1 Conversational web apps

In this chapter, we introduce the student to the concept of conversational web applications.

1.1 HTTP

HTTP is a communication protocol that allows computers (clients and servers) to communicate over the web. One important feature about HTTP is that it is **stateless**. This means, the protocol doesn't sustain or maintain a conversation taking place between a client an a server. Every connection between a client and a server is seen as a new connection, previous states are forgotten.

A connection between a client and a server is established as follows:

- The client (browser) sends basic information about itself (browser name, communication protocol being used, acceptable data types that the browser can work with, etc.) to the server.
- The server responds in kind by sending information about itself to the client (cookies).
- A connection is established between the two computers. This is called handshaking.
- A client then sends an actual request to the server.
- The server services the request and sends a response.
- The connection is disconnected.

When a client attempts to send a subsequent request, it will not be remembered by the server. It will be seen as a new connection. Handshaking takes place, a new connection is established, communication (request/response) takes place.

The statelessness of HTTP is a flaw that makes it difficult to create conversational web applications. Conversational web apps need communication that is sustained a client and the server. The server needs to remember previous conversations it had with the same client in order to accomplish a task. A classical example is the online shopping web applications. When a client adds an item to the buying cart, subsequent requests from the same client must not be seen or treated as coming from a new connection. They must be recognized as requests coming from the same connection established earlier. If a user wants to add another item, that item must find its place in the cart, added on top of the previous item.

But the challenge is the statelessness of HTTP. The question is ***"How to we make HTTP stateful?"***

1.1.1 How to make HTTP stateful?

Stateful simply means to remember previous states. HTTP is stateless, it only recalls the current state. To counter this weakness of HTTP, we use **sessions** and **url encoding** to make HTTP stateful.

1.1.2 Using sessions

When a client performs a handshake with a server, the server establishes a session with the client and sends to the client a session ID. The session ID is used to identify the client every time a request is made to the server. As a result, every time these two communicate, they exchange the session ID. Session IDs are shared between computers using cookies. Cookies are just objects that contain information about each computer involved in the exchange of messages.

JEE provides us with a class called **HttpSession** which is used to create a session between the server and the client. An example will be done that shows the implementation of sessions in a web development environment.

1.1.3 Using url encoding

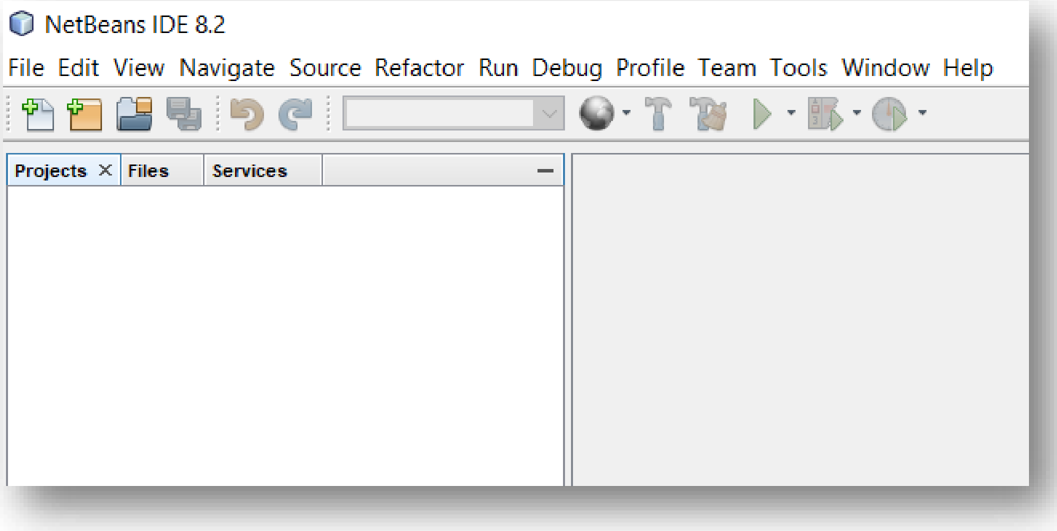
URL encoding is another technique used to establish a session between computers. Through encoding, the session ID is appended to the URL every time the server and client communicate. It is extracted and contrasted to the stored session in each computer. If a match exist, it means the two computers have communicated before, the request is not from a new connection. An example will be done to demonstrate this technique.

1.2 Example 1

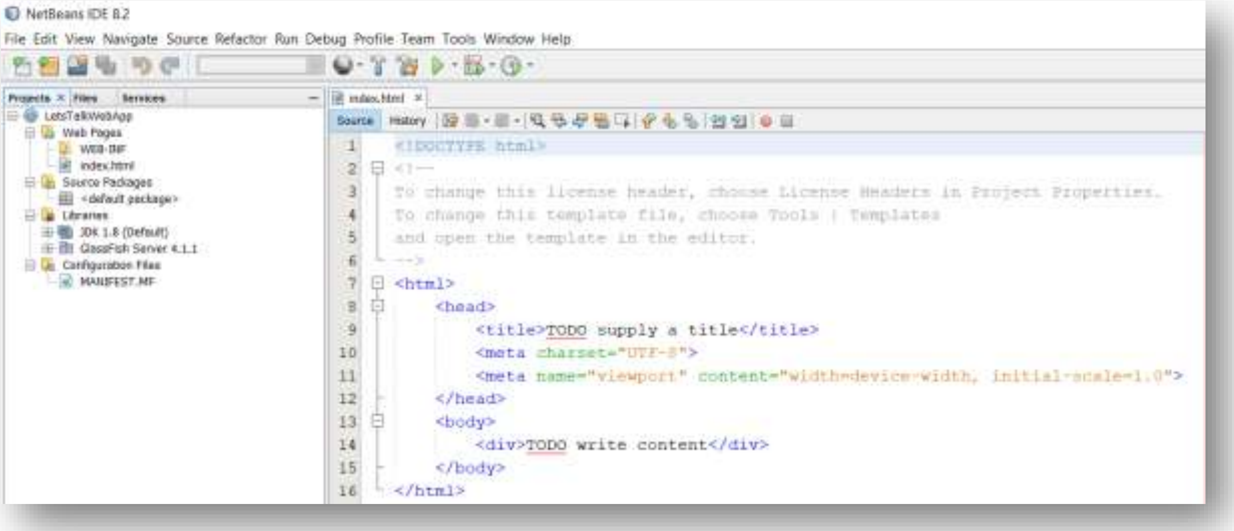
In this example we create a basic web application that uses sessions. The applications holds social conversation with a user.

