Q.1 **Cookies and Sessions** are used to store information. Cookies are only stored on the client-side machine, while sessions get stored on the client as well as a server.

Session

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

A session ends when the user closes the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. Server script sends a set of cookies to the browser. For example name, age, or identification number etc. The browser stores this information on a local machine for future use.

2 string connectionUrl = "jdbc:sqlserver://(LocalDB)\\v11.0;user:xxx;password:xxx";

Connection con = null;

Statement stmt = null;

ResultSet rs = null;

try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

con = DriverManager.getConnection(connectionUrl);

String SQL = "SELECT TOP 10 \* FROM table";

stmt = con.createStatement();

rs = stmt.executeQuery(SQL);

while (rs.next()) {

System.out.println(rs.getString(4) + " " + rs.getString(6));

}

}

catch (Exception e) {

e.printStackTrace();

}

3. **Statement interface**

The Statement interface provides methods to execute queries with the database. The statement interface is a factory of ResultSet i.e. it provides factory method to get the object of ResultSet.

Commonly used methods of Statement interface:

The important methods of Statement interface are as follows:

|  |
| --- |
| 1) public ResultSet executeQuery(String sql): is used to execute SELECT query. It returns the object of ResultSet. |
| 2) public int executeUpdate(String sql): is used to execute specified query, it may be create, drop, insert, update, delete etc. |
| 3) public boolean execute(String sql): is used to execute queries that may return multiple results. |
| 4) public int[] executeBatch(): is used to execute batch of commands.  4. **Connect using properties on the**  // URL and specifying a user ID and password  Connection c = DriverManager.getConnection(  "jdbc:as400://mySystem;naming=sql;errors=full",  "auser",  "apassword");  5. **welcome-file-list in web.xml**  The welcome-file-list element of web-app, is used to define a list of welcome files. Its sub element is welcome-file that is used to define the welcome file.  A welcome file is the file that is invoked automatically by the server, if you don't specify any file name.  By default server looks for the welcome file in following order:  welcome-file-list in web.xml  index.html  index.htm  index.jsp  If none of these files are found, server renders 404 error.  If you have specified welcome-file in web.xml, and all the files index.html, index.htm and index.jsp exists, priority goes to welcome-file.  **Q.2 A**  [**HttpSession interface**](https://www.javatpoint.com/http-session-in-session-tracking)  [How to get the HttpSession object](https://www.javatpoint.com/http-session-in-session-tracking#httpsessionhow)  [Commonly used methods of HttpSession interface](https://www.javatpoint.com/http-session-in-session-tracking#httpsessionmethod)  [Example of using HttpSession](https://www.javatpoint.com/http-session-in-session-tracking#httpsessionex)  In such case, container creates a session id for each user.The container uses this id to identify the particular user.An object of HttpSession can be used to perform two tasks:  bind objects  view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.  How to get the HttpSession object ?  The HttpServletRequest interface provides two methods to get the object of HttpSession:  public HttpSession getSession():Returns the current session associated with this request, or if the request does not have a session, creates one.  public HttpSession getSession(boolean create):Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.  Commonly used methods of HttpSession interface  public String getId():Returns a string containing the unique identifier value.  public long getCreationTime():Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.  public long getLastAccessedTime():Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.  public void invalidate():Invalidates this session then unbinds any objects bound to it.  In this example, we are setting the attribute in the session scope in one servlet and getting that value from the session scope in another servlet. To set the attribute in the session scope, we have used the setAttribute() method of HttpSession interface and to get the attribute, we have used the getAttribute method.  index.html  <form action="servlet1">  Name:<input type="text" name="userName"/><br/>  <input type="submit" value="go"/>  </form>  FirstServlet.java  import java.io.\*;  import javax.servlet.\*;  import javax.servlet.http.\*;      public class FirstServlet extends HttpServlet {    public void doGet(HttpServletRequest request, HttpServletResponse response){          try{            response.setContentType("text/html");          PrintWriter out = response.getWriter();            String n=request.getParameter("userName");          out.print("Welcome "+n);            HttpSession session=request.getSession();          session.setAttribute("uname",n);            out.print("<a href='servlet2'>visit</a>");            out.close();                    }catch(Exception e){System.out.println(e);}      }    } |

B. **Java Database Connectivity with 5 Steps**

[5 Steps to connect to the database in java](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java)

[Register the driver class](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step1)

[Create the connection object](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step2)

[Create the Statement object](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step3)

[Execute the query](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step4)

[Close the connection object](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step5)

|  |
| --- |
| There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:  Register the Driver class  Create connection  Create statement  Execute queries  Close connection |

1) Register the driver class

|  |
| --- |
| The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class. |

Syntax of forName() method

public static void forName(String className)throws ClassNotFoundException

Note: Since JDBC 4.0, explicitly registering the driver is optional. We just need to put vender's Jar in the classpath, and then JDBC driver manager can detect and load the driver automatically.

Example to register the OracleDriver class

Here, Java program is loading oracle driver to esteblish database connection.

