# Advance Java Assignment

- **Write a program using TCP socket in which client sends a string and server checks whether that string is palindrome or not and responds with appropriate message to the client.**

  - **Client.java:**

```
import java.io.*;
import java.net.*;
public class client {

     Socket socket;
     BufferedReader sock_in,kdb_in;
     PrintWriter sock_out;
     String str;
    public client()
    {
     try{

        Socket socket=new Socket("127.0.0.1",8765);
        kdb_in=new BufferedReader(new InputStreamReader(System.in));
        sock_in=new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        sock_out=new PrintWriter(socket.getOutputStream());
        while(true)
        {
                 System.out.println("Enter the msg");
                str=kdb_in.readLine();
                sock_out.println(str);
                sock_out.flush();
                System.out.println("Msg from Server");
                str=sock_in.readLine();
                System.out.println(str);
                if(str.equals("bye"))
                     break;
                } //while over
                socket.close();
     }
     catch (Exception e) {
          System.out.println(e);
     } //try-catch over
    } //constructor over

     public static void main(String[] args) {
```

```
		// TODO Auto-generated method stub
		new client();
	}

}
```

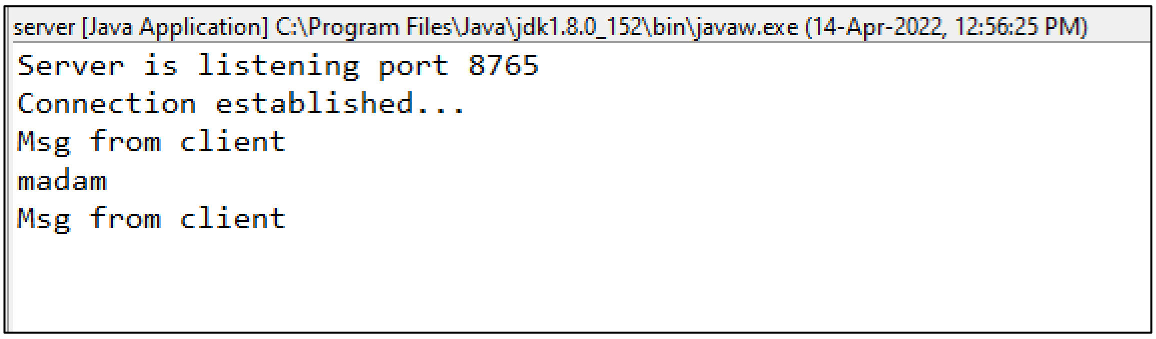
- **Server.java**

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

public class server{
	ServerSocket ss;
	Socket socket;
	BufferedReader sock_in,kdb_in;
	PrintWriter sock_out;
	String str;
	public server()
	{
	    try{
		ss=new ServerSocket(8765);
		System.out.println("Server is listening port 8765");
		socket=ss.accept();
		System.out.println("Connection established...");
		kdb_in=new BufferedReader(new InputStreamReader(System.in));
		sock_in=new BufferedReader(new
InputStreamReader(socket.getInputStream()));
		sock_out=new PrintWriter(socket.getOutputStream());
		while(true)
		{
		   System.out.println("Msg from client");
		   str=sock_in.readLine();
		   int k=str.length();
		   System.out.println(str);
		   int left=0,right=k-1;int flag=1;
		   while(left<=right)
		   {
			if(str.charAt(left)!=(str.charAt(right)))
			{
				flag=0;
				break;
			}
			else
			{
				left++;
				right--;
			}
```

```
                }
                    if(flag==1)
                        str="....Palindrome....";
                else
                        str="....Not Palindrome....";

                sock_out.println(str);
                sock_out.flush();
                if(str.equals("bye"))
                        break;
            }

        }
        catch (Exception e)
        {
         System.out.println(e);
         }
    }
public static void main(String arg[])
{
    new server();
}
}
```

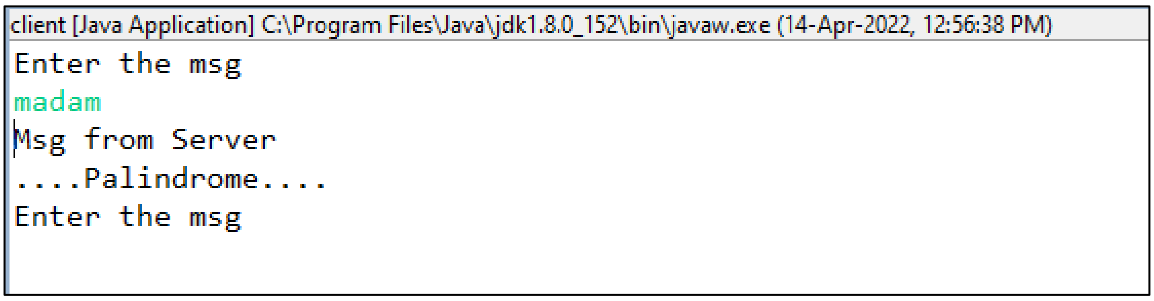
- **Output**

### Server.java



```
server [Java Application] C:\Program Files\Java\jdk1.8.0_152\bin\javaw.exe (14-Apr-2022, 12:56:25 PM)
Server is listening port 8765
Connection established...
Msg from client
madam
Msg from client
```

### Client.java