Solution

Launch NetBeans

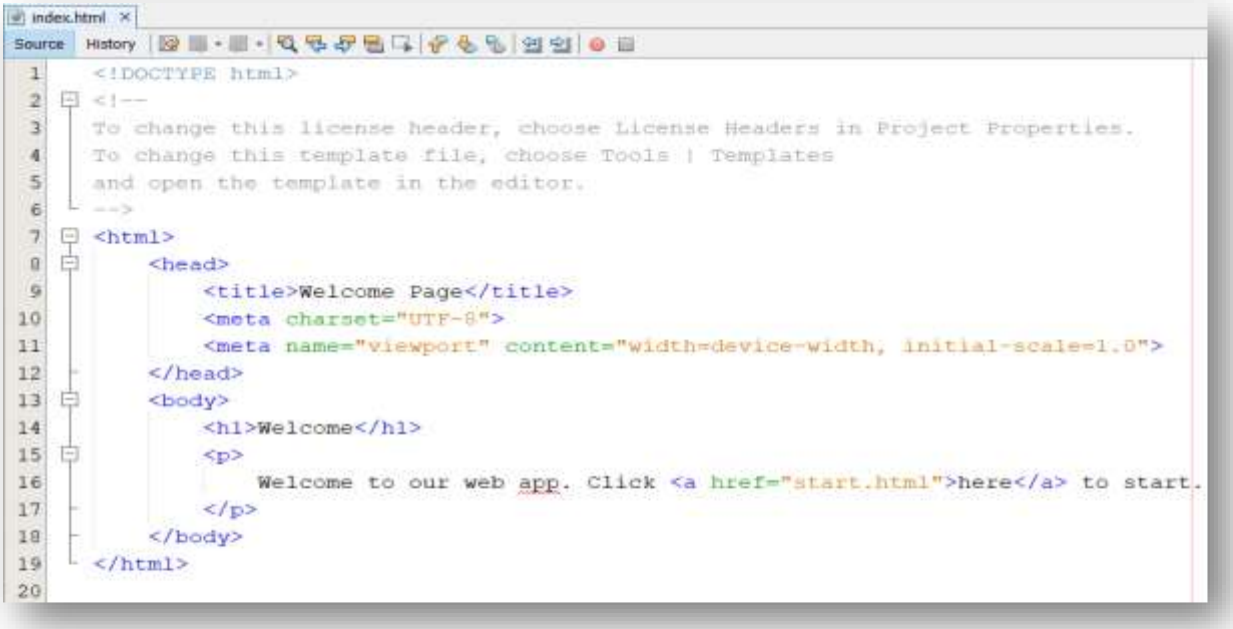

NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Files
Services

Create a web project called **LetsTalkWebApp**



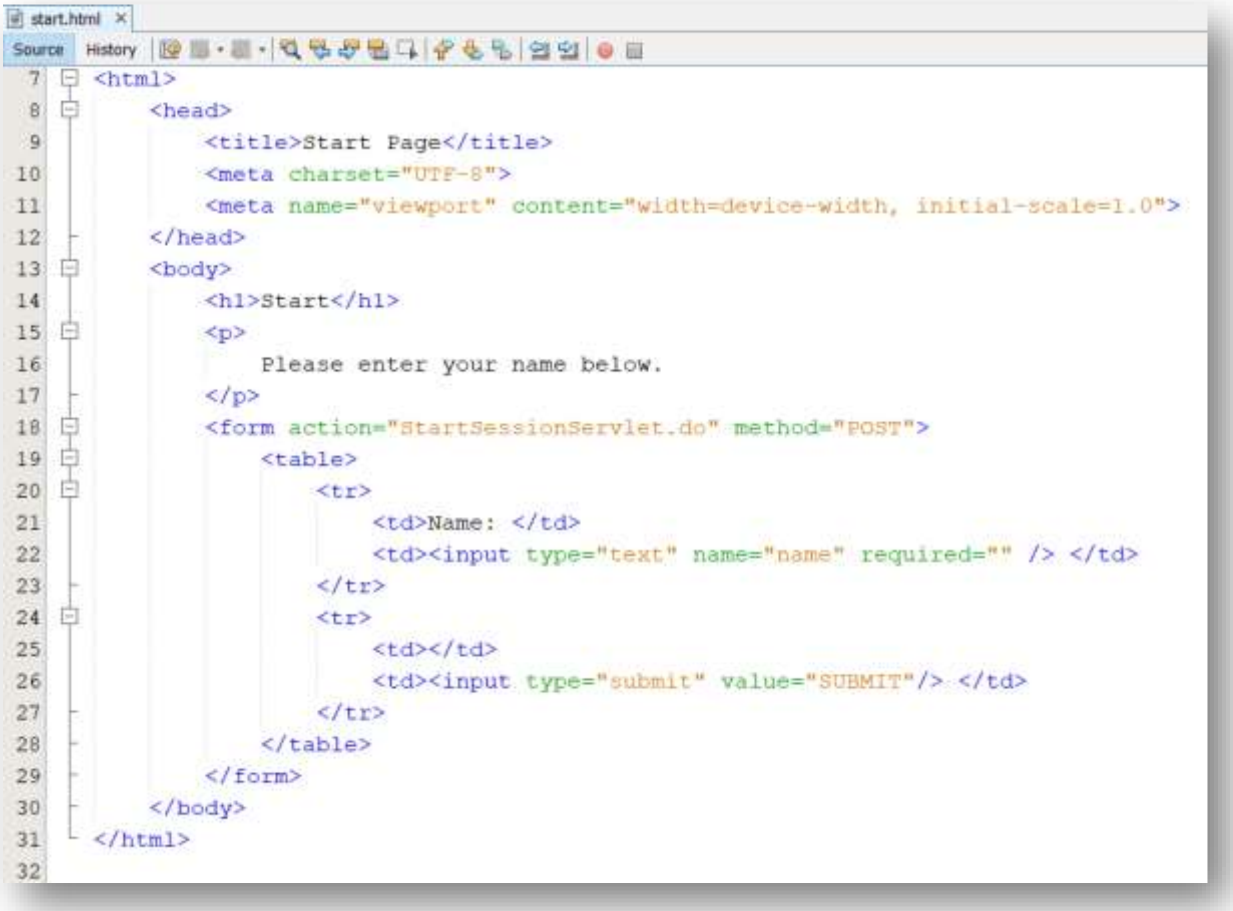
```html
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```

Modify the **index** page



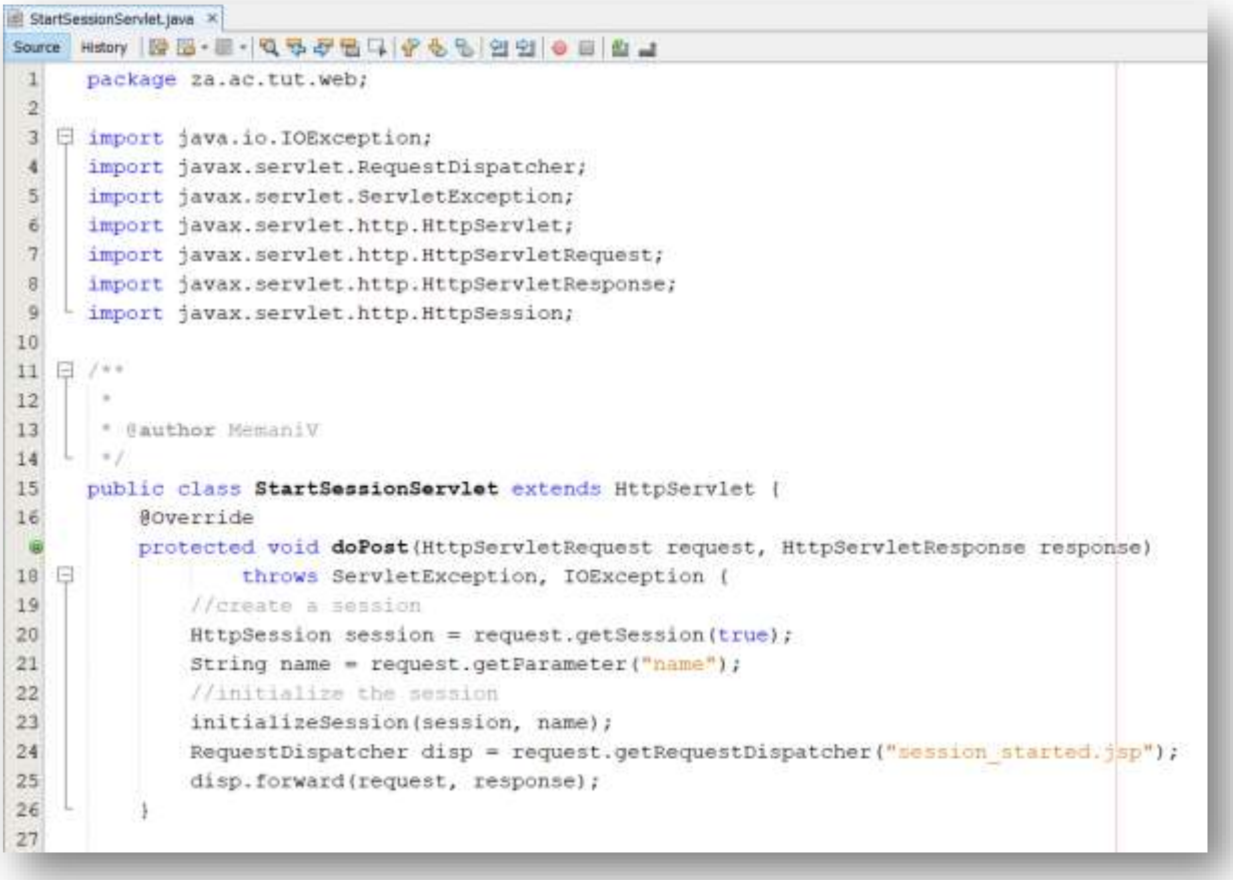
```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>Welcome Page</title>
        <meta charset="UTF-8">
        <meta name="viewport"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

Create the **start** page

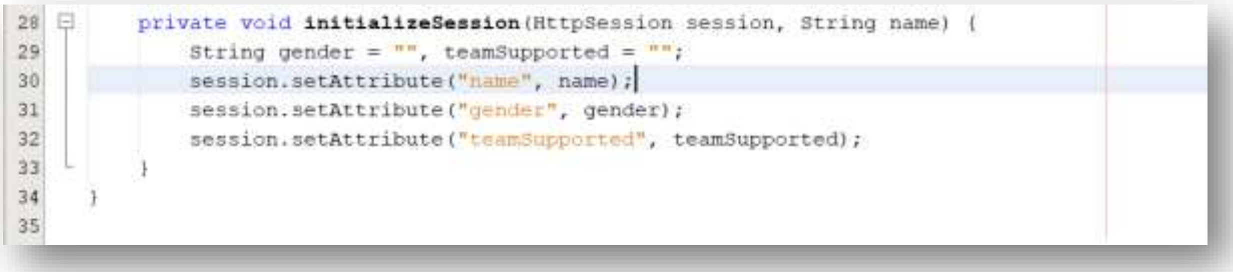


```
<html>
    <head>
        <title>Start Page</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Start</h1>
        <p>
            Please enter your name below.
        </p>
        <form action="StartSessionServlet.do" method="POST">
            <table>
                <tr>
                    <td>Name: </td>
                    <td><input type="text" name="name" required="" /> </td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="SUBMIT"/> </td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

Create the **StartSessionServlet**

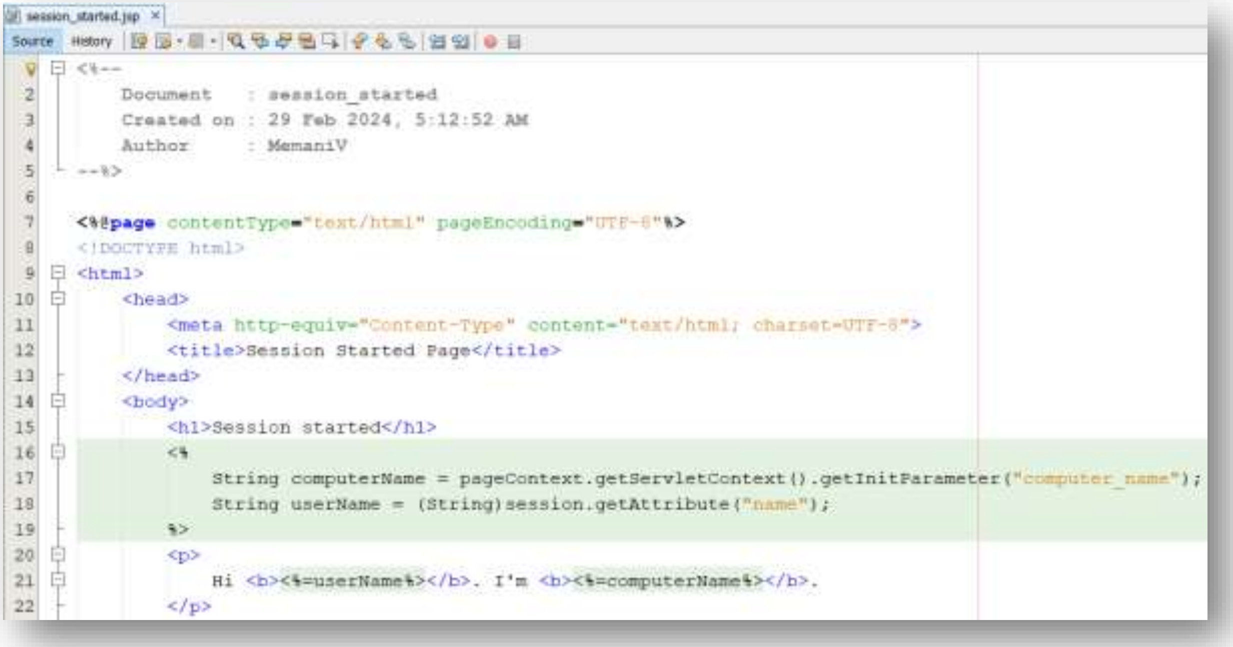


