1 Servlets – Part 1

In this chapter, we introduce the student to the concept of servlets. A servlet is a web component that is used for control purposes. It servlet resides in the web container and services HTTP requests coming from the client.

1.1 What is a servlet?

A servlet is a web component used to service HTTP requests by the server. A client sends an HTTP request to the server, the server gets an appropriate servlet to service the request, the servlet services the request, and thereafter returns an HTTP response to the client. The figure below shows the exchange of messages between a client and the server.

The server/container is responsible for the lifecycle of the servlet. This entails the following:

- The creation of a servlet.
- The loading of the servlet into memory. This is either done when the servlet is first requested or when the server reads the contents of the deployment descriptor (DD) file.
- The allocation of a servlet to a request in accordance to the client's request.
- The servicing of a request when the servlet is fully initialized.
- The end of life of a servlet.

The servlet technology is fully explained in a Java Specification Request (JSR) document called **JSR315**. The document can be found in the following website:

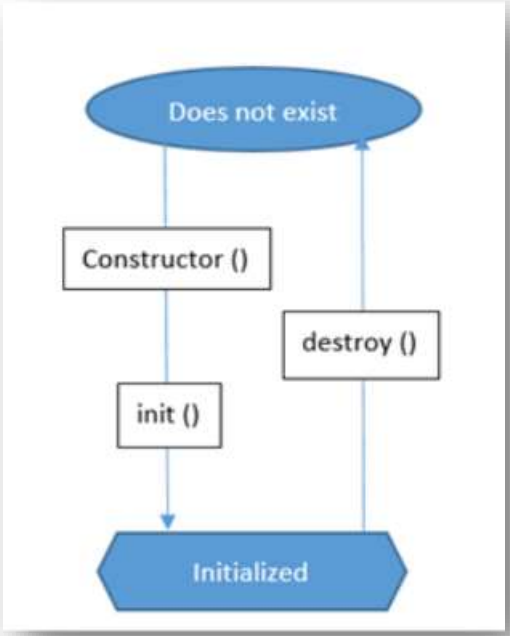
http://jcp.org/en/jsr/detail?id=315

1.2 Servlet life cycle

The servlet has two major life cycle stages, namely:

- The does not exist stage; and
- The initialized stage.

The figure below shows the two stages in the life cycle of a servlet.



In the **"Does not exist"** stage, the servlet is non-existent. When a servlet is either called by a client request or read from the deployment descriptor file, this triggers its creation and initialization. The container calls the no parameters constructor of the servlet to create the servlet. The init() method is then called to initialize the servlet.

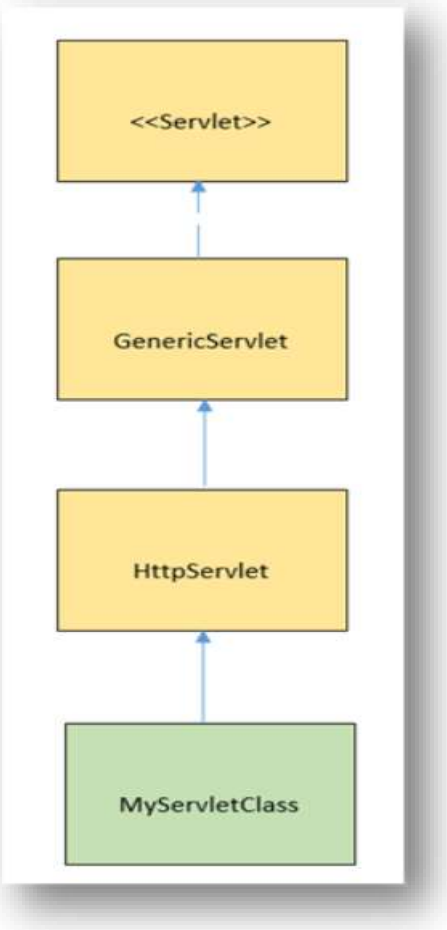
The servlet then enters its state of servicing requests, which is the **initialized** state. This is where the servlet spends most of its time. When the container decides to end the life of the servlet, it either destroys it by calling the **destroy()** method, or just takes the servlet to garbage collection.

1.3 How does a servlet work?

The web container receives an HTTP request message from a client (browser). The container sees that the request is for a specific servlet through the provided URL. The container calls the servlet and allocates it a thread to run. The container then creates and initializes two objects, **HttpServletResponse** and **HttpServletRequest**. It calls the **service** method of the servlet and passes it the two objects as arguments. The service method calls the appropriate HTTP method (**doGet**, **doPost**) to service the request.

1.4 Servlet API

According to the JEE documentation, any class that extends the **HttpServlet** is a **Servlet**. The inheritance hierarchy of the **Servlet** is as follows:



The class you create, **MyServletClass**, becomes a servlet on the basis of inheriting the **HttpServlet** class which is found in the **javax.servlet.http** package.

The HttpServlet class has the following methods that your servlet class needs to override, depending on the HTTP request of the client:

- doGet()
- doPost()
- doDelete()
- doPut() etc.

Mostly, we work with the doGet() and doPost() method. If a client sends an HTTP get request, the servlet must override the doGet() method. Alternatively, if the request is a post, the servlet must override the doPost() method. Below is the method summary of the **HttpServlet** class.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
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 doXXX methods defined in this class.
void	service(ServletRequest req, ServletResponse res) Dispatches client requests to the protected service method.

All of the above methods are called or invoked by the server when a specific method is requested by a client. The server acts as our **"main"** method.

The **HttpServlet** class inherits the **GenericServlet** class. **GenericServer** class defines methods that are independent of any communication protocol. This means this class, contrary to **HttpServlet**, which works with http, supports any communication protocol such as **ftp**. Because of inheritance, the HttpServlet class inherits all the

methods defined in GenericServlet. Below is the summary of some of the **GenericServlet** class methods.

All Methods | Instance Methods | Abstract Methods | Concrete Methods

Modifier and Type	Method and Description
void	destroy() Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
String	getInitParameter(String name) Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.
Enumeration<String>	getInitParameterNames() Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.
ServletConfig	getServletConfig() Returns this servlet's ServletConfig object.
ServletContext	getServletContext() Returns a reference to the ServletContext in which this servlet is running.
String	getServletInfo() Returns information about the servlet, such as author, version, and copyright.
String	getServletName() Returns the name of this servlet instance.
void	init() A convenience method which can be overridden so that there's no need to call super.init(config).
void	init(ServletConfig config) Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
void	log(String msg) Writes the specified message to a servlet log file, prepended by the servlet's name.
void	log(String message, Throwable t)

The **GenericServlet** class implements the **Servlet** interface. Below are the methods defined by the Servlet interface:

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
void	destroy() Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
ServletConfig	getServletConfig() Returns a ServletConfig object, which contains initialization and startup parameters for this servlet.
String	getServletInfo() Returns information about the servlet, such as author, version, and copyright.
void	init(ServletConfig config) Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
void	service(ServletRequest req, ServletResponse res) Called by the servlet container to allow the servlet to respond to a request.

1.5 ServletConfig and ServletContext

The **ServletConfig** interface is used to initialise parameters that we want to be available to a specific Servlet and JSP after the container has read the DD (Deployment Descriptor → web.xml) file. Through the **ServletContext** we are able to initialise parameters that will be available to all the files in the web application after loading of the DD file.