```
client [Java Application] C:\Program Files\Java\jdk1.8.0_152\bin\javaw.exe (14-Apr-2022, 12:56:38 PM)
Enter the msg
madam
Msg from Server
....Palindrome....
Enter the msg
```

## • Implement student information system using JDBC.

here I used servlet and jsp for run this project or implementation on web browser.

Here first register form appers. Then after submitting form program redirect to servlet and in servlet checking flag and register student in database after registering student user redirect to serch.jsp and that page user can see data of every student and updating and deleting as user want .

Here JDBC use to connect database to this project and mysql database .

### • Register.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Register</title>
</head>

<body>
<h2>register student</h2>
<form action="Main">

<% String s = "R";
        session.setAttribute("test",s);
%>
     enrollment = <input type="text" name="enrollment"/><br><br>
     first name = <input type="text" name="fn"/><br><br>
     last name = <input type="text" name="ln"/><br><br>
     guardian name = <input type="text" name="gn"/><br><br>
     <b>Gender :</b>
           <input type="radio" value="Male" name="gender"> Male
           <input type="radio" value="Female" name="gender"> Female
           <input type="radio" value="Other" name="gender"> Other
           <br><br>

     email-ID = <input type="text" name="email"/><br><br>
     number = <input type="text" name="number" size="10"/><br><br>
     course : <select name="course">
```

```
                <option value="Course" selected disabled>Course</option>
                <option value="B.Tech">B.Tech</option>
                <option value="MBA">MBA</option>
                <option value="M.Tech">M.Tech</option>
            </select>
            <br><br>
      branch : <select name="branch">
                <option value="Course" selected disabled>----</option>
                <option value="CE">CE</option>
                <option value="IT">IT</option>
                <option value="EC">EC</option>
                <option value="IC">IC</option>
                <option value="BM">EC</option>
            </select>
            <br><br>
       semester : <select name="semester">
                <option value="Course" selected disabled>----</option>
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
                <option value="5">5</option>
                <option value="6">6</option>
                <option value="7">7</option>
                <option value="8">8</option>
            </select>
            <br><br>
     <b>Current Address : </b><br><br>
            <textarea cols="80" rows="5" placeholder="Current Address"
name ="address" required></textarea>
            <br><br>

        <b>CPI/CGPA(last)</b><input type="text" name="SPI"/><br><br>
      <input type="submit"/>
</form>

</body>
</html>
```

- **Main.java (servlet)**

```
   <%
package basic;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class Main
 */
@WebServlet("/Main")
public class Main extends HttpServlet {
	private static final long serialVersionUID = 1L;


    public Main() {
        super();
        // TODO Auto-generated constructor stub
    }


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

		HttpSession s = request.getSession();
		PrintWriter o = response.getWriter();
		response.setContentType("text/html");

		try{


			Class.forName("com.mysql.jdbc.Driver");
	//Registration of Driver
```

```
                Connection c1 = 
DriverManager.getConnection("jdbc:mysql://localhost/aj_db","root","root")

           // Establishing connection
                Statement st = c1.createStatement();
                // Statement Creation



                String str = (String)s.getAttribute("test");

                if(str.equals("R")){

                     String enrollment = 
request.getParameter("enrollment");
                     String fn = request.getParameter("fn");
                     String ln = request.getParameter("ln");
                     String gn = request.getParameter("gn");
                     String gender = request.getParameter("gender");
                     String course = request.getParameter("course");
                     String branch = request.getParameter("branch");
                     String sem = request.getParameter("semester");
                     String result = request.getParameter("SPI");
                     String  address= request.getParameter("address");
                     String email = request.getParameter("email");

                     String number = request.getParameter("number");

                     st.executeUpdate("INSERT INTO 
student_detail(Enrollment,firstName,lastName,"
                                 +
"guardianName,gender,email,number,course,branch,semester,result,address)
"
                                 +
"VALUES('"+enrollment+"','"+fn+"','"+ln+"','"+gn+"','"+gender+"','"+email
+"'"
                                            +
"','"+number+"','"+course+"','"+branch+"','"+sem+"','"+result+"','"+addres
s+"')");
                     o.write("<br>its register section...");
                     response.sendRedirect("search.jsp");
                }


                if(str.equals("E")){

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

```
                        String fn = request.getParameter("fn");
                        String ln = request.getParameter("ln");
                        String gn = request.getParameter("gn");
                        String gender = request.getParameter("gender");
                        String course = request.getParameter("course");
                        String branch = request.getParameter("branch");
                        String sem = request.getParameter("semester");
                        String result = request.getParameter("SPI");
                        String  address= request.getParameter("address");
                        String email = request.getParameter("email");

                        String number = request.getParameter("number");

                        st.executeUpdate(

                        "UPDATE student_detail SET firstName='"+fn+"',
                  lastName='"+ln+"',"                  +
                  "guardianName='"+gn+"',
                  gender='"+gender+"'
                  ,email='"+email+"',number='"+number+"',"
                  +"course='"+course+"',branch='"+branch+"'
                  ,semester='"+sem+"',result='"+result+"',"+
                  "address='"+address+"',enrollment='"+enrollment+
                  "' WHERE id='"+id+"' ");


                        response.sendRedirect("search.jsp");
                  }
                  if(str.equals("D")){

                        String id = request.getParameter("id");
      st.executeUpdate("delete from student_detail where id="+id+"");
                        response.sendRedirect("search.jsp");
                  }


            }
            catch(Exception e)
            {
                  o.write("something wrong !!"+e);
            }


      }
```

```
     protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
          // TODO Auto-generated method stub
          doGet(request, response);
     }