```java
package za.ac.tut.web;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
 * @author MemaniV
 */
public class StartSessionServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //create a session
        HttpSession session = request.getSession(true);
        String name = request.getParameter("name");
        //initialize the session
        initializeSession(session, name);
        RequestDispatcher disp = request.getRequestDispatcher("session_started.jsp");
        disp.forward(request, response);
    }

```

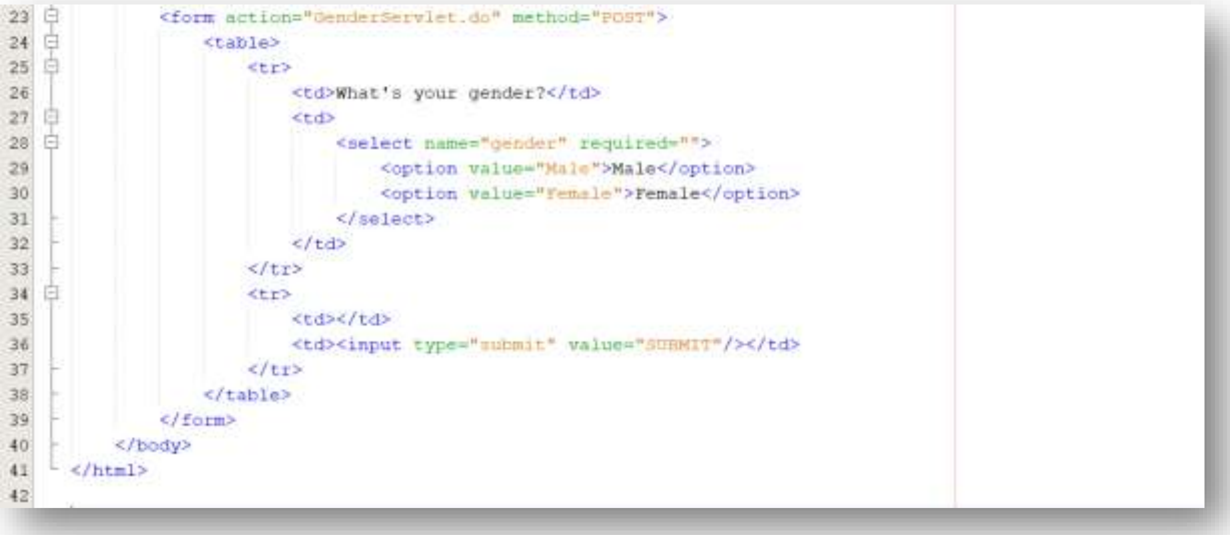


```java
    private void initializeSession(HttpSession session, String name) {
        String gender = "", teamSupported = "";
        session.setAttribute("name", name);
        session.setAttribute("gender", gender);
        session.setAttribute("teamSupported", teamSupported);
    }
}

```

Create the **session_started** file

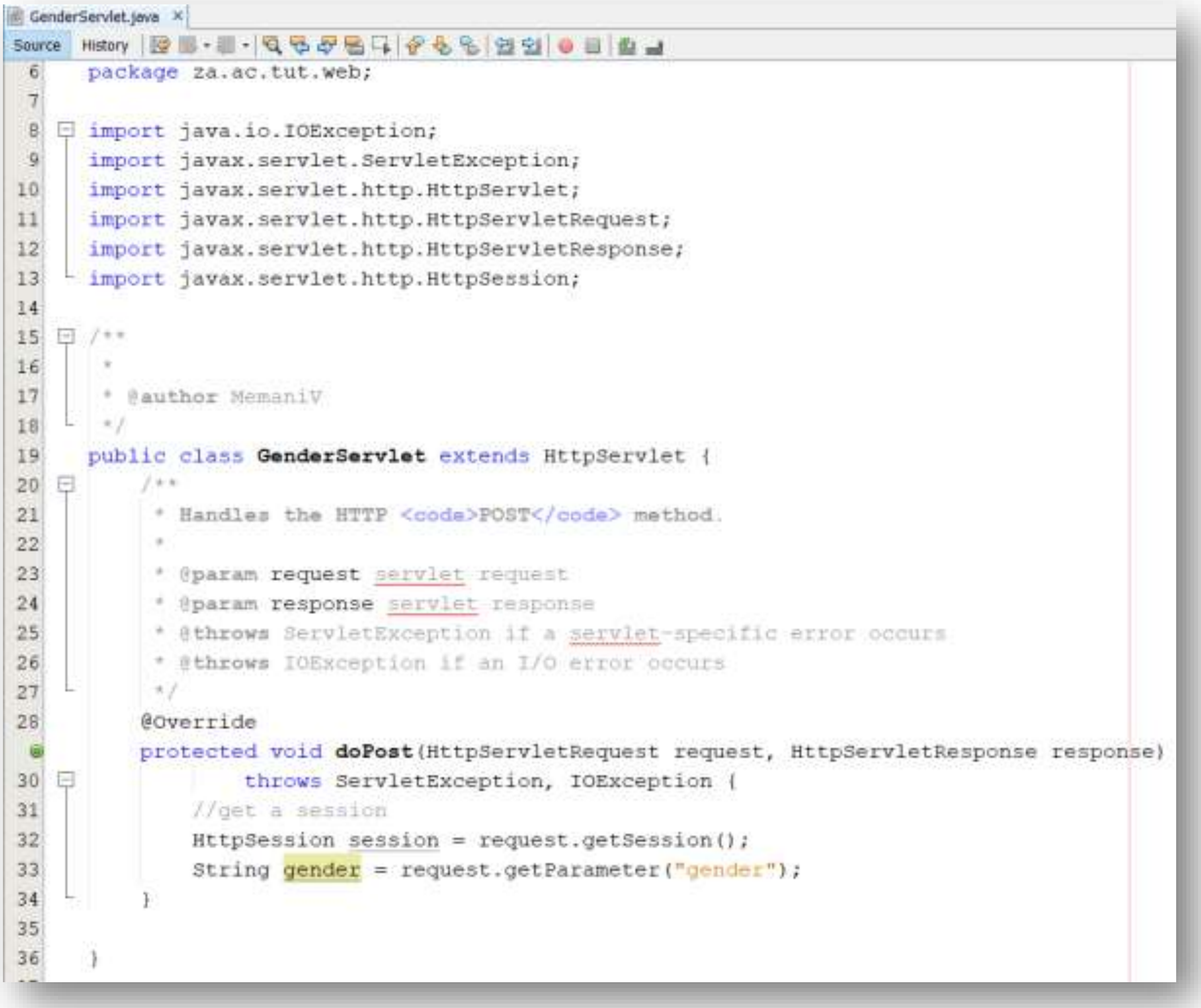


```
<%--
    Document   : session_started
    Created on : 29 Feb 2024, 5:12:52 AM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Session Started Page</title>
    </head>
    <body>
        <h1>Session started</h1>
        <%
            String computerName = pageContext.getServletContext().getInitParameter("computer_name");
            String userName = (String)session.getAttribute("name");
        %>
        <p>
            Hi <b><%=userName%></b>. I'm <b><%=computerName%></b>.
        </p>
```



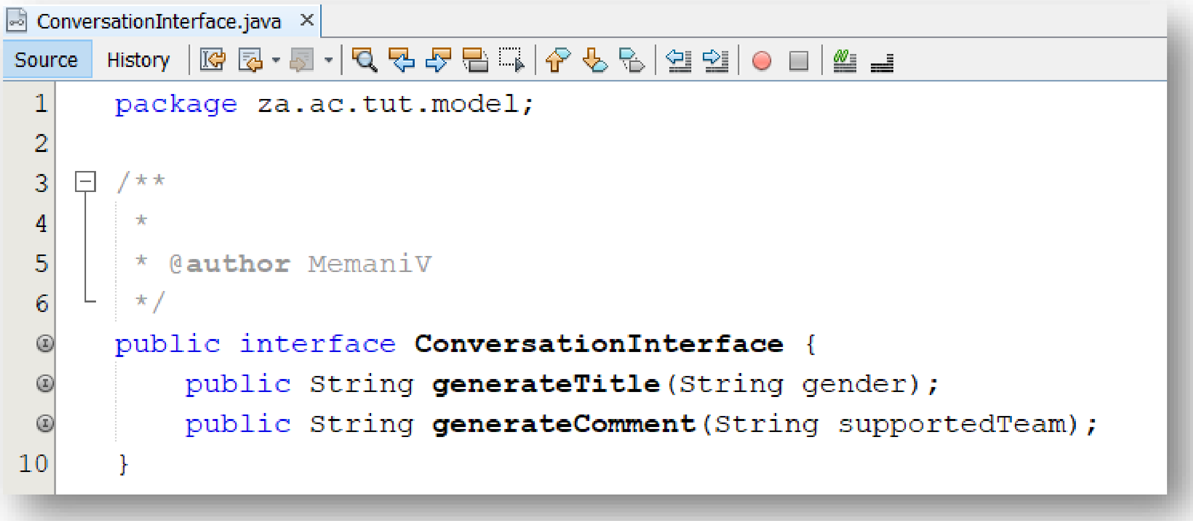
```
        <form action="GenderServlet.do" method="POST">
            <table>
                <tr>
                    <td>What's your gender?</td>
                    <td>
                        <select name="gender" required="">
                            <option value="Male">Male</option>
                            <option value="Female">Female</option>
                        </select>
                    </td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="SUBMIT"/></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

Create the **GenderServlet**



```
package za.ac.tut.web;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
 * @author MemaniV
 */
public class GenderServlet extends HttpServlet {
    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //get a session
        HttpSession session = request.getSession();
        String gender = request.getParameter("gender");
    }

}
```

