTML.

The codes are fixed for each page so the information contained in the page does not change and it looks like a printed page.

Dynamic website

Dynamic website is a collection of dynamic web pages whose content changes dynamically. It accesses content from a database or Content Management System (CMS). Therefore, when you alter or update the content of the database, the content of the website is also altered or updated.

Dynamic websites use client-side scripting or server-side scripting, or both to generate dynamic content.

Client side scripting generates content at the client computer on the basis of user input. The web browser downloads the web page from the server and processes the code within the page to render information to the user.

In server side scripting, the software runs on the server and processing is completed in the server then plain pages are sent to the user.

CRUD in Servlet

A CRUD (Create, Read, Update and Delete) application is the most important a development. In Servlet, we can easily create CRUD application.

Servlet CRUD example

Create "user905" table in Oracle Database with auto incrementing id using sequence. The password, email and country.

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER	No	-	1
NAME	VARCHAR2(4000)	Yes	-	-
PASSWORD	VARCHAR2(4000)	Yes	-	-
EMAIL	VARCHAR2(4000)	Yes	-	-
COUNTRY	VARCHAR2(4000)	Yes	-	-
				1 - 5

Session Tracking in Servlets:

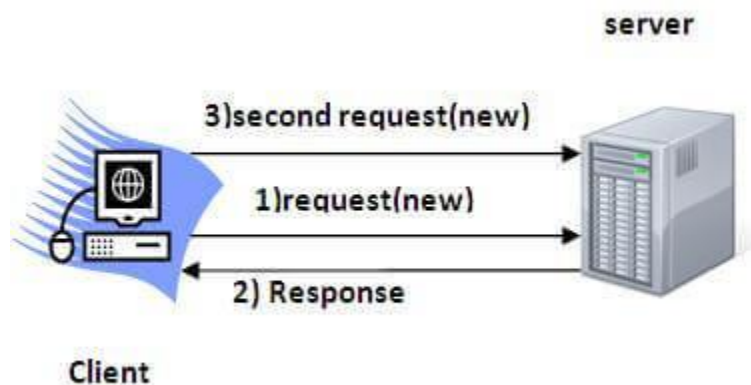
Session Tracking Session Tracking Techniques

Session simply means a particular interval of time.

Session Tracking is a way to maintain state (data) of a user. It is also known as **session management** in servlet.

Http protocol is stateless so we need to maintain state using session tracking techniques. Each time a user requests to the server, the server treats the request as the new request. So we need to maintain the state of a user to recognize a particular user.

HTTP is stateless, which means each request is considered as the new request. It is shown in the figure given below:



Why use Session Tracking?

To recognize the user It is used to recognize the particular user.

Session Tracking Techniques

There are four techniques used in Session tracking:

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

Cookies in Servlet

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookies with a response from the servlet. So the cookie is stored in the cache of the browser. After that if a request is sent by the user, a cookie is added with the request by default. Thus, we recognize the user as the old user.

and the optional attributes, so use them sparingly to improve the interoperability of your servlets.

File: index.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

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

```
<title>Insert tCookie:
```

Creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server. A cookie's value can uniquely identify a client, so cookies are commonly used for session management.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number. Some Web browsers have bugs in how they handle here</title>

```
</head>
```

```
<body>
```

<h1>Add New Employee</h1>

<form action="SaveServlet" method="post">

<table>

<tr><td>Name:</td><td><input type="text" name="name"/></td></tr>

<tr><td>Password:</td><td><input type="password" name="password"/></td></tr>

<tr><td>Email:</td><td><input type="email" name="email"/></td></tr>

<tr><td>Country:</td><td>

<select name="country" style="width:150px">

<option>India</option>

<option>USA</option>

<option>UK</option>

<option>Other</option>

</select>

</td></tr>

<tr><td colspan="2"><input type="submit" value="Save Employee"/></td></tr>

</table>

</form>


```
<a href="ViewServlet">view employees</a>
```

```
</body>
```

```
</html>
```