}
```

- **Search.jsp**

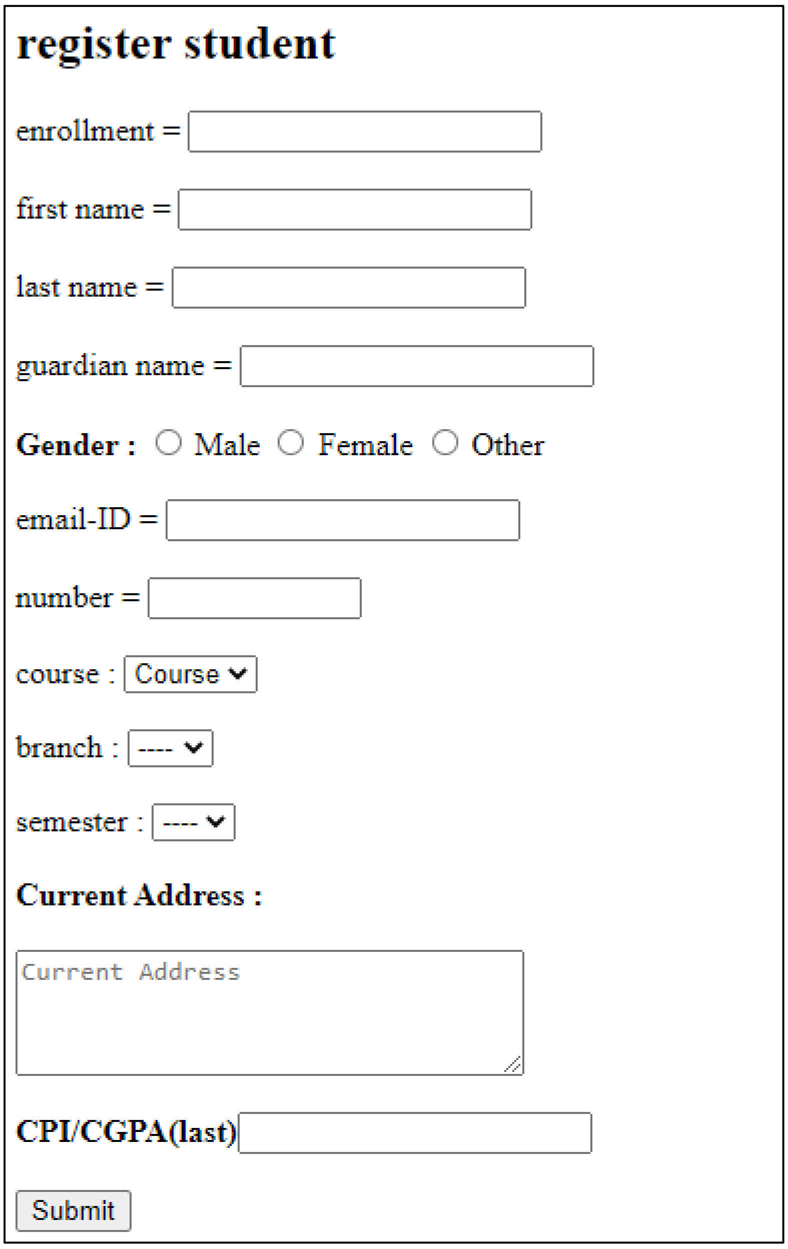
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>search</title>
</head>
<body>
<%
String s = "S";
session.setAttribute("test", s);

%>
     <%
               Class.forName("com.mysql.jdbc.Driver");
               Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/aj_db","root","root")
;
               Statement st = con.createStatement();

               ResultSet rs = st.executeQuery("select * from
student_detail ");

          %>

          <center>
          <h1>students</h1><br><br>
          <form >
          <table border="1px">
          <tr>

               <th></th>
               <th>ID</th>
               <th>enrollment</th>
               <th>first name</th>
               <th>last name</th>
```

```
                <th>guardian name</th>
                <th>gender</th>
                <th>email</th>
                <th>phone no.</th>
                <th>course</th>
                <th>branch</th>
                <th>sem</th>
                <th>result(SPI/CGPA)</th>
                <th>address</th>
                <th colspan=2 >action</th>
           </tr>
                <%
                     while(rs.next())
                     {

                     int id = rs.getInt("id");
                     String name = rs.getString("firstName");
                     String surname = rs.getString("lastName");
                     String gn = rs.getString("guardianName");
                     String gender = rs.getString("gender");
                     String email = rs.getString("email");
                     String number = rs.getString("number");
                     String course = rs.getString("course");
                     String semester = rs.getString("semester");
                     String branch = rs.getString("branch");
                     String result = rs.getString("result");
                     String address = rs.getString("address");
                     String enrollment = rs.getString("enrollment");
                 %>


           <tr>

                <td><input type="checkbox"
                     value="<%=id %>" name="check"/></td>

                <td><% out.println(id); %></td>

                <td><%out.print(enrollment); %></td>

                <td><%out.print(name); %></td>

                <td><%out.print(surname); %></td>

                <td><%out.print(gn); %></td>

                <td><%out.print(gender); %></td>

                <td><%out.print(email); %></td>
```

```
                  <td><%out.print(number); %></td>

                  <td><%out.print(course); %></td>

                  <td><%out.print(branch); %></td>

                  <td><%out.print(semester); %></td>

                  <td><%out.print(result); %></td>

                  <td><%out.print(address); %></td>

                  <td><a href="edit.jsp?id=<%=id%>">edit</a></td>

                  <td><a href="delete.jsp?id=<%=id %>">delete</a></td>
         </tr>

       <%
                  } //here while over
       %>
       </table>
       <br><br><a href="register.jsp">new student</a>
       </form>
       </center>
</body>
</html>
```