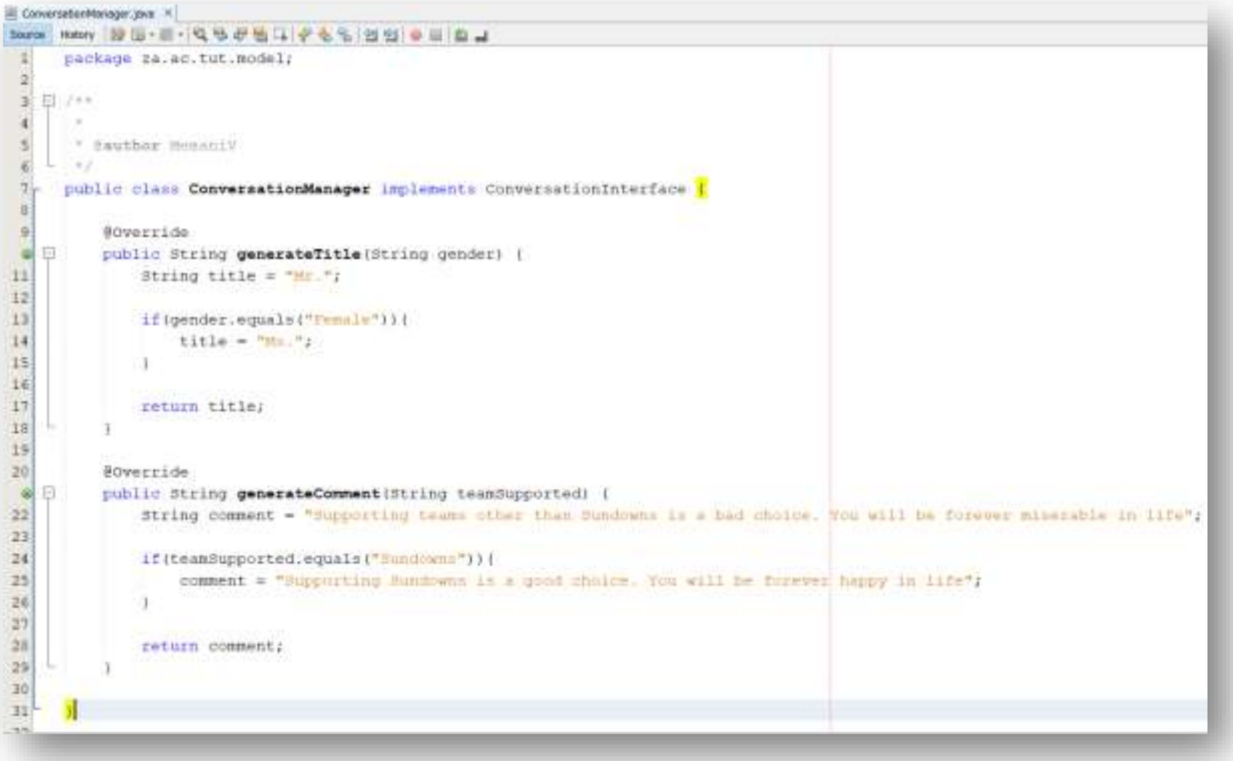
Create an interface **ConversationInterface**



ConversationInterface.java

```
package za.ac.tut.model;

/**
 *
 * @author MemaniV
 */
public interface ConversationInterface {
    public String generateTitle(String gender);
    public String generateComment(String supportedTeam);
}
```

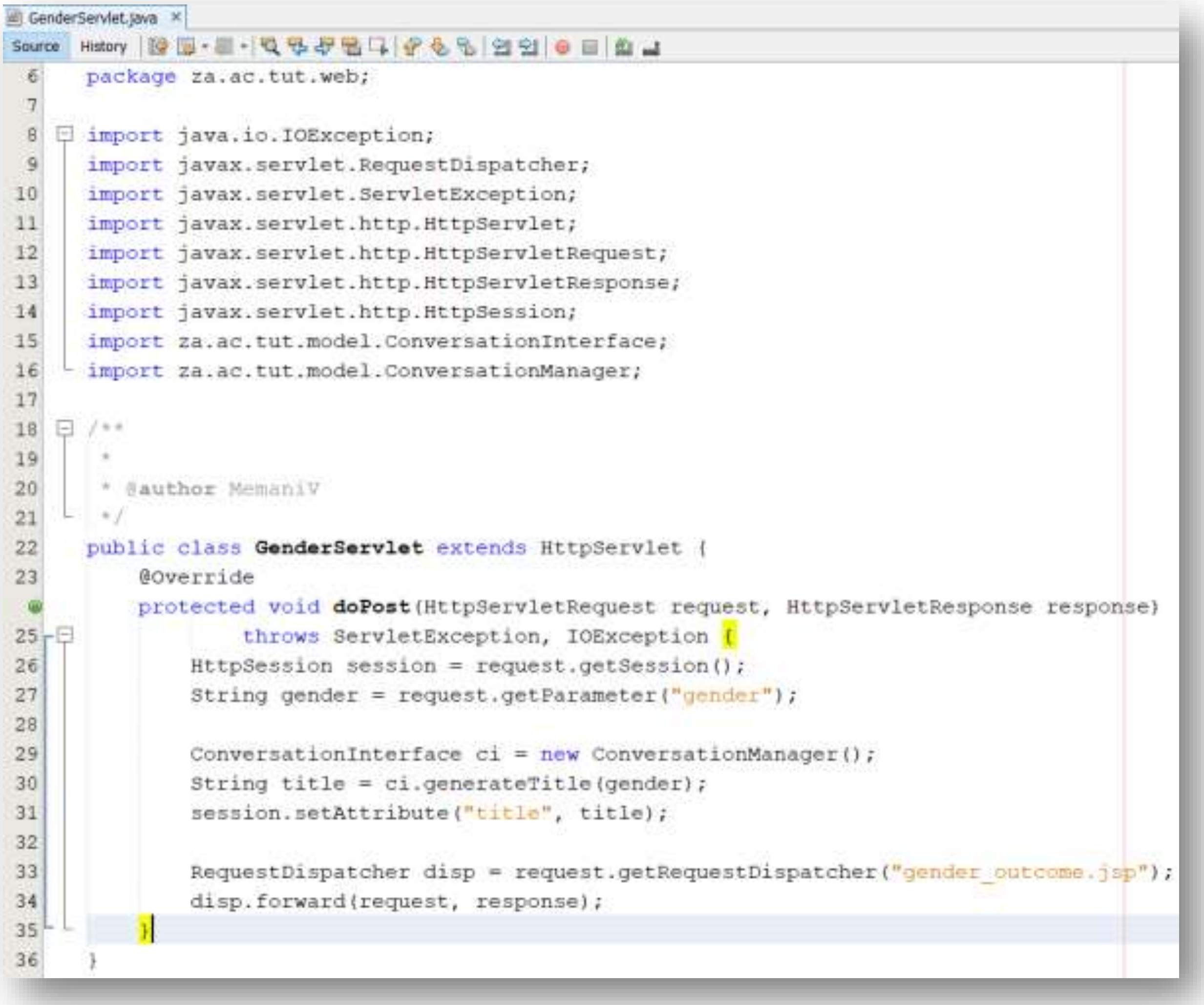
Create a class called **ConversationManager**



ConversationManager.java

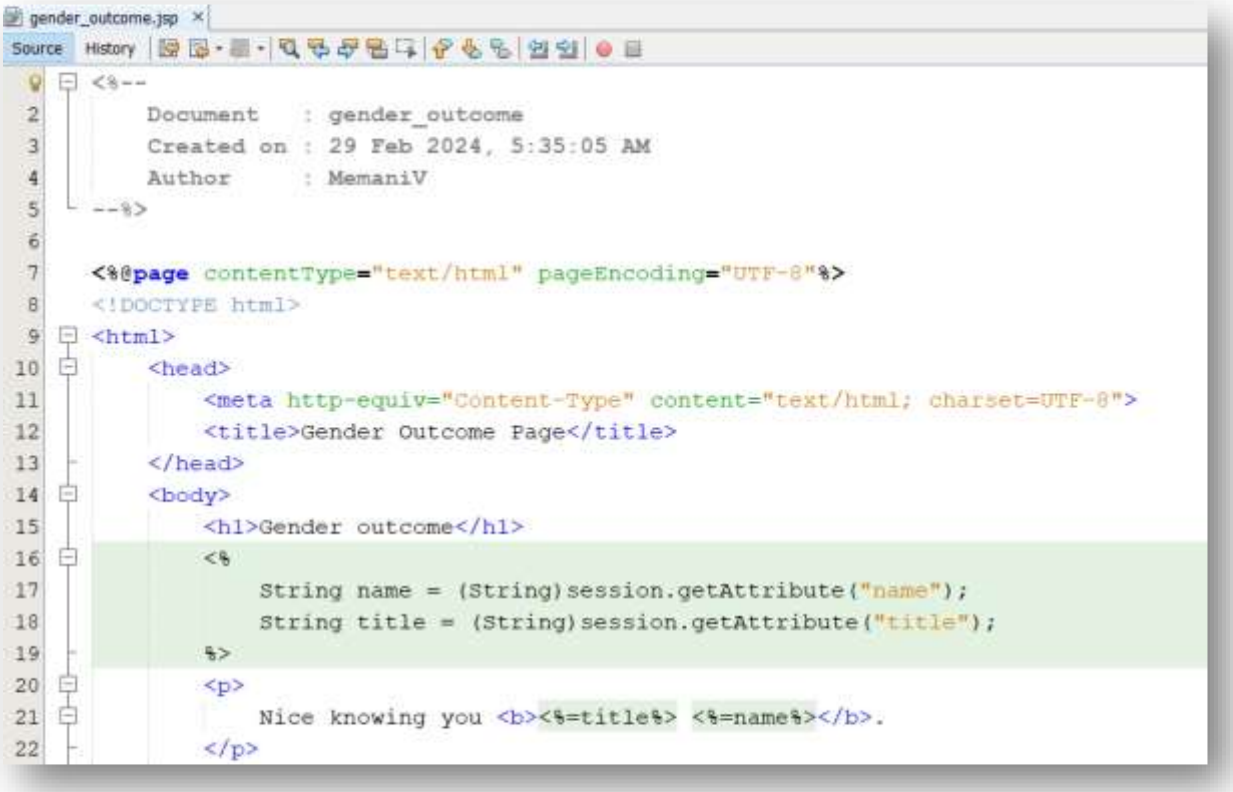
```
package za.ac.tut.model;

/**
 *
 * @author MemaniV
 */
public class ConversationManager implements ConversationInterface {

    @Override
    public String generateTitle(String gender) {
        String title = "Mr.";

        if(gender.equals("Female")){
            title = "Ms.";
        }

        return title;
    }

    @Override
    public String generateComment(String teamSupported) {
        String comment = "Supporting teams other than Sundowns is a bad choice. You will be forever miserable in life";

        if(teamSupported.equals("Sundowns")){
            comment = "Supporting Sundowns is a good choice. You will be forever happy in life";
        }

        return comment;
    }

}
```