File: Emp.java

```
public class Emp {
```

```
private int id;
```

```
private String name,password,email,country;
```

```
public int getId() {
```

```
    return id;
```

```
}
```

```
public void setId(int id) {
```

```
    this.id = id;
```

```
}
```

```
public String getName() {
```

```
    return name;
```

```
}
```

```
public void setName(String name) {
```

```
    this.name = name;

}

public String getPassword() {

    return password;

}

public void setPassword(String password) {

    this.password = password;

}

public String getEmail() {

    return email;

}

public void setEmail(String email) {

    this.email = email;

}

public String getCountry() {

    return country;

}

public void setCountry(String country) {

    this.country = country;
```

```
}}
```

File: EmpDao.java

```
import java.util.*;
```

```
import java.sql.*;
```

```
public class EmpaDao {
```

```
    public static Connection getConnection(){
```

```
        Connection con=null;
```

```
        try{
```

```
            Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","orac  
le");
```

```
        }catch(Exception e){System.out.println(e);} 
```

```
        return con;
```

```
    }
```

```
    public static int save(Emp e){
```

```
        int status=0;
```

```
        try{
```

```
            Connection con=EmpDao.getConnection();
```



```

PreparedStatement ps=con.prepareStatement(

    "insert into user905(name,password,email,country) values (?,?,?,?)");

ps.setString(1,e.getName());

ps.setString(2,e.getPassword());

ps.setStringString (3,e.getEmail());

ps.setString(4,e.getCountry());


status=ps.executeUpdate();

con.close();

}catch(Exception ex){ex.printStackTrace();}


return status;

}

public static int update(Emp e){

    int status=0;

    try{

        Connection con=EmpDao.getConnection();

        PreparedStatement ps=con.prepareStatement(

            "update user905 set name=?,password=?,email=?,country=? where id=?");

```

```

        ps.setString(1,e.getName());

        ps.setString(2,e.getPassword());

        ps.setString(3,e.getEmail());

        ps.setString(4,e.getCountry());

        ps.setInt(5,e.getId());


        status=ps.executeUpdate();


        con.close();

    }catch(Exception ex){ex.printStackTrace();}


    return status;

}

public static int delete(int id){

    int status=0;

    try{

        Connection con=EmpDao.getConnection();

        PreparedStatement ps=con.prepareStatement("delete from user905 where id=?");

        ps.setInt(1,id);

```

```
status=ps.executeUpdate();
```

```
con.close();
```

```
}catch(Exception e){e.printStackTrace();}
```

```
return status;
```

```
}
```

```
public static Emp getEmployeeById(int id){
```

```
Emp e=new Emp();
```

```
try{
```

```
Connection con=EmpDao.getConnection();
```

```
PreparedStatement ps=con.prepareStatement("select * from user905 where  
id=?");
```

```
ps.setInt(1,id);
```

```
ResultSet rs=ps.executeQuery();
```

```
if(rs.next()){
```

```
    e.setld(rs.getInt(1));
```

```
    e.setName(rs.getString(2));
```

```
        e.setPassword(rs.getString(3));

        e.setEmail(rs.getString(4));

        e.setCountry(rs.getString(5));

    }

    con.close();

} catch (Exception ex) {ex.printStackTrace();}

return e;

}

public static List<Emp> getAllEmployees(){

    List<Emp> list=new ArrayList<Emp>();

    try{

        Connection con=EmpDao.getConnection();

        PreparedStatement ps=con.prepareStatement("select * from user905");

        ResultSet rs=ps.executeQuery();

        while(rs.next()){

            Emp e=new Emp();

            e.setld(rs.getInt(1));
```

```
        e.setName(rs.getString(2));

        e.setPassword(rs.getString(3));

        e.setEmail(rs.getString(4));

        e.setCountry(rs.getString(5));

        list.add(e);

    }

    con.close();

} catch (Exception e) {e.printStackTrace();}

return list;

}

}
```

File: SaveServlet.java

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
@WebServlet("/SaveServlet")
```

```
public class SaveServlet extends HttpServlet {
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {
```