- **Output:**

## Register.jsp



**register student**

enrollment =

first name =

last name =

guardian name =

**Gender :** ◯ Male ◯ Female ◯ Other

email-ID =

number =

course : Course

branch : ----

semester : ----

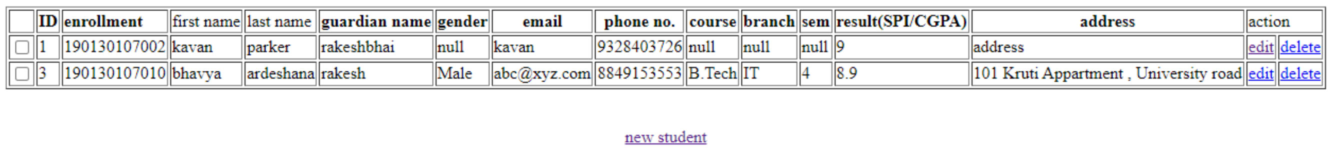
**Current Address :**

Current Address

**CPI/CGPA(last)**

Submit

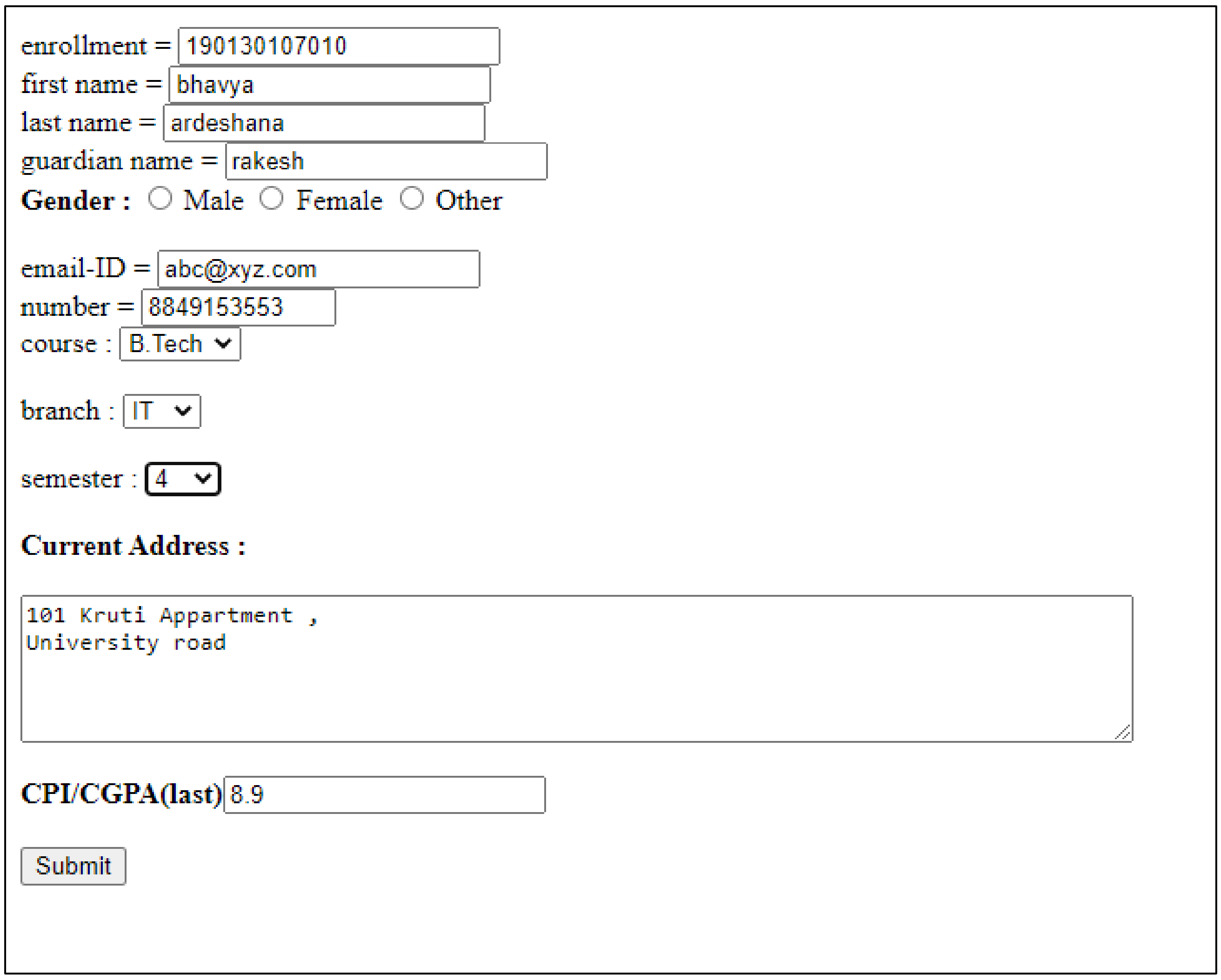
## Search.jsp



| | ID | enrollment | first name | last name | guardian name | gender | email | phone no. | course | branch | sem | result(SPI/CGPA) | address | action | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | 190130107002 | kavan | parker | rakeshbhai | null | kavan | 9328403726 | null | null | null | 9 | address | edit | delete |
| ☐ | 3 | 190130107010 | bhavya | ardeshana | rakesh | Male | abc@xyz.com | 8849153553 | B.Tech | IT | 4 | 8.9 | 101 Kruti Appartment , University road | edit | delete |

new student

## edit.jsp



enrollment = 190130107010
first name = bhavya
last name = ardeshana
guardian name = rakesh
**Gender :** ○ Male ○ Female ○ Other

email-ID = abc@xyz.com
number = 8849153553
course : B.Tech

branch : IT

semester : 4

**Current Address :**

```
101 Kruti Appartment ,
University road
```

**CPI/CGPA(last)** 8.9

Submit

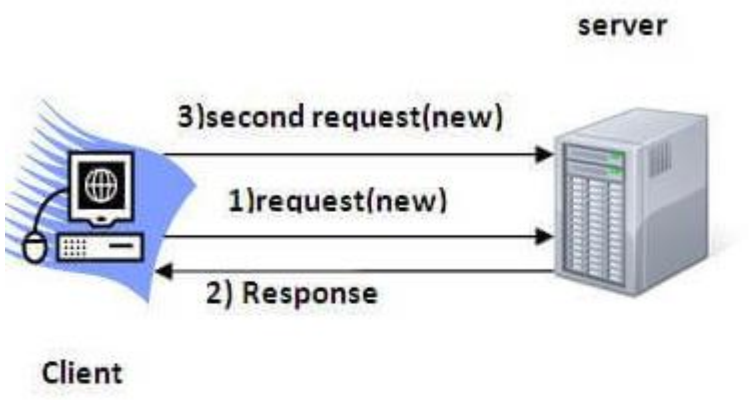
## Explain various session management methods of servlet with example.

### Session:

**The time interval in which two systems(i.e. the client and the server) communicate with each other** can be termed as a session. In simpler terms, a session is a state consisting of several requests and response between the client and the server.

It is a known fact that HTTP and Web Servers are both stateless. Hence, the only way to maintain the state of the user is by making use of technologies that implement session tracking.

Session tracking in servlets can be implemented by a number of methods, cookies being one of them.



Session management is very important in order to manage project, data and request effectively.

There are four techniques used in Session tracking/management :

- Cookies
- Hidden Form Field
- URL Rewriting
- HttpSession

## Cookies:

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.

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

A webserver can assign a unique session ID as a cookie to each web client and for subsequent requests from the client they can be recognized using the received cookie. This may not be an effective way because many time browser does not support a cookie, so I would not recommend to use this procedure to maintain the sessions.