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

2) Create the connection object

|  |
| --- |
| The getConnection() method of DriverManager class is used to establish connection with the database. |

Syntax of getConnection() method

1) public static Connection getConnection(String url)throws SQLException

2) public static Connection getConnection(String url,String name,String password)

throws SQLException

Example to establish connection with the Oracle database

Connection con=DriverManager.getConnection(

"jdbc:oracle:thin:@localhost:1521:xe","system","password");

3) Create the Statement object

|  |
| --- |
| The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database. |

Syntax of createStatement() method

public Statement createStatement()throws SQLException

Example to create the statement object

Statement stmt=con.createStatement();

4) Execute the query

|  |
| --- |
| The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table. |

Syntax of executeQuery() method

public ResultSet executeQuery(String sql)throws SQLException

Example to execute query

ResultSet rs=stmt.executeQuery("select \* from emp");

while(rs.next()){

System.out.println(rs.getInt(1)+" "+rs.getString(2));

}

5) Close the connection object

|  |
| --- |
| By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

Syntax of close() method

public void close()throws SQLException

Example to close connection

con.close();

**Q.3 A**

**Life Cycle of a Servlet (Servlet Life Cycle)**

[Servlet class is loaded](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle1)

[Servlet instance is created](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle2)

[init method is invoked](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle3)

[service method is invoked](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle4)

[destroy method is invoked](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle5)

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

As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

1) Servlet class is loaded

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

2) Servlet instance is created

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

3) init method is invoked

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

public void init(ServletConfig config) throws ServletException

4) service method is invoked

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

public void service(ServletRequest request, ServletResponse response)

  throws ServletException, IOException

5) destroy method is invoked

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

public void destroy()

**B Cookies in Servlet**

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

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

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add 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.

Types of Cookie

There are 2 types of cookies in servlets.

Non-persistent cookie

Persistent cookie

Non-persistent cookie

It is valid for single session only. It is removed each time when user closes the browser.

Persistent cookie

It is valid for multiple session . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

Simplest technique of maintaining the state.

Cookies are maintained at client side.

Disadvantage of Cookies

It will not work if cookie is disabled from the browser.

Only textual information can be set in Cookie object.

Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

Cookie class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of Cookie class

|  |  |
| --- | --- |
| Constructor | Description |
| Cookie() | constructs a cookie. |
| Cookie(String name, String value) | constructs a cookie with a specified name and value. |

Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

|  |  |
| --- | --- |
| Method | Description |
| public void setMaxAge(int expiry) | Sets the maximum age of the cookie in seconds. |
| public String getName() | Returns the name of the cookie. The name cannot be changed after creation. |
| public String getValue() | Returns the value of the cookie. |
| public void setName(String name) | changes the name of the cookie. |
| public void setValue(String value) | changes the value of the cookie. |

Other methods required for using Cookies

|  |
| --- |
| For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:  public void addCookie(Cookie ck):method of HttpServletResponse interface is used to add cookie in response object.  public Cookie[] getCookies():method of HttpServletRequest interface is used to return all the cookies from the browser. |

How to create Cookie?

Let's see the simple code to create cookie.

Cookie ck=new Cookie("user","sonoo jaiswal");//creating cookie object

response.addCookie(ck);//adding cookie in the response

How to delete Cookie?

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

Cookie ck=new Cookie("user","");//deleting value of cookie

ck.setMaxAge(0);//changing the maximum age to 0 seconds

response.addCookie(ck);//adding cookie in the response

How to get Cookies?

Let's see the simple code to get all the cookies.

Cookie ck[]=request.getCookies();

for(int i=0;i<ck.length;i++){

 out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie

}

Simple example of Servlet Cookies

In this example, we are storing the name of the user in the cookie object and accessing it in another servlet. As we know well that session corresponds to the particular user. So if you access it from too many browsers with different values, you will get the different value.

index.html

<form action="servlet1" method="post">

Name:<input type="text" name="userName"/><br/>

<input type="submit" value="go"/>

</form>

FirstServlet.java

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class FirstServlet extends HttpServlet {

  public void doPost(HttpServletRequest request, HttpServletResponse response){

    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

    out.print("<form action='servlet2'>");

    out.print("<input type='submit' value='go'>");

    out.print("</form>");

    out.close();

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

  }

}

SecondServlet.java

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class SecondServlet extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response){

    try{

    response.setContentType("text/html");

    PrintWriter out = response.getWriter();

    Cookie ck[]=request.getCookies();

    out.print("Hello "+ck[0].getValue());

    out.close();

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

    }

}

web.xml

<web-app>

<servlet>

<servlet-name>s1</servlet-name>

<servlet-class>FirstServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>s1</servlet-name>

<url-pattern>/servlet1</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>s2</servlet-name>

<servlet-class>SecondServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>s2</servlet-name>

<url-pattern>/servlet2</url-pattern>

</servlet-mapping>

</web-app>

**Q.4**

**A Servlet HttpSession Login and Logout Example**

We can bind the objects on HttpSession instance and get the objects by using setAttribute and getAttribute methods.

In the previous page, we have learnt about what is HttpSession, How to store and get data from session object etc.