Edit the servlet

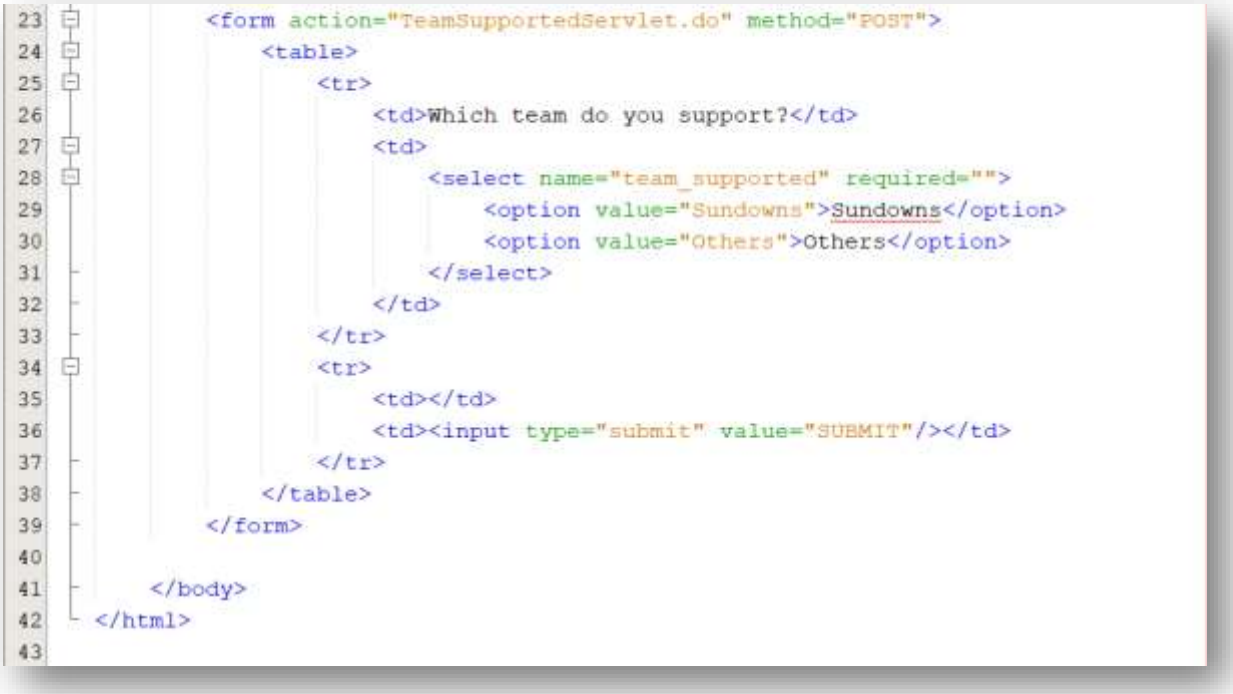


```java
package za.ac.tut.web;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import za.ac.tut.model.ConversationInterface;
import za.ac.tut.model.ConversationManager;

/**
 *
 * @author MemaniV
 */
public class GenderServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        HttpSession session = request.getSession();
        String gender = request.getParameter("gender");

        ConversationInterface ci = new ConversationManager();
        String title = ci.generateTitle(gender);
        session.setAttribute("title", title);

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

Create the **gender_outcome** page

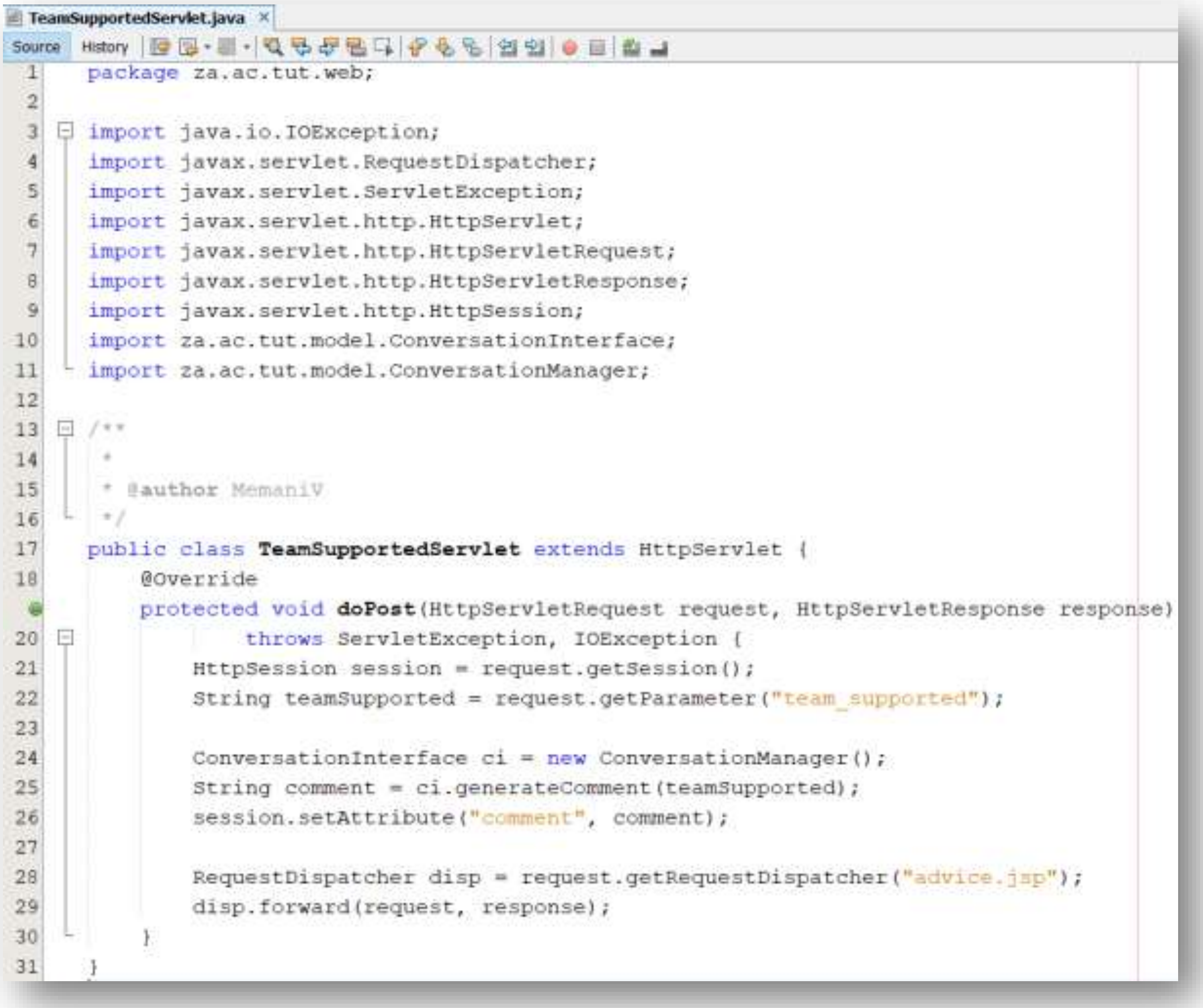


```
<%--
    Document   : gender_outcome
    Created on : 29 Feb 2024, 5:35:05 AM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Gender Outcome Page</title>
    </head>
    <body>
        <h1>Gender outcome</h1>
        <%
            String name = (String)session.getAttribute("name");
            String title = (String)session.getAttribute("title");
        %>
        <p>
            Nice knowing you <b><%=title%> <%=name%></b>.
        </p>
```



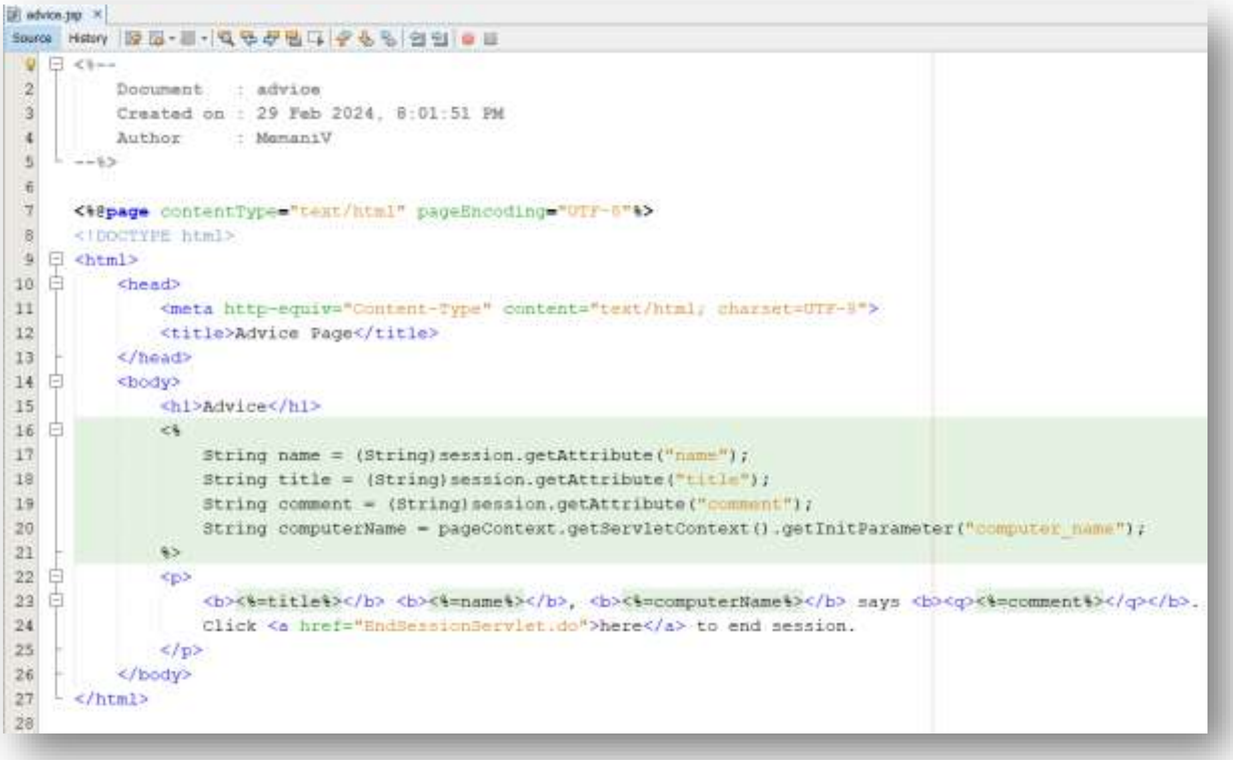
```
        <form action="TeamSupportedServlet.do" method="POST">
            <table>
                <tr>
                    <td>Which team do you support?</td>
                    <td>
                        <select name="team_supported" required="">
                            <option value="Sundowns">Sundowns</option>
                            <option value="Others">Others</option>
                        </select>
                    </td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="SUBMIT"/></td>
                </tr>
            </table>
        </form>