## Hidden Form Fields :

A web server can send a hidden HTML form field along with a unique session ID as follows

```
<input type = "hidden" name = "sessionid" value = "12345">
```

This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data. Each time when web browser sends request back, then session_id value can be used to keep the track of different web browsers.

This could be an effective way of keeping track of the session but clicking on a regular (<A href...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.

## URL Rewriting :

You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.

For example, with http://xyz.com/file.html;sessionid = 12345, the session identifier is attached as sessionid = 12345 which can be accessed at the web server to identify the client.

URL rewriting is a better way to maintain sessions and it works even when browsers don't support cookies. The drawback of URL re-writing is that you would have to generate every URL dynamically to assign a session ID, even in case of a simple static HTML page.

## The HttpSession Object

Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

You would get HttpSession object by calling the public method getSession() of HttpServletRequest, as below −

HttpSession session = request.getSession();

You need to call request.getSession() before you send any document content to the client.

## Example:

### Index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="servlet1" method="post">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="go"/>
</form>
</body>
</html>
```

### FirstServlet.java

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

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/FirstServlet")
public class FirstServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

    public FirstServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

     protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
           // TODO Auto-generated method stub
           try{
           response.setContentType("text/html");
         PrintWriter out = response.getWriter();

         String n=request.getParameter("userName");
         out.print("Welcome "+n);

         Cookie ck=new Cookie("uname",n);//creating cookie object
         response.addCookie(ck);//adding cookie in the response

         //creating submit button with url rewriting
         out.print("<form action='servlet2'>");
         out.print("<input type='hidden' name='enrollment'
         value='190130107002'>");
         out.print("<input type='submit' value='go'>");
         out.print("</form>");

         out.close();

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


     protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
           // TODO Auto-generated method stub
           doGet(request, response);
     }

}
```

## SecondServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.*;

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

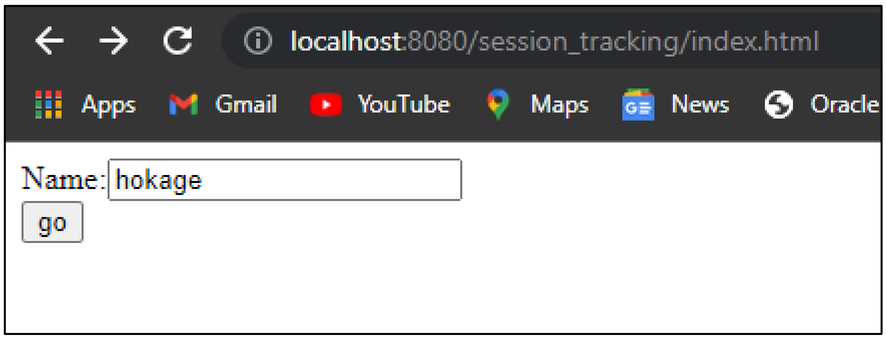
```java
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;


@WebServlet("/SecondServlet")
public class SecondServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

    public SecondServlet() {
        super();

    }



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

	      HttpSession session = request.getSession(true);
	      PrintWriter out = response.getWriter();

	      // Get session creation time.
	       Date createTime = new Date(session.getCreationTime());

	       // Get last access time of this web page.
	       Date lastAccessTime = new Date(session.getLastAccessedTime());

	       String title = "Welcome Back to my website";
	       Integer visitCount = new Integer(0);
	       String visitCountKey = new String("visitCount");
	       String userIDKey = new String("userID");
	       String userID = new String("ABCD");

	       Cookie ck[]=request.getCookies();
	       out.print("Hello "+ck[1].getValue());

	       // Check if this is new comer on your web page.
	       if (session.isNew()) {

	          title = "Welcome to my website";
	          session.setAttribute(userIDKey, userID);

	       } else {

	          visitCount = (Integer)session.getAttribute(visitCountKey);
	          visitCount = visitCount + 1;
```

```
                userID = (String)session.getAttribute(userIDKey);
             }

             session.setAttribute(visitCountKey,  visitCount);

             String enrollment = request.getParameter("enrollment");
             // Set response content type
             response.setContentType("text/html");
//           PrintWriter out = response.getWriter();

             String docType =
                "<!doctype html public \"-//w3c//dtd html 4.0 " +
                "transitional//en\">\n";

             out.println(docType +
                "<html>\n" +
                   "<head><title>" + title + "</title></head>\n" +

                   "<body bgcolor =\"#f0f0f0\">\n" +
                      "<h1 align =\"center\">" + title + "</h1>\n" +
                      "<h2 align =\"center\">Hello
                       "+ck[1].getValue()+"</h2>\n"+
                      "<h2 align =\"center\">enrollment
                       "+enrollment+"</h2>\n"+
                      "<h3 align =\"center\">Session Infomation</h3>\n" +
                      "<table border =\"1\" align = \"center\">\n" +

                         "<tr bgcolor =\"#949494\">\n" +
                            "  <th>Sessioninfo</th><th>value</th></tr>\n"+
                         "<tr>\n" +
                            "  <td>id</td>\n" +
                            "  <td>" + session.getId() + "</td>"+
                         "</tr>\n" +

                         "<tr>\n" +
                            "  <td>Creation Time</td>\n" +
                            "  <td>" + createTime + "  </td>"+
                         "</tr>\n" +

                         "<tr>\n" +
                            "  <td>Time of Last Access</td>\n" +
                            "  <td>" + lastAccessTime + "  </td>"+
                         "</tr>\n" +

                         "<tr>\n" +
                            "  <td>User ID</td>\n" +
                            "  <td>" + userID + "  </td>"+
                         "</tr>\n" +
                                "<tr>\n" +
                                   "  <td>Number of visits</td>\n" +
                                   "  <td>" + visitCount + "</td>"+
                                "</tr>\n" +
                             "</table>\n" +
                          "</body>"+
                       "</html>");
      }
      protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

           doGet(request, response);
      }

}
```

Compile the above servlet **SessionTrack** and create appropriate entry in web.xml file. Now running *http://localhost:8080/SessionTrack* would display the following result when you would run for the first time

In this practical we use all session management methods like URL rewriting , http session and cookies management .