1.6 Example 1

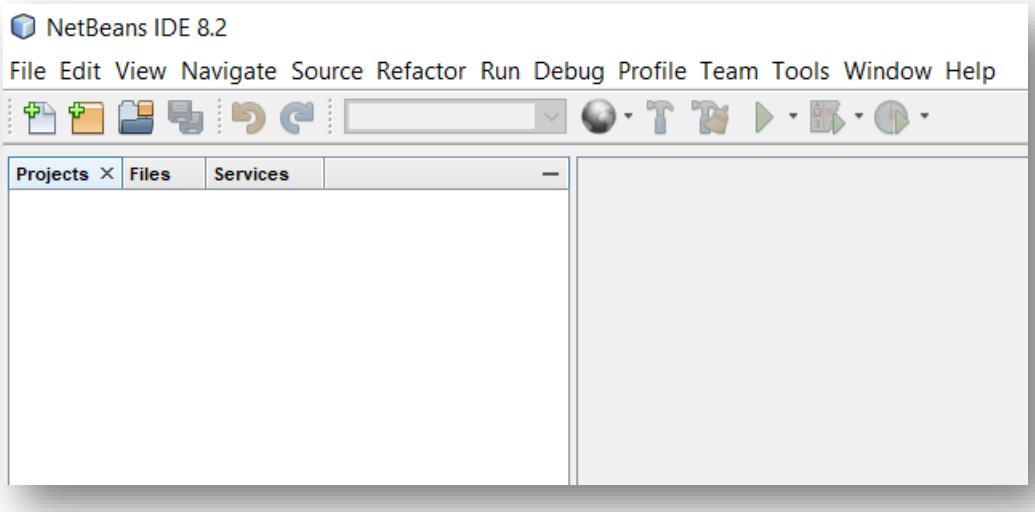
In this example we test the **ServletConfig** and **ServletContext** interfaces. To do that, we create a web application that provides subject information to a student. Upon running the application, the following information will be displayed about INT316D:

- Name and email address of the Subject Head;
- Name and code of the subject;
- Name and email address of the eMalahleni lecturer;
- Name and email address of the Polokwane lecturer; and
- Name and email address of the Soshanguve lecturer.

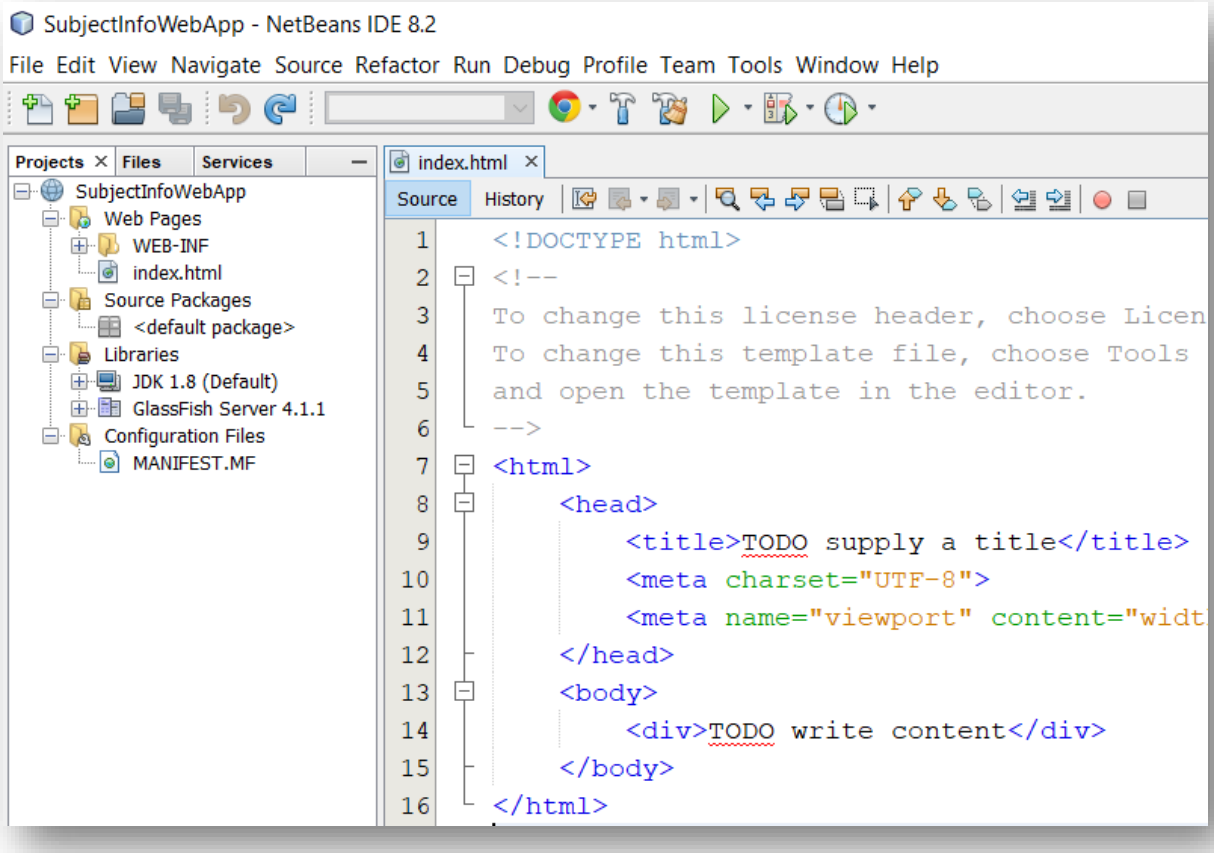
To demonstrate the difference between **ServletConfig** (**local variables**) and **ServletContext (global variables)**, the **details of the subject and that of the Subject Head** will be accessed via the **ServletContext,** and the rest through the **ServletConfig**.

Solution

Launch NetBeans.



Create a Web project called SubjectInfoWebApp



Modify the index page

```html
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>Welcome Page</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Welcome</h1>
        <p>
            Welcome to our subject information web app. Click <a href="menu.html">here</a> to start.
        </p>
    </body>
</html>
```

Create the menu page.

```html
<html>
    <head>
        <title>Menu Page</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Menu</h1>
        <p>
            Please select one of the following options:
        </p>
        <ol>
            <li><a href="SubjectDetailsServlet.do">Click here to display the subject details</a></li>
            <li><a href="SoshLecturersServlet.do">Click here to display Sosh lecturers details</a></li>
            <li><a href="eMalahleniLecturerServlet.do">Click here to display eMalahleni lecturer details</a></li>
            <li><a href="PolokwaneLecturerServlet.do">Click here to display Polokwane lecturer details</a></li>
        </ol>
    </body>
</html>
```

Create the **SubjectDetailsServlet** and keep in the **za.ac.tut.web** package

```java
/**
 *
 * @author MemaniV
 */
public class SubjectDetailsServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String subjectName = (String)getServletContext().getAttribute("subject_name");
        String subjectCode = (String)getServletContext().getAttribute("subject_code");
        String subjectHeadName = (String)getServletContext().getAttribute("subject_head_name");
        String subjectHeadEmail = (String)getServletContext().getAttribute("subject_head_email");
        Integer numClassTests = (Integer)getServletContext().getAttribute("num_class_tests");
        Integer numSemesterTests = (Integer)getServletContext().getAttribute("num_semester_tests");
        Integer numQuizzes = (Integer)getServletContext().getAttribute("num_quizzes");
        Integer numProjects = (Integer)getServletContext().getAttribute("num_projects");

        System.out.println("Subject Details" + "\n" + "---------------");
        System.out.println("Subject name: " + subjectName);
        System.out.println("Subject code: " + subjectCode);
        System.out.println("Subject head name: " + subjectHeadName);
        System.out.println("Subject head email: " + subjectHeadEmail);
        System.out.println("Number of Class Tests to be written: " + numClassTests);
        System.out.println("Number of Semester Tests to be written: " + numSemesterTests);
        System.out.println("Number of Quizzes to be written: " + numQuizzes);
        System.out.println("Number of Projects to be done: " + numProjects);
    }
}
```

Put the invoked attributes in the context of the DD file. Open the DD file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="
    <servlet>
        <servlet-name>SubjectDetailsServlet</servlet-name>
        <servlet-class>za.ac.tut.web.SubjectDetailsServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SubjectDetailsServlet</servlet-name>
        <url-pattern>/SubjectDetailsServlet.do</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