    </body>
</html>
```

Create the **TeamSupportedServlet**



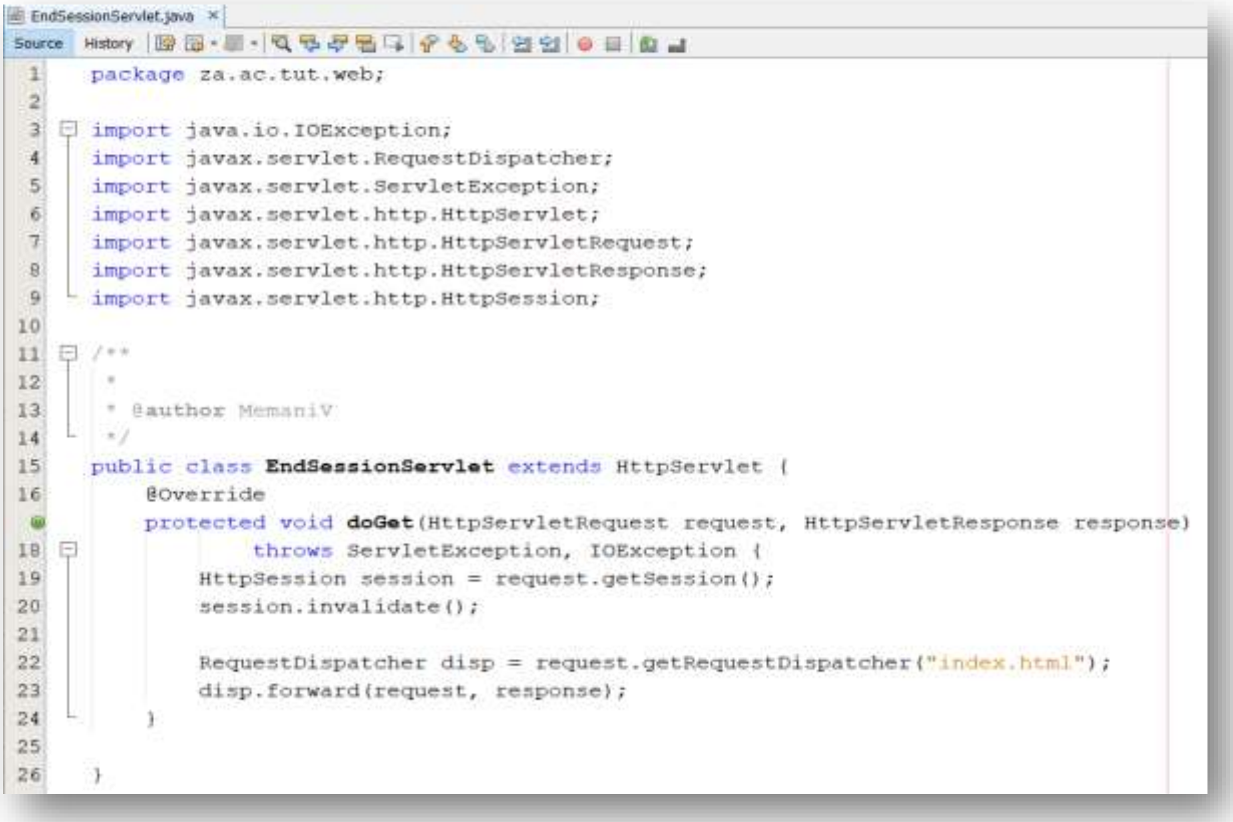
```java
package za.ac.tut.web;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import za.ac.tut.model.ConversationInterface;
import za.ac.tut.model.ConversationManager;

/**
 *
 * @author MemaniV
 */
public class TeamSupportedServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        HttpSession session = request.getSession();
        String teamSupported = request.getParameter("team_supported");

        ConversationInterface ci = new ConversationManager();
        String comment = ci.generateComment(teamSupported);
        session.setAttribute("comment", comment);

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

Create the **advice.jsp** file



```
<%--
    Document   : advice
    Created on : 29 Feb 2024, 8:01:51 PM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Advice Page</title>
    </head>
    <body>
        <h1>Advice</h1>
        <%
            String name = (String)session.getAttribute("name");
            String title = (String)session.getAttribute("title");
            String comment = (String)session.getAttribute("comment");
            String computerName = pageContext.getServletContext().getInitParameter("computer_name");
        %>
        <p>
            <b><%=title%></b> <b><%=name%></b>, <b><%=computerName%></b> says <b><q><%=comment%></q></b>.
            Click <a href="EndSessionServlet.do">here</a> to end session.
        </p>
    </body>
</html>
```

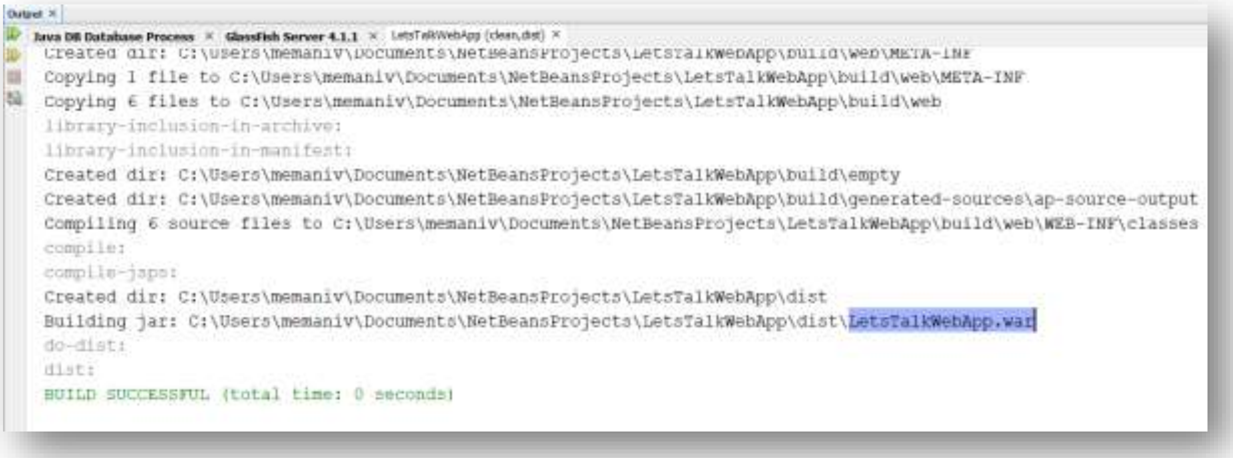
Create the **EndSessionServlet**



```
package za.ac.tut.web;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
 * @author MemaniV
 */
public class EndSessionServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        HttpSession session = request.getSession();
        session.invalidate();

        RequestDispatcher disp = request.getRequestDispatcher("index.html");
        disp.forward(request, response);
    }

}
```