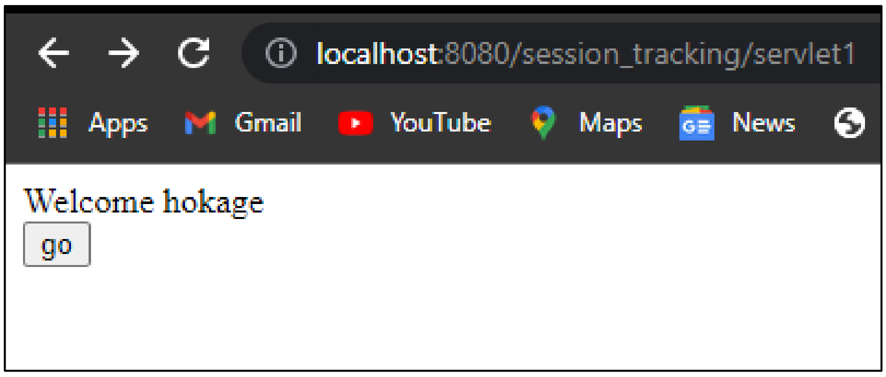
## Output :



localhost:8080/session_tracking/index.html

Name: hokage

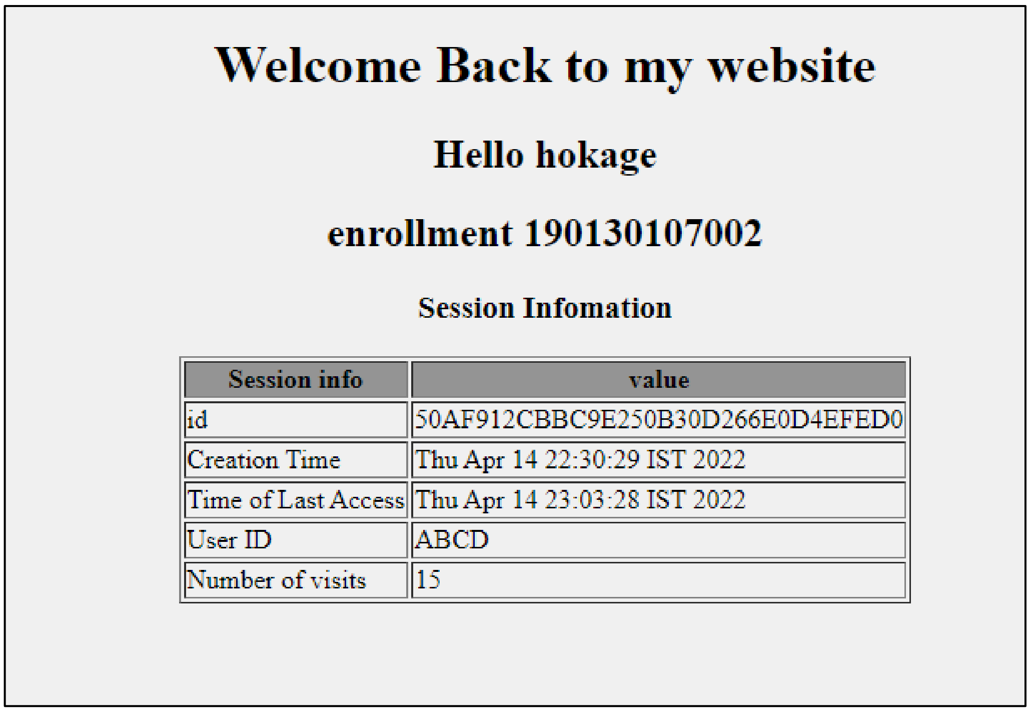
go



localhost:8080/session_tracking/servlet1

Welcome hokage

go



# Welcome Back to my website

## Hello hokage

## enrollment 190130107002

### Session Infomation

| Session info | value |
|---|---|
| id | 50AF912CBBC9E250B30D266E0D4EFED0 |
| Creation Time | Thu Apr 14 22:30:29 IST 2022 |
| Time of Last Access | Thu Apr 14 23:03:28 IST 2022 |
| User ID | ABCD |
| Number of visits | 15 |

## Explain JSP tag library

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

The JSP tag library is collection of useful jsp tag where encapulates to core functionality common to many JSP application.

JSTL has supports for common structures tasks such as iteration and conditions tags for manipulating XML documents , internationalization tags and sql tags . it also provides a framework for interpreting the existing custom tag with JSTL .

**Advantage of JSTL**

- **Fast Development** JSTL provides many tags that simplify the JSP.
- **Code Reusability** We can use the JSTL tags on various pages.
- **No need to use scriptlet tag** It avoids the use of scriptlet tag.

**JSTL Tags**

There JSTL mainly provides five types of tags:

| Tag Name | Description |
|---|---|
| Core tags | The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/core**. The prefix of core tag is **c**. |
| Function tags | The functions tags provide support for string manipulation and string length. The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**. |
| Formatting tags | The Formatting tags provide support for message formatting, number and date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**. |

| | |
|---|---|
| XML tags | The XML tags provide flow control, transformation, etc. The URL for the XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**. |
| SQL tags | The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**. |

## List of core JSTL tags:

The JSTL core tag provides variable support, URL management, flow control etc. The syntax used for including JSTL core library in your JSP is:

| Tags | Description |
|---|---|
| c:out | It display the result of an expression, similar to the way <%=...%> tag work. |
| c:import | It Retrives relative or an absolute URL and display the contents to either a String in 'var',a Reader in 'varReader' or the page. |
| c:set | It sets the result of an expression under evaluation in a 'scope' variable. |
| c:remove | It is used for removing the specified scoped variable from a particular scope. |
| c:catch | It is used for Catches any Throwable exceptions that occurs in the body. |
| c:if | It is conditional tag used for testing the condition and display the body content only if the expression evaluates is true. |

| | |
|---|---|
| c:choose, c:when, c:otherwise | It is the simple conditional tag that includes its body content if the evaluated condition is true. |
| c:forEach | It is the basic iteration tag. It repeats the nested body content for fixed number of times or over collection. |
| c:forTokens | It iterates over tokens which is separated by the supplied delimeters. |
| c:param | It adds a parameter in a containing 'import' tag's URL. |
| c:redirect | It redirects the browser to a new URL and supports the context-relative URLs. |
| c:url | It creates a URL with optional query parameters. |

## Formatting jstl tags :

The jstl tag are used to format and display text , date , time , character , numbers on web page or jsp page .