Include the context parameters and save.

web.xml

Source | General | Servlets | Filters | Pages | References | Security | History

```
11      <session-config>
12          <session-timeout>
13              30
14          </session-timeout>
15      </session-config>
16      <!-- context parameters -->
17      <context-param>
18          <param-name>subject_name</param-name>
19          <param-value>Internet Programming</param-value>
20      </context-param>
21      <context-param>
22          <param-name>subject_code</param-name>
23          <param-value>INT316D</param-value>
24      </context-param>
25      <context-param>
26          <param-name>subject_head_name</param-name>
27          <param-value>Vuyisile Memani</param-value>
28      </context-param>
29      <context-param>
30          <param-name>subject_head_email</param-name>
31          <param-value>MemaniV@tut.ac.za</param-value>
32      </context-param>
```

```
33      <context-param>
34          <param-name>num_class_tests</param-name>
35          <param-value>6</param-value>
36      </context-param>
37      <context-param>
38          <param-name>num_semester_tests</param-name>
39          <param-value>2</param-value>
40      </context-param>
41      <context-param>
42          <param-name>num_quizzes</param-name>
43          <param-value>12</param-value>
44      </context-param>
45      <context-param>
46          <param-name>num_projects</param-name>
47          <param-value>1</param-value>
48      </context-param>
49  </web-app>
```

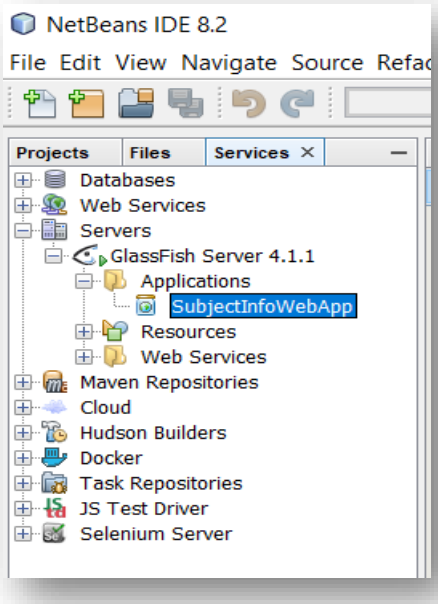
Clean and Build the project

```
Output - SubjectInfoWebApp (clean,dist)
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\SubjectInfoWebApp\build\web\META-INF
Copying 3 files to C:\Users\memaniv\Documents\NetBeansProjects\SubjectInfoWebApp\build\web
library-inclusion-in-archive:
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\SubjectInfoWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\SubjectInfoWebApp\build\generated-sources\ap-source-output
Compiling 1 source file to C:\Users\memaniv\Documents\NetBeansProjects\SubjectInfoWebApp\build\web\WEB-INF\classes
compile:
compile-jsps:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\SubjectInfoWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\SubjectInfoWebApp\dist\SubjectInfoWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 1 second)
```

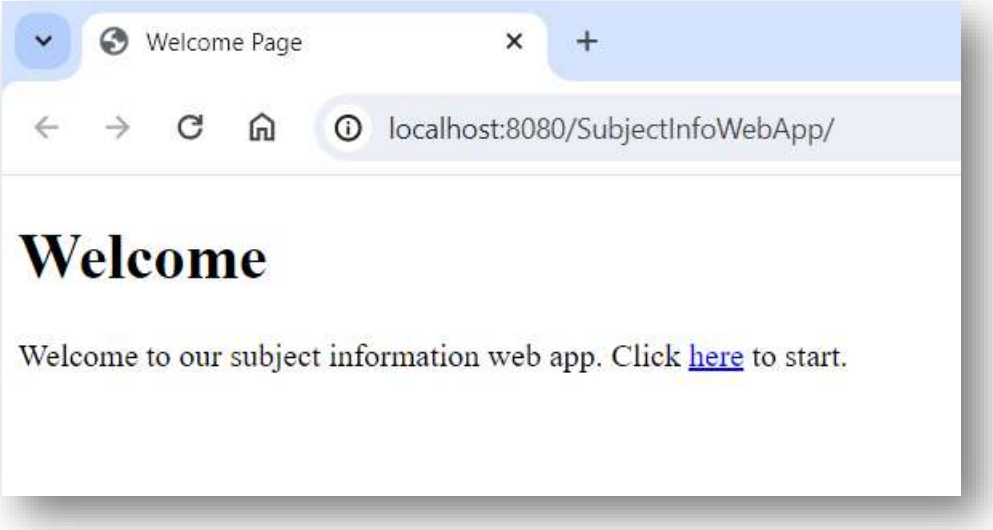
Start the server

```
Output
SubjectInfoWebApp (clean,dist) | Java DB Database Process | GlassFish Server 4.1.1
Info:   Created virtual server __asadmin
Info:   Setting JAAS app name glassfish-web
Info:   Virtual server server loaded default web module
Info:   Java security manager is disabled.
Info:   Entering Security Startup Service.
Info:   Loading policy provider com.sun.enterprise.security.provider.PolicyWrapper.
Info:   Security Service(s) started successfully.
Info:   visiting unvisited references
Info:   visiting unvisited references
Info:   visiting unvisited references
Info:   Initializing Mojarra 2.2.12 ( 20150720-0848 https://svn.java.net/svn/mojarra~svn/tags/2.2.12@14885) for context ''
Info:   Loading application [__admingui] at [/]
Info:   Loading application __admingui done in 6,149 ms
```

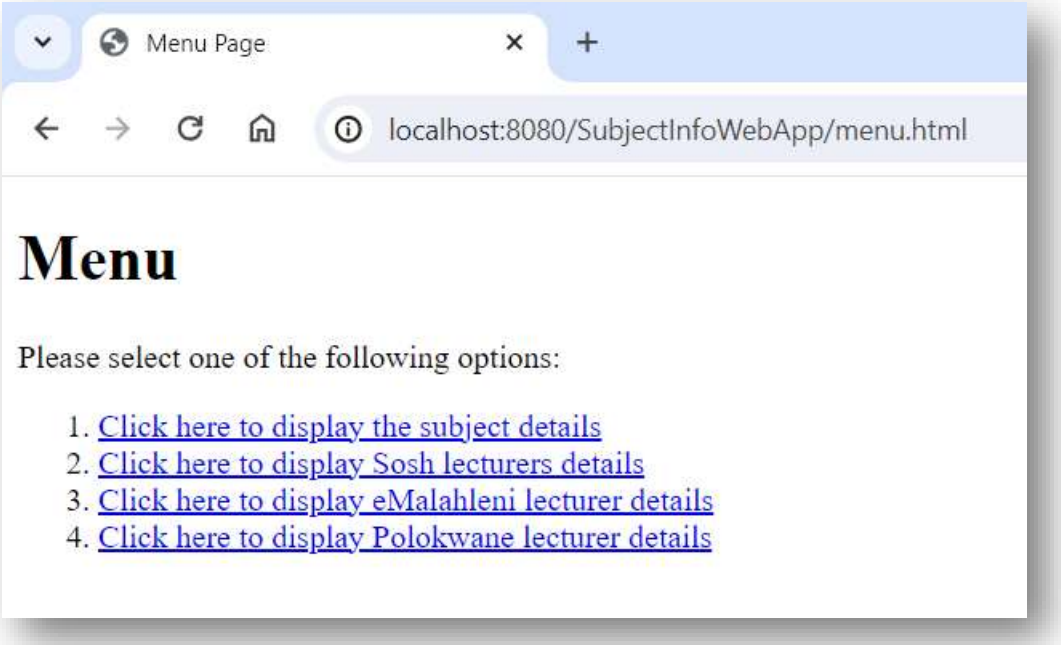
Deploy the application (WAR file)