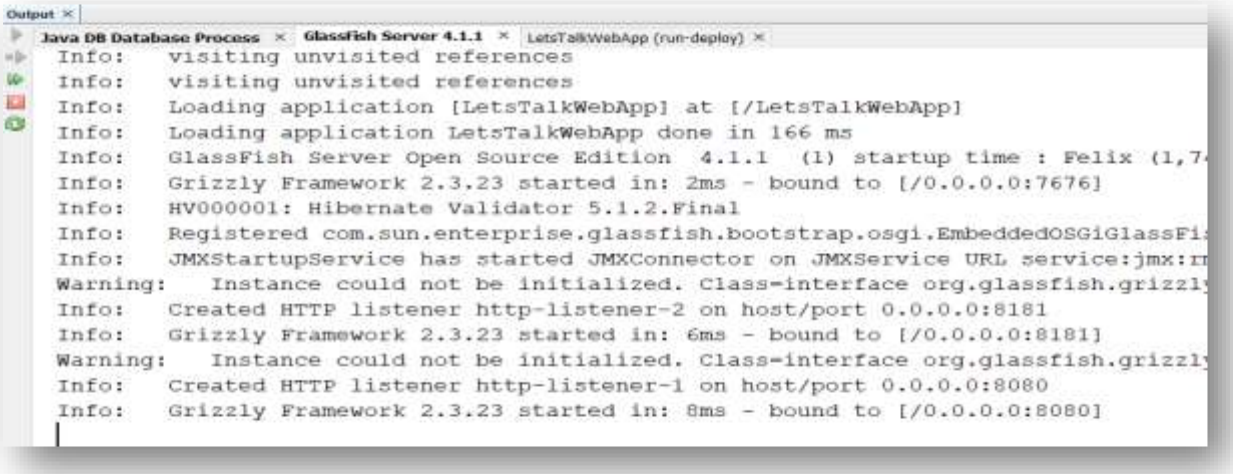
Run the program

Build and clean the project



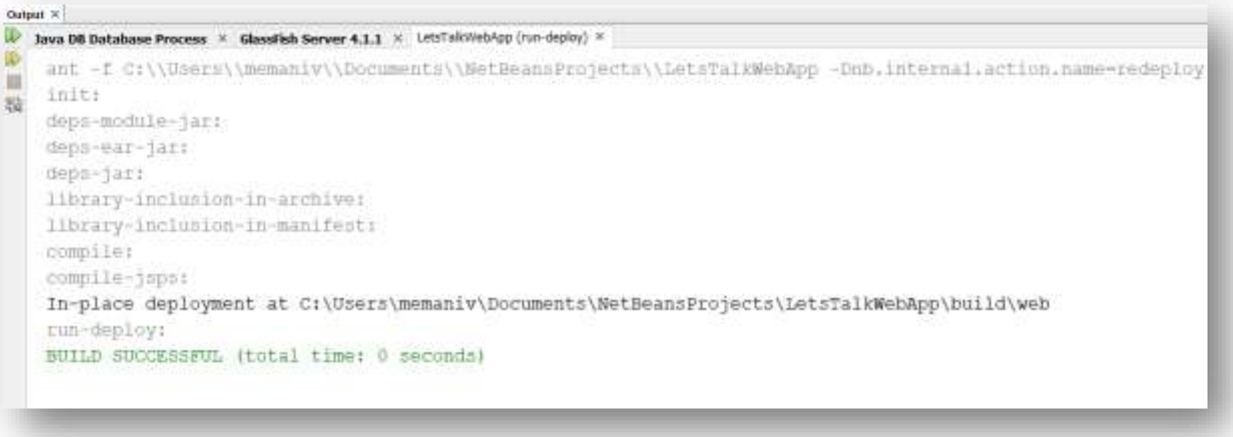
```
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\build\web\META-INF
Copying 6 files to C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\build\web
library-inclusion-in-archive:
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\build\generated-sources\ap-source-output
Compiling 6 source files to C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\build\web\WEB-INF\classes
compile:
compile-jsps:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\dist\LetsTalkWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Start GlassFish



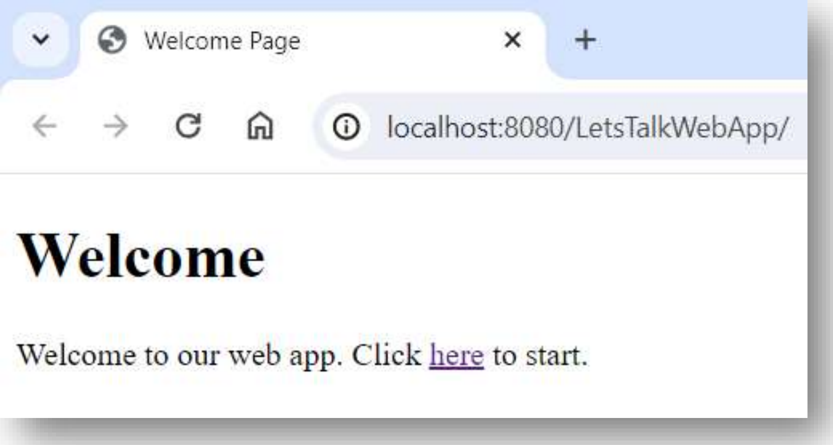
```
Info:   visiting unvisited references
Info:   visiting unvisited references
Info:   Loading application [LetsTalkWebApp] at [/LetsTalkWebApp]
Info:   Loading application LetsTalkWebApp done in 166 ms
Info:   GlassFish Server Open Source Edition  4.1.1  (1) startup time : Felix (1,7
Info:   Grizzly Framework 2.3.23 started in: 2ms - bound to [/0.0.0.0:7676]
Info:   HV000001: Hibernate Validator 5.1.2.Final
Info:   Registered com.sun.enterprise.glassfish.bootstrap.osgi.EmbeddedOSGiGlassFi
Info:   JMXStartupService has started JMXConnector on JMXService URL service:jmx:rm
Warning:   Instance could not be initialized. Class=interface org.glassfish.grizzly
Info:   Created HTTP listener http-listener-2 on host/port 0.0.0.0:8181
Info:   Grizzly Framework 2.3.23 started in: 6ms - bound to [/0.0.0.0:8181]
Warning:   Instance could not be initialized. Class=interface org.glassfish.grizzly
Info:   Created HTTP listener http-listener-1 on host/port 0.0.0.0:8080
Info:   Grizzly Framework 2.3.23 started in: 8ms - bound to [/0.0.0.0:8080]
```

Deploy the project