The formatting tags provide support for message formatting, number and date formatting etc. The url for the formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt.**

The JSTL formatting tags are used for internationalized web sites to display and format text, the time, the date and numbers. The syntax used for including JSTL formatting library in your JSP is:

| Formatting Tags | Descriptions |
|---|---|
| fmt:parseNumber | It is used to Parses the string representation of a currency, percentage or number. |
| fmt:timeZone | It specifies a parsing action nested in its body or the time zone for any time formatting. |
| fmt:formatNumber | It is used to format the numerical value with specific format or precision. |
| fmt:parseDate | It parses the string representation of a time and date. |
| fmt:setTimeZone | It stores the time zone inside a time zone configuration variable. |
| fmt:message | It display an internationalized message. |
| fmt:formatDate | It formats the time and/or date using the supplied pattern and styles. |

## JSTL SQL Tags

The sql tags provide SQL support. The the sql tags prefix is **sql** and url for **http://java.sun.com/jsp/jstl/sql**

The SQL tag library allows the tag to interact with RDBMSs (Relational Databases) such as Microsoft SQL Server, mySQL, or Oracle. The syntax used for including JSTL SQL tags library in your JSP is:

| SQL Tags | Descriptions |
|---|---|
| sql:setDataSource | It is used for creating a simple data source suitable only for prototyping. |
| sql:query | It is used for executing the SQL query defined in its sql attribute or the body. |
| sql:update | It is used for executing the SQL update defined in its sql attribute or in the tag body. |
| sql:param | It is used for sets the parameter in an SQL statement to the specified value. |
| sql:dateParam | It is used for sets the parameter in an SQL statement to a specified java.util.Date value. |
| sql:transaction | It is used to provide the nested database action with a common connection. |

## JSTL XML tags

The JSTL XML tags are used for providing a JSP-centric way of manipulating and creating XML documents.

The xml tags provide flow control, transformation etc. The url for the xml tags is **http://java.sun.com/jsp/jstl/xml** and prefix is x. The JSTL XML tag library has custom tags used for interacting with XML data.

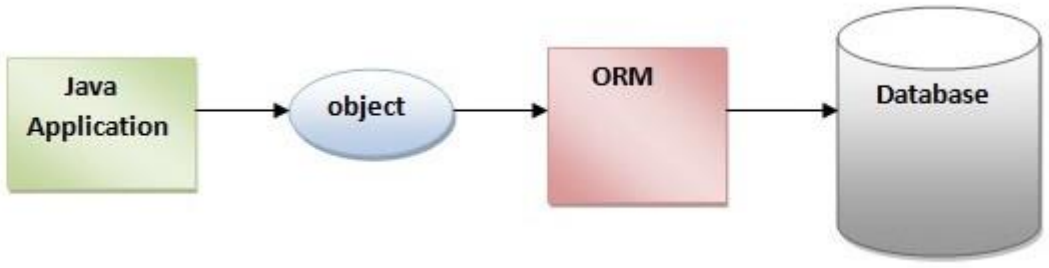
| XML Tags | Descriptions |
|---|---|
| x:out | Similar to <%= ... > tag, but for XPath expressions. |
| x:parse | It is used for parse the XML data specified either in the tag body or an attribute. |
| x:set | It is used to sets a variable to the value of an XPath expression. |
| x:choose | It is a conditional tag that establish a context for mutually exclusive conditional operations. |
| x:when | It is a subtag of that will include its body if the condition evaluated be 'true'. |
| x:otherwise | It is subtag of that follows tags and runs only if all the prior conditions evaluated be 'false'. |
| x:if | It is used for evaluating the test XPath expression and if it is true, it will processes its body content. |
| x:transform | It is used in a XML document for providing the XSL(Extensible Stylesheet Language) transformation. |
| x:param | It is used along with the transform tag for setting the parameter in the XSLT style sheet. |

## Explain O/R Mapping and Hibernate architecture

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

ORM Tool

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.