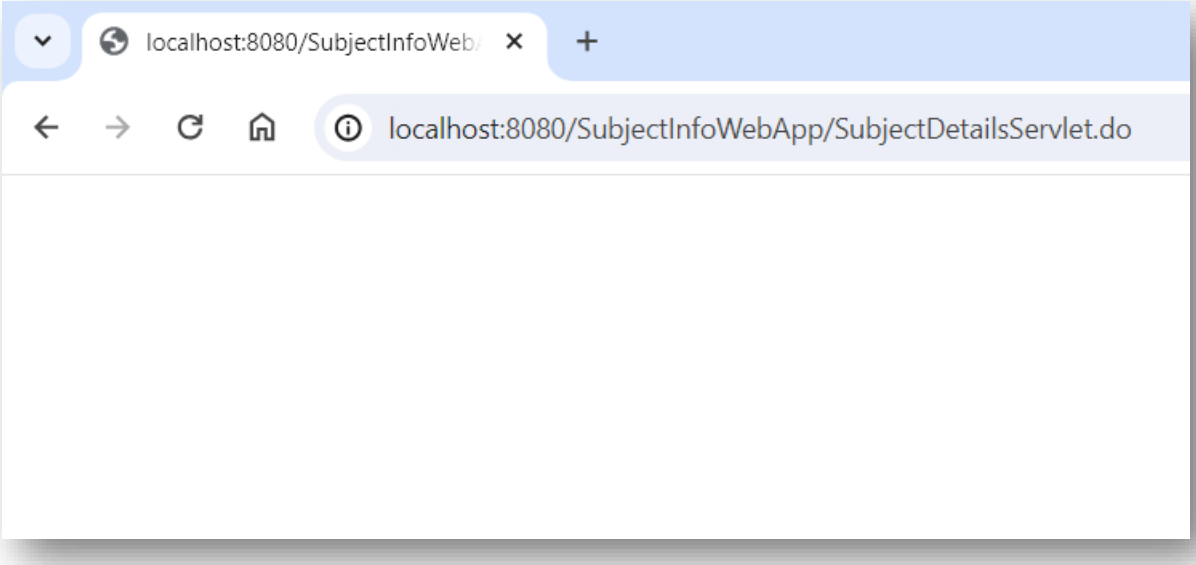
Run the application



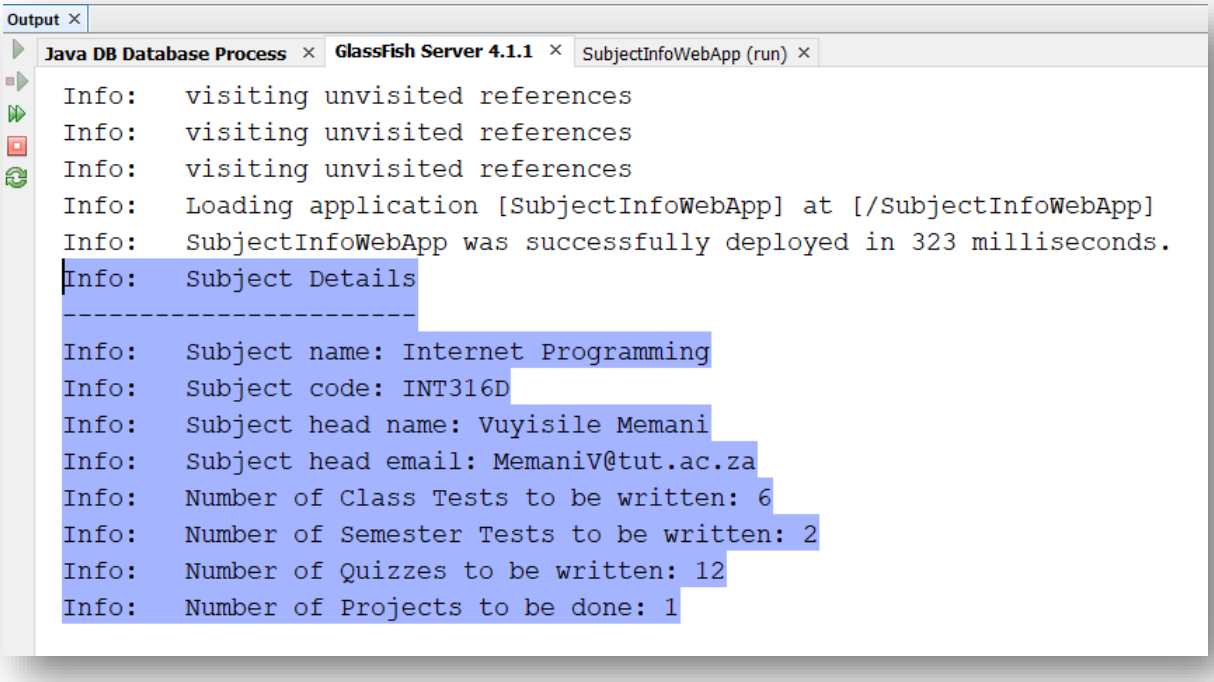
Click on the link



Click on the first link



Look on the output of GlassFish



Let’s create a JSP page called **subject_details.jsp** to access the values from there.

```jsp
<%--
    Document   : subject_details
    Created on : 19 Feb 2024, 3:47:09 AM
    Author     : Mamani7
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Subject Details Page</title>
    </head>
    <body>
        <h1>Subject details!</h1>
        <%
            String subjectName = pageContext.getServletContext().getInitParameter("subject_name");
            String subjectCode = pageContext.getServletContext().getInitParameter("subject_code");
            String subjectHeadName = pageContext.getServletContext().getInitParameter("subject_head_name");
            String subjectHeadEmail = pageContext.getServletContext().getInitParameter("subject_head_email");
            Integer numClassTests = Integer.parseInt(pageContext.getServletContext().getInitParameter("num_class_tests"));
            Integer numSemesterTests = Integer.parseInt(pageContext.getServletContext().getInitParameter("num_semester_tests"));
            Integer numQuizzes = Integer.parseInt(pageContext.getServletContext().getInitParameter("num_quizzes"));
            Integer numProjects = Integer.parseInt(pageContext.getServletContext().getInitParameter("num_projects"));
        %>
```

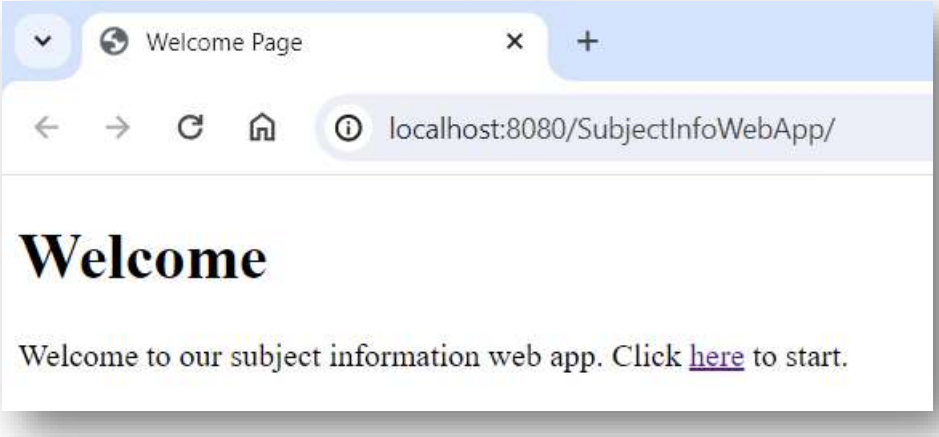
```jsp
        <p>
            Below are the subject details:
        </p>
        <table border="1">
            <tr>
                <td>Subject name:</td>
                <td><%=subjectName%></td>
            </tr>
            <tr>
                <td>Subject code:</td>
                <td><%=subjectCode%></td>
            </tr>
            <tr>
                <td>Subject head name:</td>
                <td><%=subjectHeadName%></td>
            </tr>
            <tr>
                <td>Subject head email:</td>
                <td><%=subjectHeadEmail%></td>
            </tr>
            <tr>
                <td>Number of Class Tests:</td>
                <td><%=numClassTests%></td>
            </tr>
```

```
            <tr>
                <td>Number of Semester Tests:</td>
                <td><%=numSemesterTests%></td>
            </tr>
            <tr>
                <td>Number of Quizzes:</td>
                <td><%=numQuizzes%></td>
            </tr>
            <tr>
                <td>Number of Projects:</td>
                <td><%=numProjects%></td>
            </tr>
        </table>
        <p>
            Click <a href="index.html">here</a> to go back to the main page.
        </p>
    </body>
</html>
```