```
        response.setContentType("text/html");
```

```
        PrintWriter out=response.getWriter();
```

```
        String name=request.getParameter("name");
```

```
        String password=request.getParameter("password");
```

```
        String email=request.getParameter("email");
```

```
        String country=request.getParameter("country");
```

```
        Emp e=new Emp();
```

```
        e.setName(name);
```

```
        e.setPassword(password);
```

```
        e.setEmail(email);
```

```
        e.setCountry(country);
```

```
int status=EmpDao.save(e);

if(status>0){

    out.print("<p>Record saved successfully!</p>");

    request.getRequestDispatcher("index.html").include(request, response);

}else{

    out.println("Sorry! unable to save record");

}

out.close();

} }
```

File: EditServlet.java

```
import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;
```

```
@WebServlet("/EditServlet")
```

```
public class EditServlet extends HttpServlet {
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
    {
        throws ServletException, IOException {
```

```
        response.setContentType("text/html");
```

```
        PrintWriter out=response.getWriter();
```

```
        out.println("<h1>Update Employee</h1>");
```

```
        String sid=request.getParameter("id");
```

```
        int id=Integer.parseInt(sid);
```

```
        Emp e=EmpDao.getEmployeeById(id);
```

```
        out.print("<form action='EditServlet2' method='post'>");
```

```
        out.print("<table>");
```

```
        out.print("<tr><td></td><td><input type='hidden' name='id'
value='"+e.getId()+"></td></tr>");
```

```
        out.print("<tr><td>Name:</td><td><input type='text' name='name'
value='"+e.getName()+"></td></tr>");
```

```
        out.print("<tr><td>Password:</td><td><input type='password' name='password'
value='"+e.getPassword()+"></td></tr>");
```



```
        </td></tr>");

    out.print("<tr><td>Email:</td><td><input type='email' name='email'
value='"+e.getEmail()+"'/></td></tr>");

    out.print("<tr><td>Country:</td><td>");

    out.print("<select name='country' style='width:150px'>");

    out.print("<option>India</option>");

    out.print("<option>USA</option>");

    out.print("<option>UK</option>");

    out.print("<option>Other</option>");

    out.print("</select>");

    out.print("</td></tr>");

    out.print("<tr><td colspan='2'><input type='submit' value='Edit & Save '/></td></tr>");

    out.print("</table>");

    out.print("</form>");

    out.close();

}

}
```

ServletConfig and ServletContext:

ServletConfig and ServletContext, both are objects created at the time of [servlet](#) initialization and used to provide some initial parameters or configuration information to the servlet. But, the difference lies in the fact that information shared by ServletConfig is for a specific servlet, while information shared by ServletContext is available for all servlets in the web application. ServletConfig:

- ServletConfig is an object containing some initial parameters or configuration information created by Servlet Container and passed to the servlet during initialization.
- ServletConfig is for a particular servlet, that means one should store servlet specific information in web.xml and retrieve them using this object.
- Example: Suppose, one is building a job portal and desires to share different email ids (which may get change over time) to recruiter and job applicant. So, he decides to write two servlets one for handling recruiter's request and another one for the job applicant. Where to store email-ids? Put email-id as a name-value pair for different servlet inside web.xml which can further be retrieved using `getServletConfig().getInitParameter("name")` in the servlet.

ServletContext:

- ServletContext is the object created by Servlet Container to share initial parameters or configuration information to the whole application.
- Example: Suppose, the name of one's job portal is "NewWebsite.tg". Showing the website name at the top of webpages delivered by different servlets, one needs to store the website name in every servlet inviting redundancy. Since the information shared by ServletContext can be accessed by every Servlet, it is better to go with ServletContext and retrieve the website name using `getServletContext().getInitParameter("Name")` whenever required.

Advantage of ServletConfig

The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.

Methods of ServletConfig interface

1. **public String getInitParameter(String name):Returns the parameter value for the specified parameter name.**
2. **public Enumeration getInitParameterNames():Returns an enumeration of all the initialization parameter names.**
3. **public String getServletName():Returns the name of the servlet.**
4. **public ServletContext getServletContext():Returns an object of ServletContext.**