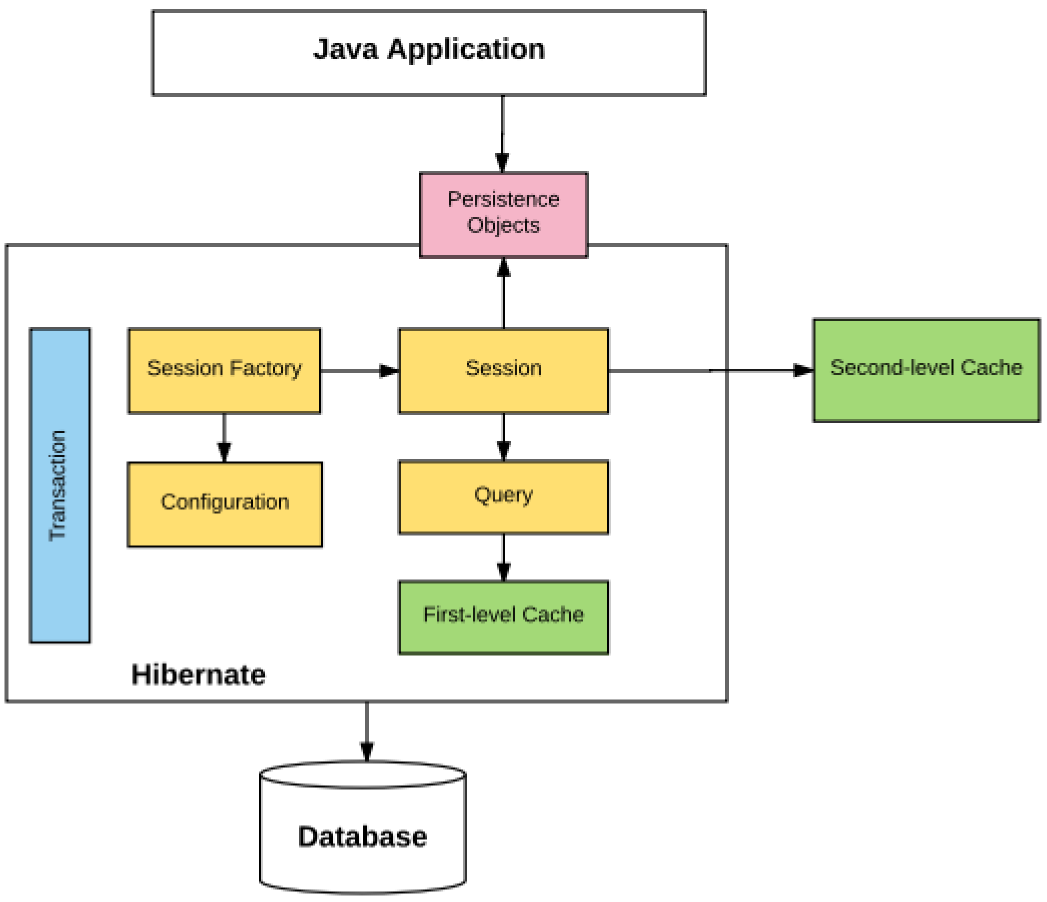


The ORM tool internally uses the JDBC API to interact with the database.

Initially started as an ORM framework, Hibernate has spun off into many projects, such as *Hibernate Search*, *Hibernate Validator*, *Hibernate OGM* (for NoSQL databases), and so on.

**Hibernate Architecture**

The following diagram summarizes the main building blocks in hibernate architecture.



Hibernate Architecture

Let’s understand what each block represents.

1. **Configuration** : Generally written in hibernate.properties or hibernate.cfg.xml files. For Java configuration, you may find class annotated with @Configuration. It is used by Session Factory to work with Java Application and the Database. It represents an entire set of mappings of an application Java Types to an SQL database.
2. **Session Factory** : Any user application requests Session Factory for a session object. Session Factory uses configuration information from above listed files, to instantiates the session object appropriately.
3. **Session** : This represents the interaction between the application and the database at any point of time. This is represented by the org.hibernate.Session class. The instance of a session can be retrieved from the SessionFactory bean.

4. **Query** : It allows applications to query the database for one or more stored objects. Hibernate provides different techniques to query database, including NamedQuery and Criteria API.
5. **First-level cache** : It represents the default cache used by Hibernate Session object while interacting with the database. It is also called as session cache and caches objects within the current session. All requests from the Session object to the database must pass through the first-level cache or session cache. One must note that the first-level cache is available with the session object until the Session object is live.
6. **Transaction** : enables you to achieve data consistency, and rollback incase something goes unexpected.
7. **Persistent objects** : These are plain old Java objects (POJOs), which get persisted as one of the rows in the related table in the database by hibernate.They can be configured in configurations files (hibernate.cfg.xml or hibernate.properties) or annotated with @Entity annotation.
8. **Second-level cache** : It is used to store objects across sessions. This needs to be explicitly enabled and one would be required to provide the cache provider for a second-level cache. One of the common second-level cache providers is *EhCache*.

## Advantages of Hibernate Framework

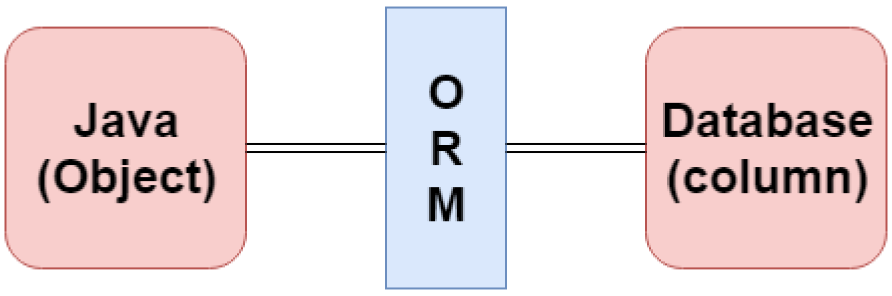
- Hibernate framework is open source under the LGPL license and lightweight.
- The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.
- HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

- Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.
- Fetching data from multiple tables is easy in hibernate framework.
- Hibernate supports Query cache and provide statistics about query and database status.

## Object/Relational Mapping

Hibernate, as an ORM framework, allows the mapping of the Java domain object with database tables and vice versa. As a result, business logic is able to access and manipulate database entities via Java objects. It helps to speed up the overall development process by taking care of aspects such as transaction management, automatic primary key generation, managing database connections and related implementations, and so on.

Object Relational Mapping (ORM) is a functionality which is used to develop and maintain a relationship between an object and relational database by mapping an object state to database column. It is capable to handle various database operations easily such as inserting, updating, deleting etc.



Mapping Directions

Mapping Directions are divided into two parts: -

- **Unidirectional relationship -** In this relationship, only one entity can refer the properties to another. It contains only one owing side that specifies how an update can be made in the database.
- **Bidirectional relationship -** This relationship contains an owning side as well as an inverse side. So here every entity has a relationship field or refer the property to other entity.

Types of Mapping

Following are the various ORM mappings: -

- **One-to-one -** This association is represented by @OneToOne annotation. Here, instance of each entity is related to a single instance of another entity.
- **One-to-many -** This association is represented by @OneToMany annotation. In this relationship, an instance of one entity can be related to more than one instance of another entity.
- **Many-to-one -** This mapping is defined by @ManyToOne annotation. In this relationship, multiple instances of an entity can be related to single instance of another entity.
- **Many-to-many -** This association is represented by @ManyToMany annotation. Here, multiple instances of an entity can be related to multiple instances of another entity. In this mapping, any side can be the owing side.