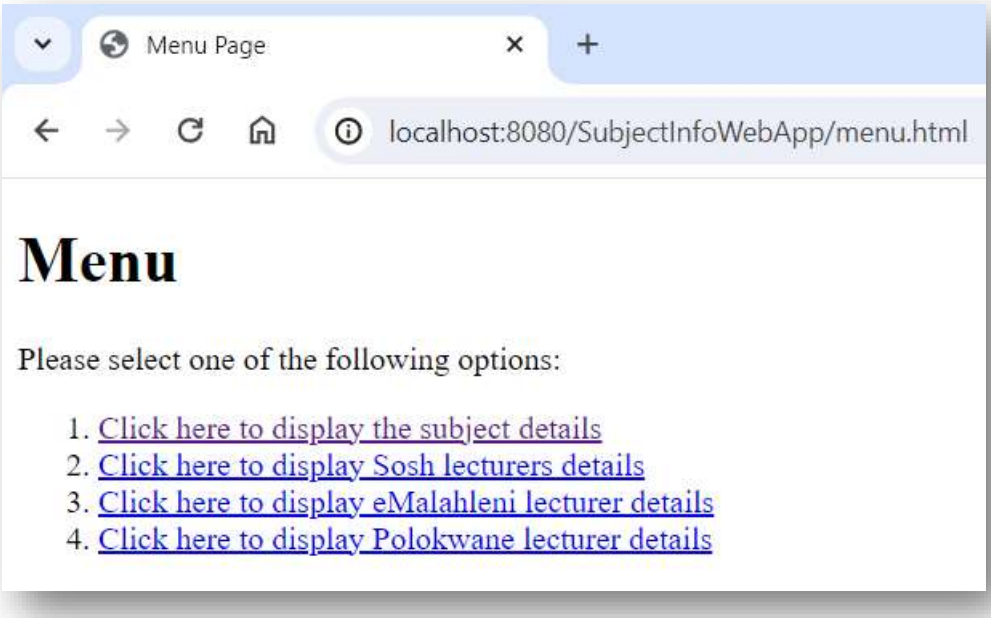
Connect the **SubjectDetailsServlet** to the JSP

```
web.xml | subject_details.jsp | SubjectDetailsServlet.java
Source | History
        Integer numQuizzes = Integer.parseInt(getServletContext().getInitParameter("num_quizzes"));
        Integer numProjects = Integer.parseInt(getServletContext().getInitParameter("num_projects"));

        System.out.println("Subject Details" + "\n" + "-----------------------");
        System.out.println("Subject name: " + subjectName);
        System.out.println("Subject code: " + subjectCode);
        System.out.println("Subject head name: " + subjectHeadName);
        System.out.println("Subject head email: " + subjectHeadEmail);
        System.out.println("Number of Class Tests to be written: " + numClassTests);
        System.out.println("Number of Semester Tests to be written: " + numSemesterTests);
        System.out.println("Number of Quizzes to be written: " + numQuizzes);
        System.out.println("Number of Projects to be done: " + numProjects);

        RequestDispatcher disp = request.getRequestDispatcher("subject_details.jsp");
        disp.forward(request, response);
    }
}
```

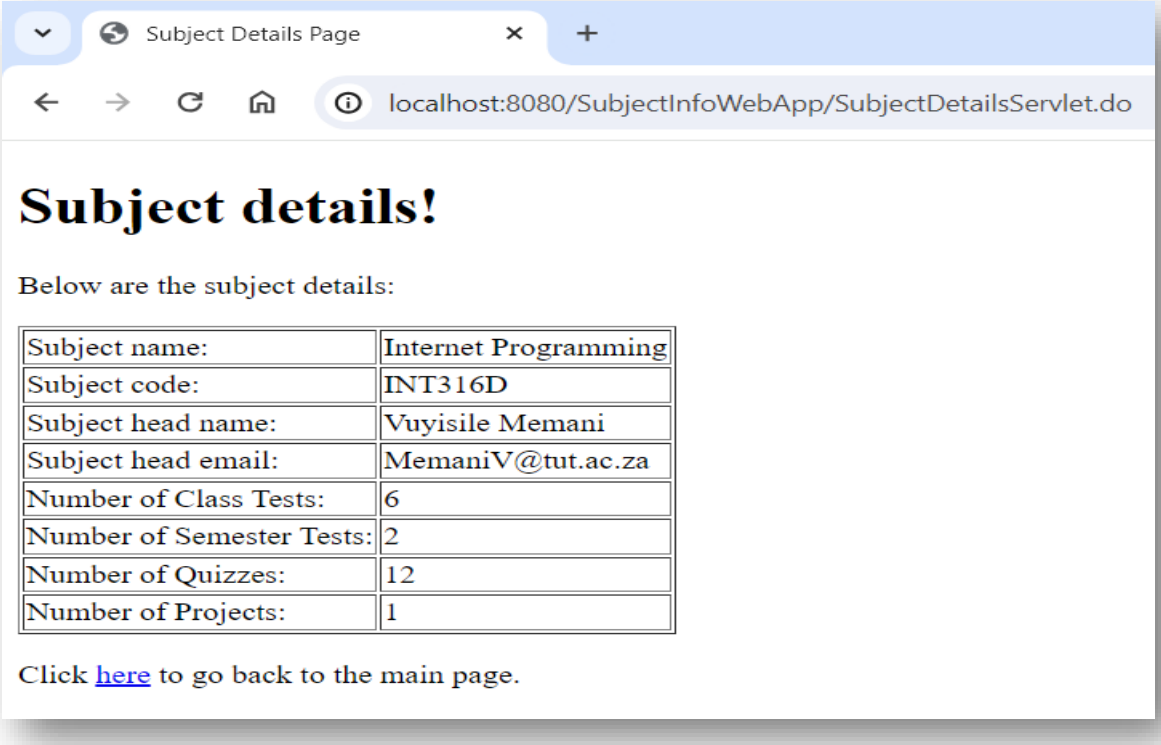
Clean and Build, Deploy, and Run the project.