Here, we are going to create a real world login and logout application without using database code. We are assuming that password is admin123.

Visit here for login and logout application using cookies only [servlet login and logout example using cookies](https://www.javatpoint.com/servlet-login-and-logout-example-using-cookies)

In this example, we are creating 3 links: login, logout and profile. User can't go to profile page until he/she is logged in. If user is logged out, he need to login again to visit profile.

In this application, we have created following files.

index.html

link.html

login.html

LoginServlet.java

LogoutServlet.java

ProfileServlet.java

web.xml

 File: index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Servlet Login Example</title>

</head>

<body>

<h1>Login App using HttpSession</h1>

<a href="login.html">Login</a>|

<a href="LogoutServlet">Logout</a>|

<a href="ProfileServlet">Profile</a>

</body>

</html>

File: link.html

<a href="login.html">Login</a> |

<a href="LogoutServlet">Logout</a> |

<a href="ProfileServlet">Profile</a>

<hr>

File: login.html

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

Name:<input type="text" name="name"><br>

Password:<input type="password" name="password"><br>

<input type="submit" value="login">

</form>

File: LoginServlet.java

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class LoginServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

                    throws ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out=response.getWriter();

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

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

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

        if(password.equals("admin123")){

        out.print("Welcome, "+name);

        HttpSession session=request.getSession();

        session.setAttribute("name",name);

        }

        else{

            out.print("Sorry, username or password error!");

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

        }

        out.close();

    }

}

File: LogoutServlet.java

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class LogoutServlet extends HttpServlet {

        protected void doGet(HttpServletRequest request, HttpServletResponse response)

                                throws ServletException, IOException {

            response.setContentType("text/html");

            PrintWriter out=response.getWriter();

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

            HttpSession session=request.getSession();

            session.invalidate();

            out.print("You are successfully logged out!");

            out.close();

    }

}

File: ProfileServlet.java

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class ProfileServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)

                      throws ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out=response.getWriter();

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

        HttpSession session=request.getSession(false);

        if(session!=null){

        String name=(String)session.getAttribute("name");

        out.print("Hello, "+name+" Welcome to Profile");

        }

        else{

            out.print("Please login first");

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

        }

        out.close();

    }

}

File: web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" id="WebApp\_ID" version="2.5">

  <servlet>

    <description></description>

    <display-name>LoginServlet</display-name>

    <servlet-name>LoginServlet</servlet-name>

    <servlet-class>LoginServlet</servlet-class>

  </servlet>

  <servlet-mapping>

    <servlet-name>LoginServlet</servlet-name>

    <url-pattern>/LoginServlet</url-pattern>

  </servlet-mapping>

  <servlet>

    <description></description>

    <display-name>ProfileServlet</display-name>

    <servlet-name>ProfileServlet</servlet-name>

    <servlet-class>ProfileServlet</servlet-class>

  </servlet>

  <servlet-mapping>

    <servlet-name>ProfileServlet</servlet-name>

    <url-pattern>/ProfileServlet</url-pattern>

  </servlet-mapping>

  <servlet>

    <description></description>

    <display-name>LogoutServlet</display-name>

    <servlet-name>LogoutServlet</servlet-name>

    <servlet-class>LogoutServlet</servlet-class>

  </servlet>

  <servlet-mapping>

    <servlet-name>LogoutServlet</servlet-name>

    <url-pattern>/LogoutServlet</url-pattern>

  </servlet-mapping>

</web-app>

**Q.4 index.html**

<form action="servlet1" method="post">

Name:<input type="text" name="userName"/><br/>

Password:<input type="password" name="userPass"/><br/>

<input type="submit" value="login"/>

</form>

Login.java

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class Login extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

    response.setContentType("text/html");

    PrintWriter out = response.getWriter();

    String n=request.getParameter("userName");

    String p=request.getParameter("userPass");

    if(p.equals("servlet"){

        RequestDispatcher rd=request.getRequestDispatcher("servlet2");

        rd.forward(request, response);

    }

    else{

        out.print("Sorry UserName or Password Error!");

        RequestDispatcher rd=request.getRequestDispatcher("/index.html");

        rd.include(request, response);

        }

    }

}

WelcomeServlet.java

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class WelcomeServlet extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

    response.setContentType("text/html");

    PrintWriter out = response.getWriter();

    String n=request.getParameter("userName");

    out.print("Welcome "+n);

    }

}

web.xml

<web-app>

 <servlet>

    <servlet-name>Login</servlet-name>

    <servlet-class>Login</servlet-class>

  </servlet>

  <servlet>

    <servlet-name>WelcomeServlet</servlet-name>

    <servlet-class>WelcomeServlet</servlet-class>

  </servlet>

  <servlet-mapping>

    <servlet-name>Login</servlet-name>

    <url-pattern>/servlet1</url-pattern>

  </servlet-mapping>

  <servlet-mapping>

    <servlet-name>WelcomeServlet</servlet-name>

    <url-pattern>/servlet2</url-pattern>

  </servlet-mapping>

  <welcome-file-list>

   <welcome-file>index.html</welcome-file>

  </welcome-file-list>

</web-app>