```
ant -f C:\\Users\\memaniv\\Documents\\NetBeansProjects\\LetsTalkWebApp -Dnb.internal.action.name=redeploy
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
library-inclusion-in-archive:
library-inclusion-in-manifest:
compile:
compile-jsps:
In-place deployment at C:\Users\memaniv\Documents\NetBeansProjects\LetsTalkWebApp\build\web
run-deploy:
BUILD SUCCESSFUL (total time: 0 seconds)
```

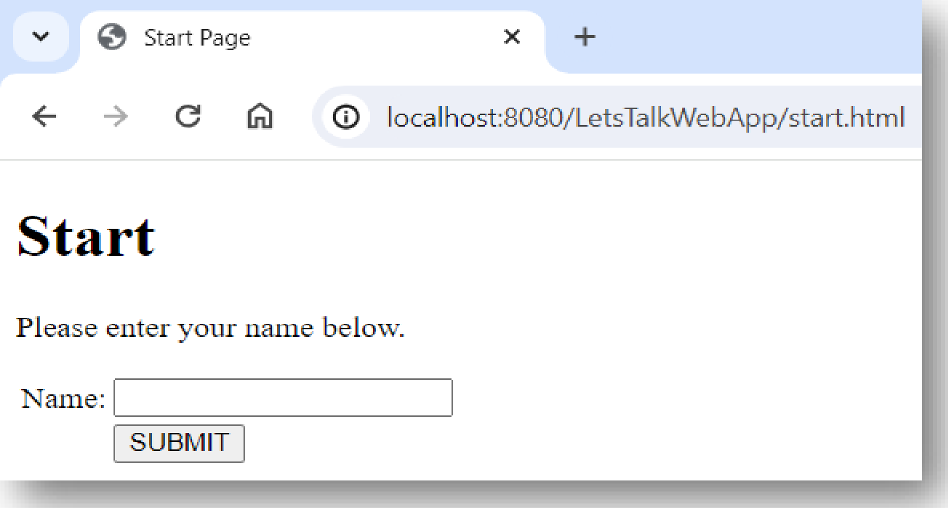
Run the program



Welcome

Welcome to our web app. Click here to start.

Click on the link



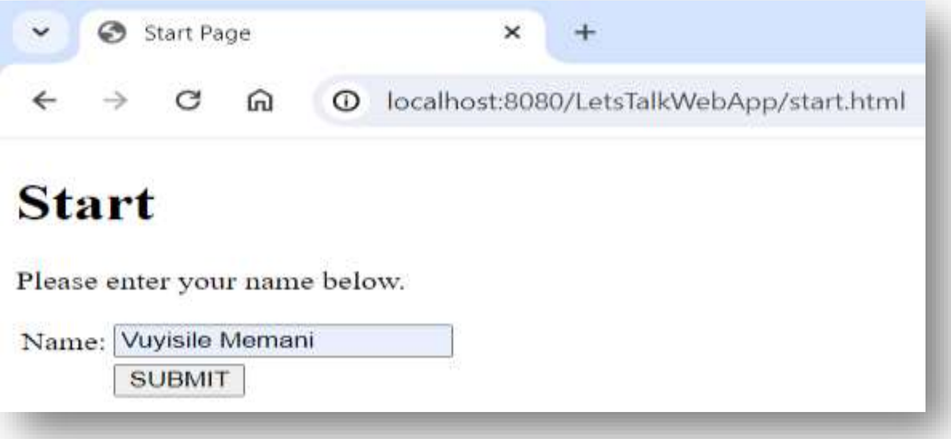
Start

Please enter your name below.

Name:

SUBMIT

Enter your name



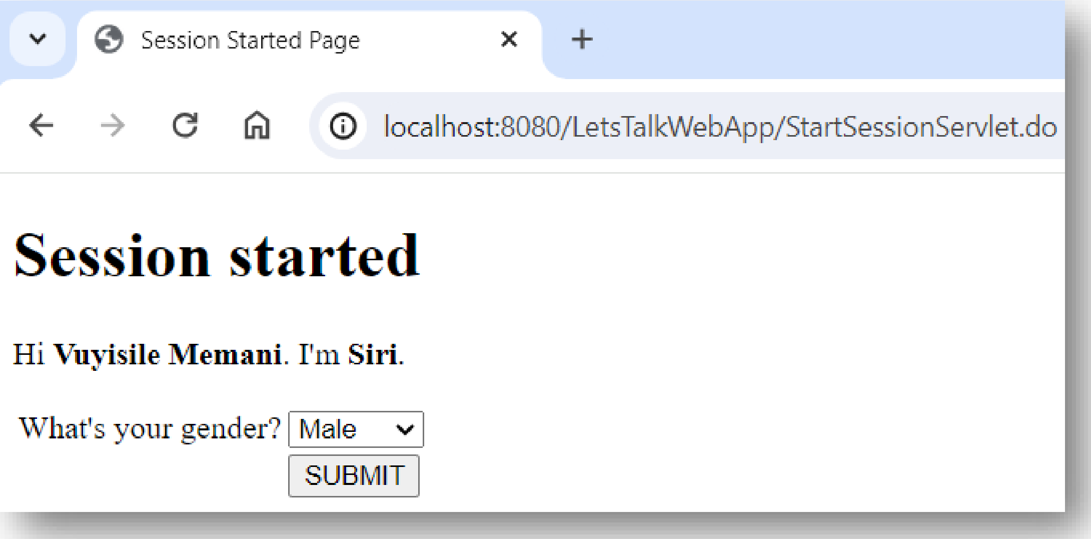
Start

Please enter your name below.

Name: Vuyisile Memani

SUBMIT

Click on the submit button

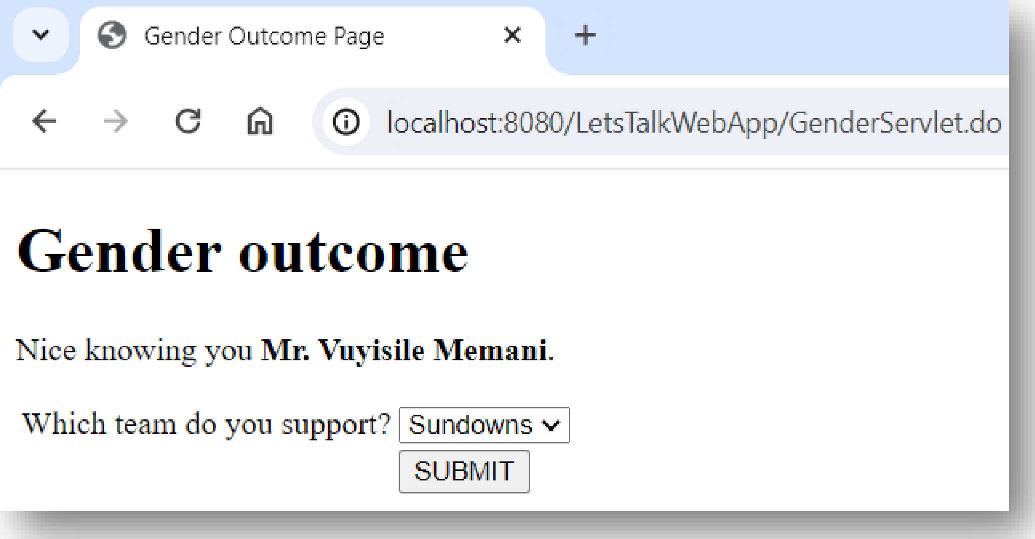


Session started

Hi **Vuyisile Memani**. I'm **Siri**.

What's your gender? Male

Click on the submit button

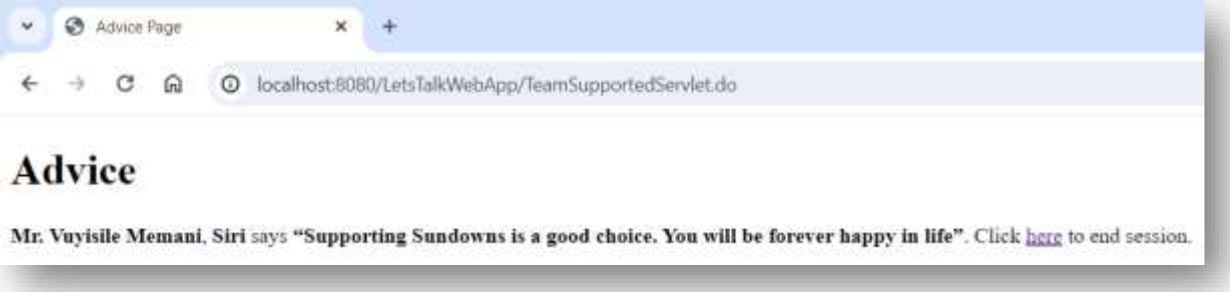


Gender outcome

Nice knowing you **Mr. Vuyisile Memani**.

Which team do you support? Sundowns

Click on the submit button



Advice

Mr. Vuyisile Memani, **Siri** says **"Supporting Sundowns is a good choice. You will be forever happy in life"**. Click here to end session.

1.3 Example 2

In this example we need create a basic web application that uses url rewriting to manage sessions.

NB: This will serve as Home work. Use example 1 and use url rewriting instead of sessions.

1.4 DIY (Do It Yourself)

In this chapter we introduced you to conversational web applications. We showed you how the two techniques work in keeping web applications conversational. 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 take an arithmetic test with five randomly generated questions. The application must first allow a user to enter his/her name. Upon receiving the name, the application must greet the user using their name and explain that there are five questions to be asked. The application must provide a link for the user to start the test. The user must select which of the four arithmetic operations he/she would like the questions to be based on.

Subsequently, the application must give the user one question at a time. After a user has answered a question, the application must determine and display the outcome. The outcome must consist of the following information:

- The question asked;
- The answer of the user;
- The correct answer; and
- The outcome, that is whether the user got the question correct or not.

After all the questions have been asked and answered, the application must display a summary report. The report must include the following information:

- The total number of questions asked;
- The number of questions that the user got correct;
- The number of questions that he user got wrong; and
- The percentage mark obtained.

Task #2

Mulumba is a programming intern at ***WeDoWebApps*** company. The company specializes in developing conversational web apps that are personalized for their clients. Mr Maluleke, a senior developer at WeDoWebApps, has been assigned the responsibility of mentoring Mulumba. As his first task, Mr Maluleke gives Mulumba an opportunity of creating a personalized web app for a client. The client is Ms Skosana, a teacher by profession.

Ms Skosana wants a personalized web application that will allow students to take a test. Ms Skosana has already prepared a test for the students. It has seven (7) multiple choice questions with corresponding answers. The table below shows the test.

No.	**Question**	**Answer**
1.	1 + 1 = ? A. 1 B. 11 C. 2 D. 0	C
2.	1 * 1 = ? A. 1 B. 11 C. 2 D. 0	A
3.	1 / 1 = ? A. 1 B. 11 C. 2 D. 0	A
4.	1 - 1 = ? A. 1 B. 11 C. 2 D. 0	D
5.	1 % 1 = ?	D

	A. 1 B. 11 C. 2 D. 0	
6.	(1 + 1) *2 = ? A. 1 B. 2 C. 4 D. 6	C
7.	(1 + 1) / 2 = ? A. 1 B. 2 C. 4 D. 6	A

Ms Skosana wants the web application to randomly select five (5) questions from the test and give them to a student for answering. The student must be given one question at a time to answer. After all the questions have been answered, the web application is required to do two things, namely:

- Mark the work of the student.
- Display a summary report. The report should include the following **information:**
- ✓ The name of the student.
- ✓ The number of questions asked.
- ✓ The number of correct answers.
- ✓ The percentage mark obtained.
- ✓ The list of questions asked.
- ✓ The correct answer for each question.
- ✓ The answers provided by the student for each question.
- ✓ The marking outcome for each student answer (“Correct” or “Wrong”).

<u>To do</u>

Assuming that you are Mulumba, create such a web application for Ms Skosana.

Task #3

Thabo has just joined a company called ***WeDoWebApps***. The company specializes in developing personalized and conversational web applications for their clients. Thabo is tasked with the responsibility of creating such a web application that will work with Fibonacci numbers in a form of a game. The game is played between a player and a computer named **Siri**. Siri gives a player a number and the player is expected to tell Siri whether the number is a **Fibonacci number** or not.

A Fibonacci number is a number that is part of the **Fibonacci series**. The Fibonacci series is formed by using three numbers, less than the number in question, which are labeled as previous, current and next. First we assume that the first two numbers of the Fibonacci series are **1** and **1**. The **first 1** is labeled as the **previous number**, and the **subsequent 1** is labeled as the **current number**. To determine the **next number** we **sum** the **previous** and **current** number. So the general formula for determining the next number is:

Next number = **previous number** + **current number**

For example, if we have the following starting series:

Fibonacci series = 1, 1

To get the next number in the series, we will have:

Next number = previous number + current number
= 1 + 1
= **2**

This will give us the following new series:

Fibonacci series = 1, 1, **2**

We can continue building the series. At this stage, 2 is the current number, and the 1 next to it is the previous number. Consequently the next number will be determined as follows:

Next number = previous number + current number
= 1 + 2
= **3**

This will give us the following new series:

Fibonacci series = 1, 1, 2, **3**

Repeating the above process, we can determine the last next number as follows:

Next number = previous number + current number
= 2 + 3
= 5

This will give us the following new series:

Fibonacci series = 1, 1, 2, 3, **5**

So, given a number, we can determine whether it is a Fibonacci number or not by checking if the number is part of the Fibonacci series. For example, if given **eleven** (**11**), we can determine whether it is a Fibonacci number or not, by doing the following:

Generate a Fibonacci series using numbers less than 11.
1,1,2,3,5,8,13

Check if 11 is part of the series.
11 is not part of the series. So 11 is not a Fibonacci number.

Let us do the last example. Say we are given 21. We will repeat the above steps. Generate a Fibonacci series using numbers less than 21.

1,1,2,3,5,8,13,21

Check if 21 is part of the series.
21 is part of the series. So 21 is a Fibonacci number.

Back to the problem statement. In this game web application, Thabo is required to create web application that will have two roles, a computer (Siri) and a player. Each of the roles must be capable of doing the following:

Computer

Siri must generate a random in the range of 2 and 100, both numbers inclusive and send it to a player to determine whether the number is a Fibonacci number or not.

Player

A player is expected to tell whether a given number is a Fibonacci number or not. After a player has provided an answer, the web application must do the following:

- Generate a Fibonacci series for the given number.
- Determine whether the number is a Fibonacci number or not.
- Determine whether the answer of the player is correct or not. The player's score must be incremented by 1 if the player got the question correct.
- Send back an outcome to the player with the following information:
  - Whether the player's answer is correct or not.
  - The generated Fibonacci series.

After all the questions have been answered by the player, the web application must do two things, namely:

Display a game summary report. The report must include the following information:

- The number of questions sent to the player.
- The number of correct answers by the player.

To do

Assuming that you are Thabo, create such a web application.

Task #4

Rendani has just joined a company called ***WeDoWebApps*** as a senior programmer. The company specializes in developing personalized and conversational web applications for their clients. Rendani is tasked with the responsibility of creating a numbers guessing game. A game is played between a player and a computer named Siri. Siri generates a number in the range of 1 to 10, both numbers inclusive. The player is expected to guess the generated number. If the guess is correct, the score of the player incremented by one (1). At all times, the player's guess must be in the range of 1 to 10.

The game between the player and Siri continues until the player decides to end it. Upon ending the game, the following must happen:

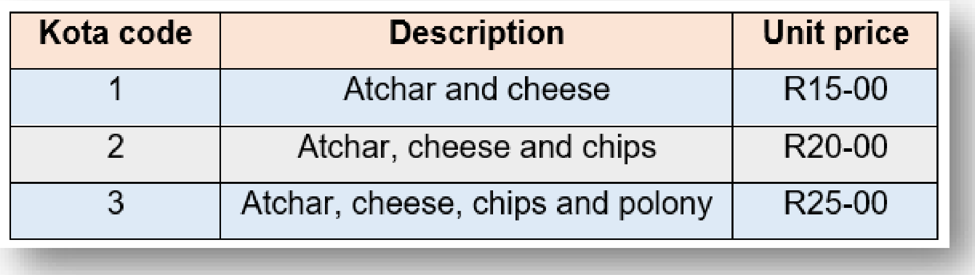
- A game summary report must be displayed. The summary must contain the following information:
  - The number of games played.
  - The number of games won by the player, meaning the number of times when the player's guess was correct.

To do

Assuming that you are Rendani, create such a web application.

Task #5

Mamazala is running a thriving business in Soshanguve, selling kotas/sphatlhos to customers. The table below shows the pricelist for the kotas.



Kota code	Description	Unit price
1	Atchar and cheese	R15-00
2	Atchar, cheese and chips	R20-00
3	Atchar, cheese, chips and polony	R25-00

She needs a web application that will help her run the business. The application must be capable of doing the following:

- Serve a customer by doing the following:
  - ✓ Display the kota menu shown in the above table.
  - ✓ Allow her to enter the kota code and quantity. The application must also allow for the user to add into the order other kotas.
  - ✓ Determine and display the amount due for the order.
  - ✓ Accept money tendered by the client and give appropriate change.
  - ✓ Update the total daily amount made.
- Serve the next customer.
- Display the following information when the shop gets closed:
  - ✓ The total number of kotas bought.
  - ✓ The number of kotas bought per code.
  - ✓ The total daily amount made.

To do

Create such a web application for Mamazala.

1.5 Conclusion

In this chapter we managed to introduce the student to conversational web applications and the techniques used.

Thank you for having taken time to go through this chapter. Enjoy the rest of the day and God bless you.