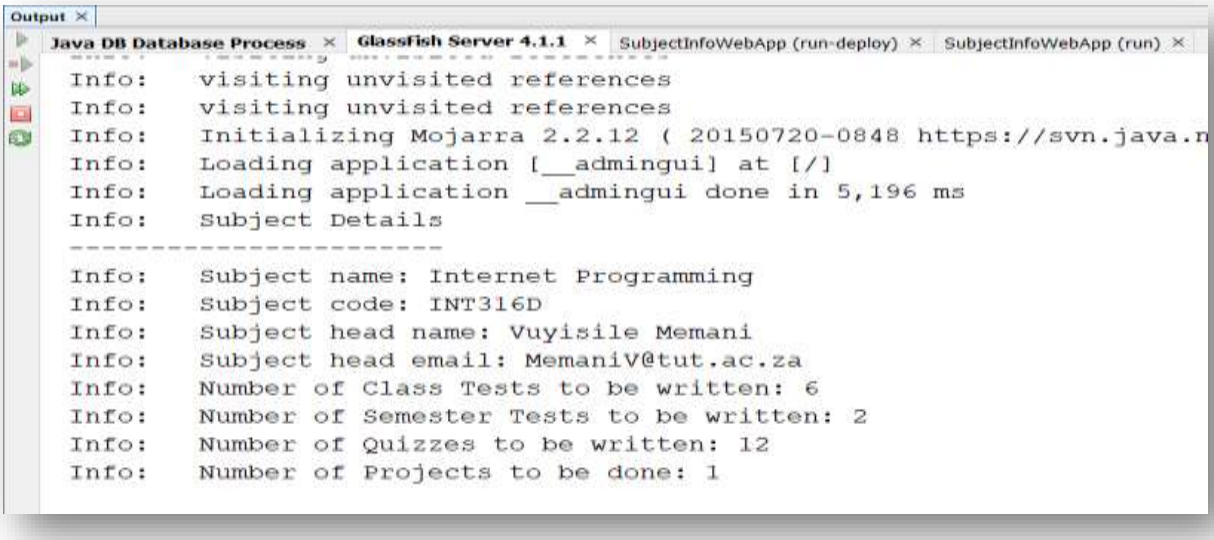
Click on the link.



Click on the first link



Also look on GlassFish output



Learning Moment (LM):

Context data defined insde the DD file is available throughout the components of a project.

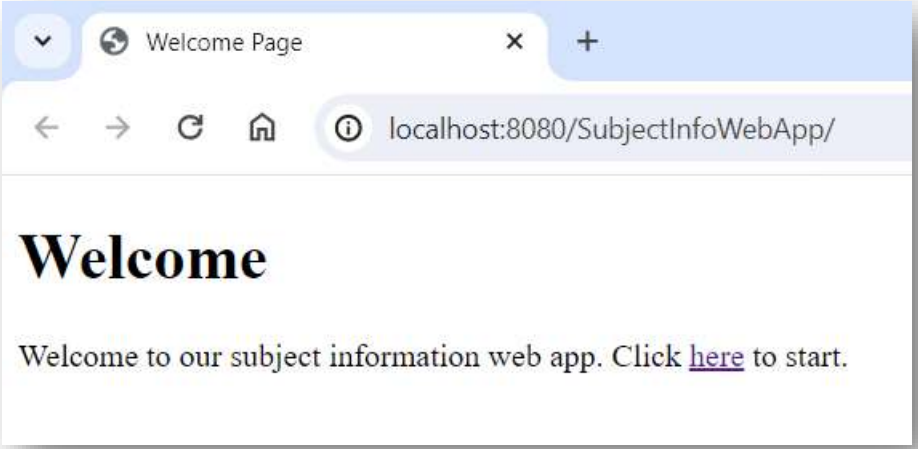
Create a servlet called **SoshLecturersServlet**.

```java
/**
 *
 * @author MemaniV
 */
public class SoshLecturersServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String firstLecturerName = getServletConfig().getInitParameter("lec1_name");
        String firstLecturerEmail = getServletConfig().getInitParameter("lec1_email");
        String secondLecturerName = getServletConfig().getInitParameter("lec2_name");
        String secondLecturerEmail = getServletConfig().getInitParameter("lec2_email");
        String thirdLecturerName = getServletConfig().getInitParameter("lec3_name");
        String thirdLecturerEmail = getServletConfig().getInitParameter("lec3_email");

        System.out.println("Sosh lecturers" + "\n" + "----------------------");
        System.out.println("Name: " + firstLecturerName);
        System.out.println("Email address: " + firstLecturerEmail);
        System.out.println("");
        System.out.println("Name: " + secondLecturerName);
        System.out.println("Email address: " + secondLecturerEmail);
        System.out.println("");
        System.out.println("Name: " + thirdLecturerName);
        System.out.println("Email address: " + thirdLecturerEmail);
        System.out.println("");
    }
}
```

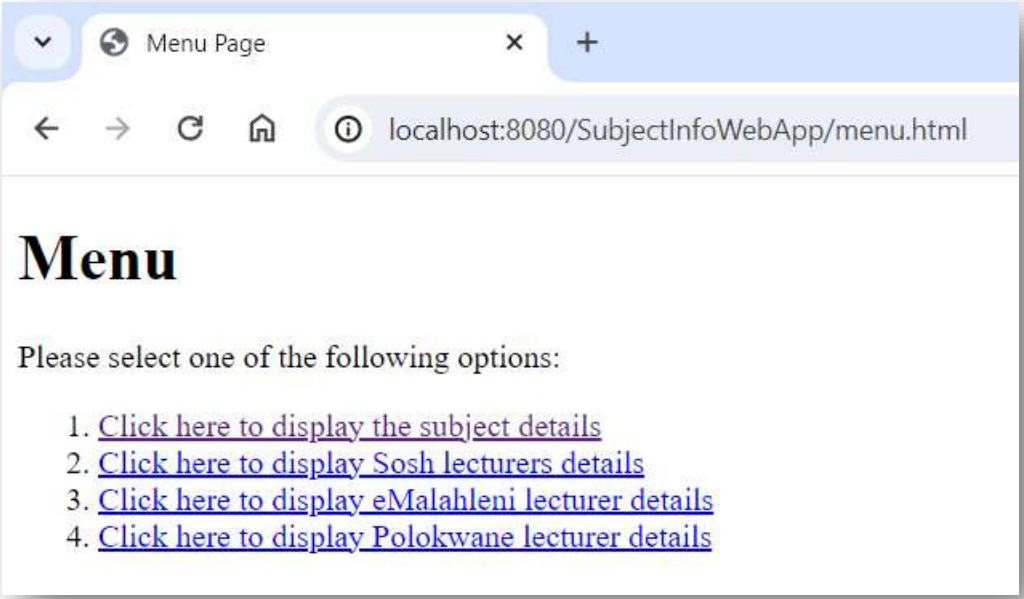
Include the parameters in the confifg element of the servlet in the DD

```xml
    <servlet>
        <servlet-name>SoshLecturersServlet</servlet-name>
        <servlet-class>za.ac.tut.web.SoshLecturersServlet</servlet-class>
        <init-param>
            <param-name>lec1_name</param-name>
            <param-value>Vuyisile Memani</param-value>
        </init-param>
        <init-param>
            <param-name>lec1_email</param-name>
            <param-value>MemaniV@tut.ac.za</param-value>
        </init-param>
        <init-param>
            <param-name>lec2_name</param-name>
            <param-value>Johnson Dehinbo</param-value>
        </init-param>
        <init-param>
            <param-name>lec2_email</param-name>
            <param-value>DehinboOJ@tut.ac.za</param-value>
        </init-param>
        <init-param>
            <param-name>lec3_name</param-name>
            <param-value>To be announced soon</param-value>
        </init-param>
        <init-param>
            <param-name>lec3_email</param-name>
            <param-value>N/A</param-value>
        </init-param>
    </servlet>
```

Clean and Build, deploy and run the project



Click on the link



Click on the second link

Look on the output of GlassFish

```
Info:   Sosh lecturers
------------------------
Info:   Name: Vuyisile Memani
Info:   Email address: MemaniV@tut.ac.za
Info:   Name: Johnson Dehinbo
Info:   Email address: DehinboOJ@tut.ac.za
Info:   Name: To be announced soon
Info:   Email address: N/A
```

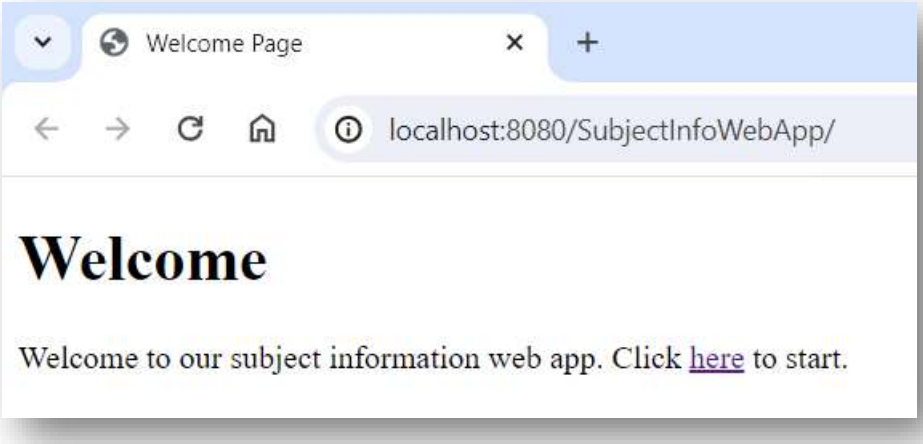
Connect the servlet to a JSP called **sosh_lecs.jsp.**

```
21      @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
23              throws ServletException, IOException {
24          String firstLecturerName = getServletConfig().getInitParameter("lec1_name");
25          String firstLecturerEmail = getServletConfig().getInitParameter("lec1_email");
26          String secondLecturerName = getServletConfig().getInitParameter("lec2_name");
27          String secondLecturerEmail = getServletConfig().getInitParameter("lec2_email");
28          String thirdLecturerName = getServletConfig().getInitParameter("lec3_name");
29          String thirdLecturerEmail = getServletConfig().getInitParameter("lec3_email");
30
31          System.out.println("Sosh lecturers" + "\n" + "------------------------");
32          System.out.println("Name: " + firstLecturerName);
33          System.out.println("Email address: " + firstLecturerEmail);
34          System.out.println("");
35          System.out.println("Name: " + secondLecturerName);
36          System.out.println("Email address: " + secondLecturerEmail);
37          System.out.println("");
38          System.out.println("Name: " + thirdLecturerName);
39          System.out.println("Email address: " + thirdLecturerEmail);
40          System.out.println("");
41
            RequestDispatcher disp = request.getRequestDispatcher("sosh_lecs.jsp");
43          disp.forward(request, response);
44      }
45  }
```

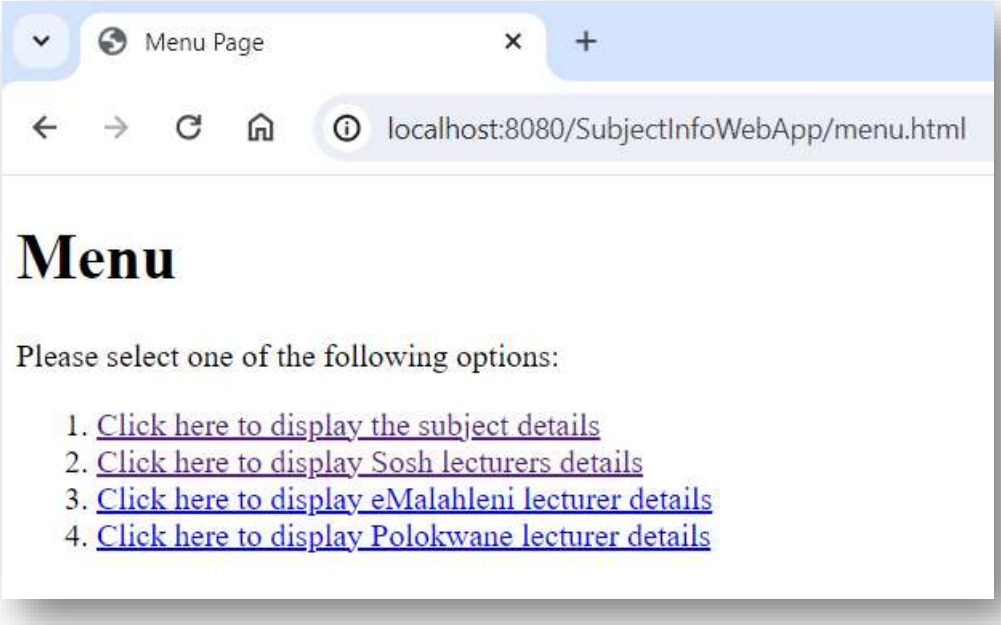
Create the sosh_lecs.jsp file

```
15          <h1>Soshanguve lecturers</h1>
16          <%
17              String firstLecturerName = config.getInitParameter("lec1_name");
18              String firstLecturerEmail = config.getInitParameter("lec1_email");
19              String secondLecturerName = config.getInitParameter("lec2_name");
20              String secondLecturerEmail = config.getInitParameter("lec2_email");
21              String thirdLecturerName = config.getInitParameter("lec3_name");
22              String thirdLecturerEmail = config.getInitParameter("lec3_email");
23          %>
24          <p>
25              Below are the details of Soshanguve lecturers:
26          </p>
27          <table border="1">
28              <th>Name</th>
29              <th>Email address</th>
                <tr>
31                  <td><%=firstLecturerName%></td>
32                  <td><%=firstLecturerEmail%></td>
33              </tr>
34              <tr>
35                  <td><%=secondLecturerName%></td>
36                  <td><%=secondLecturerEmail%></td>
37              </tr>
38              <tr>
39                  <td><%=thirdLecturerName%></td>
40                  <td><%=thirdLecturerEmail%></td>
41              </tr>
42          </table>
```

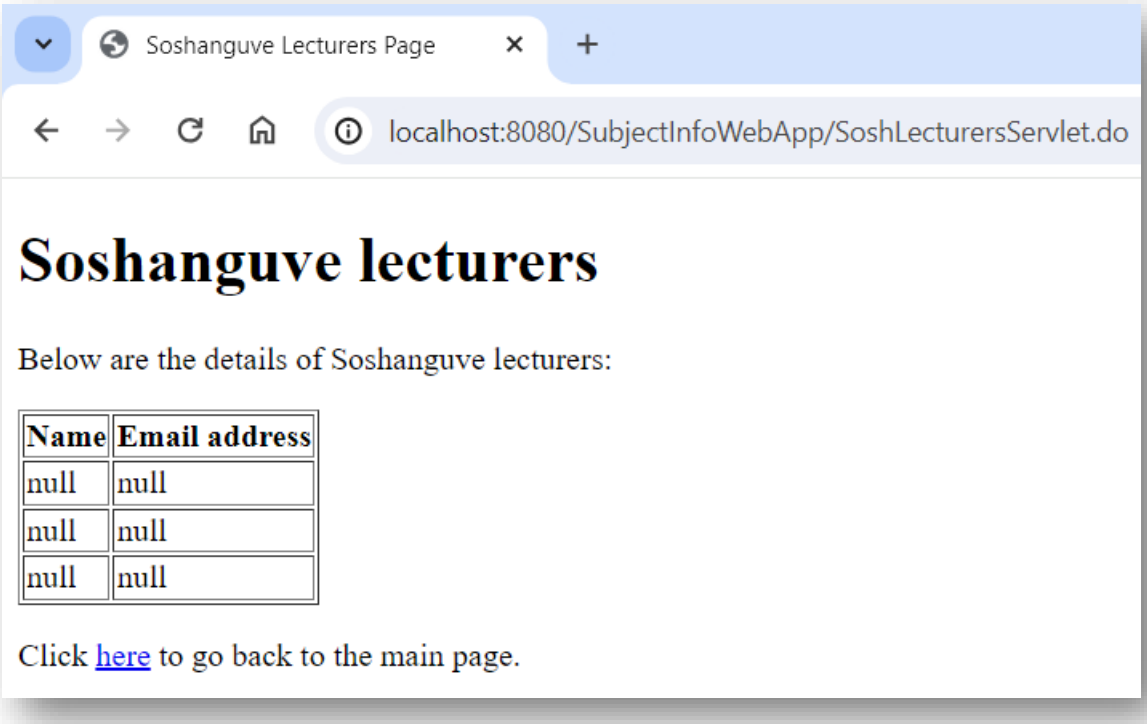
Clean and Build, deploy and run



Click on the link



Click on the second link



Look also on GlassFish output

```
Info:   Sosh lecturers
------------------------
Info:   Name: Vuyisile Memani
Info:   Email address: MemaniV@tut.ac.za
Info:   Name: Johnson Dehinbo
Info:   Email address: DehinboOJ@tut.ac.za
Info:   Name: To be announced soon
Info:   Email address: N/A
```

Learning Moment (LM):

ServletConfig is used to access init data for a specific servlet (local access)

ServletContext is used to access init data applicable to the entire project (global access)

1.7 DIY (Do It Yourself)

In this chapter we introduced you to mathematical operators. We showed you how each work. In this DIY, we want you to undertake three tasks in line with what you have learnt.

Task #1

Create a web application that will allow a user to login details. If the login details do not match the following:

- Username: app
- Password: 123

The application must display an error message and ask the user to contact and admin person with a given email address. The name of the admin and email must be context init parameters. If the user details match the given username and password, a welcome message must be displayed.

Task #2

Mujinga wants a web application that will allow him to play a coin guessing game with the computer. The game must be personalized with the user being addressed by their name and the computer taking the name of Siri.

When the application commences, it must ask the user to enter their name. After that a user must be allowed to enter a toss, that is either **heads** or **tails**. The user's toss value sent to the server for Siri to guess the toss. The guess must be compared to the toss and a winner determined and displayed. Assumming that the user is Sipho, a typical outcome could look as follows:

- **Sipho**: heads
- **Siri**: tails
- **Outcome**: Sipho has won

OR

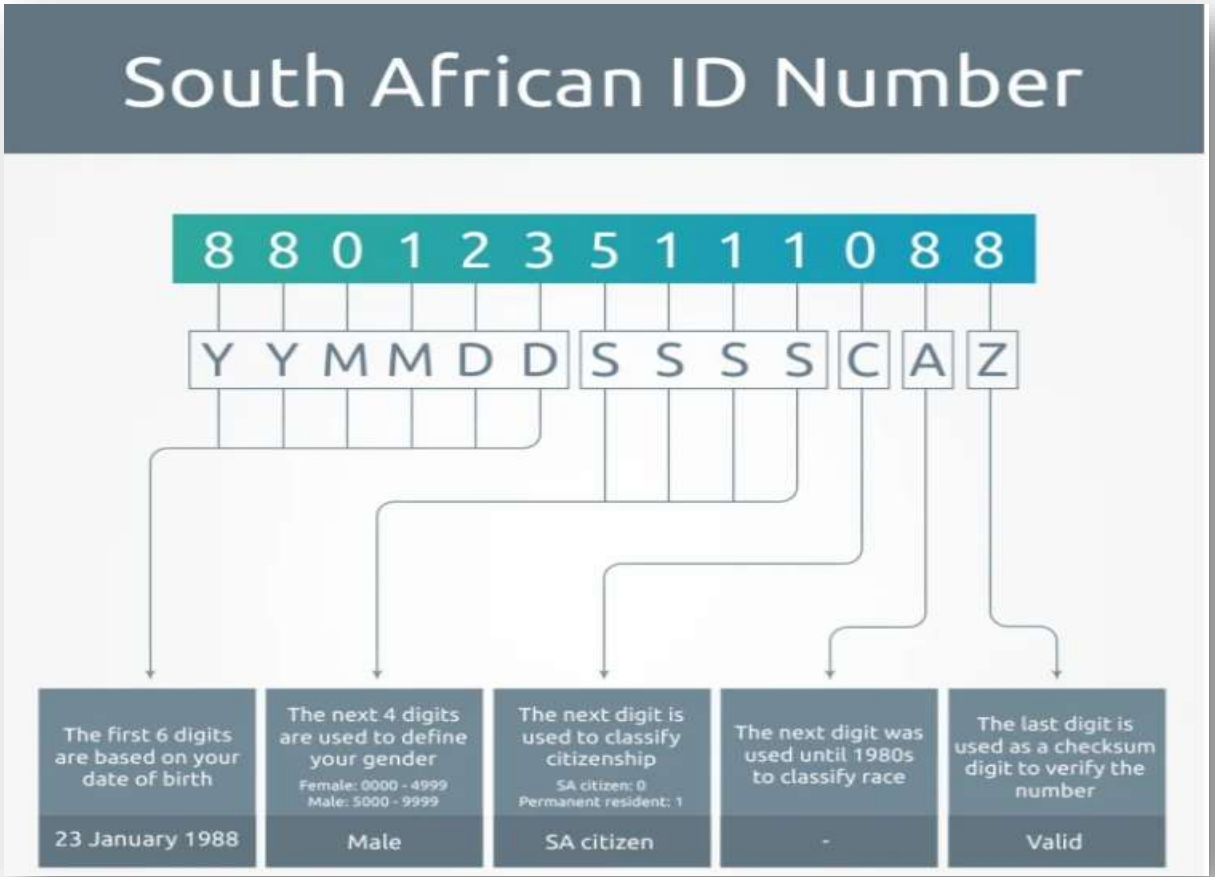
- **Sipho**: heads
- **Siri**: heads
- **Outcome**: Siri has won

After displaying the guess outcome, the application must start again. The Siri name must be a ServletContext or ServletCong parameter. You make the call.

Task #3

Sipho is a Software Developer at the Department of Home Affairs (DHA). He is working under the supervision of Ms Baloyi, a senior developer at DHA. As his first task, Ms Baloyi gives Sipho an opportunity of participating in the Identification Document (ID) Verification Project (ID-VP).

The task given to Sipho is to develop a web application that will determine whether a given ID is valid or not. The criteria for a valid ID is based on the ID number. Below is a figure describing the SA ID using a fictitious ID number:



So the SA ID has the following features:

- It is 13 digits long.
- The first six digits represent the date of birth (YYMMDD).
- The next four digits represent the gender of a person. Females are allocated the the range 0000 – 4999, and males 5000 - 9999).
- The next digit denotes citizenship status. The digit 0 denotes one was born a South African citizen, and 2 says the person is a permanent resident.
- The 12th digit is no longer used.
- The 13th is used to verify an ID using Luhn's algorithm.

Luhn's algorithm work as follows:

Every digit in an even position is doubled. The first digit is in postion 1, and the last digit is in position 13. So working with our fictitious ID we will have

8	8	0	1	2	3	5	1	1	1	0	8	8
	16		**2**		**6**		**2**		**2**		**16**	

If the double outcome is more than 9, modulo division is performed to get the remainder.

8	8	0	1	2	3	5	1	1	1	0	8	8
	7		**2**		**6**		**2**		**2**		**7**	

The digits in odd postions are added up to the new digits in even positions.

Sum = 8 + **7 +** 0 **+ 2 +** 2 **+ 6 +** 5 **+ 2 +** 1 **+ 2 +** 0 **+ 7 +** 8

= 50

A modulo division of 10 is perfromed on the sum. If the remainder is 0, the ID ais valid, otherwise it is invalid.

50 % 10 = 0

<u>To do</u>

Sipho is required to create a web application that will work as follows:

- Allow a user to enter their ID number.
- Determine whether the ID is valid or not.
- Generate a summary report that will show the following:
  - Display the ID number
  - Date of birth of the user.
  - Gender of the user.
  - Citizenship status
  - Verification outcome (valid or not valid)
  - Home Affairs query email (querries@dha.gov.za) and toll free number (087 7777 000). This must either be a context or config initialization

Assumming that you are Sipho, create such an application for DHA.

1.8 Conclusion

In this chapter we managed to introduce the student to Servlets. In the next chapter we will continue with the discussion. Thank you for having taken time to go through this chapter. Enjoy the rest of the day and God bless you.