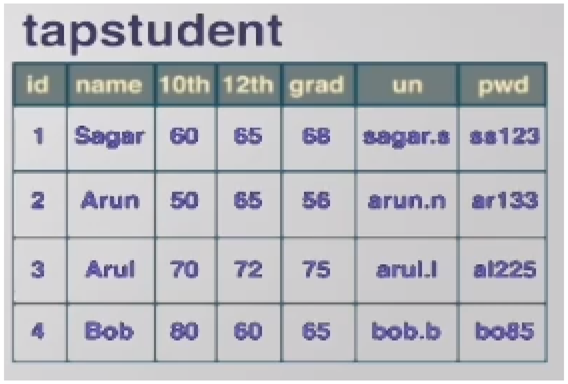
Day - 6
Valid and Invalid response of a servlet

In this session let us try to understand the different capabilities of the Servlet through an example.

Let us create a table in our database called as **“tapstudent”** with the following details-



tapstudent

id	name	10th	12th	grad	un	pwd
1	Sagar	60	65	68	sagar.s	ss123
2	Arun	50	65	56	arun.n	ar133
3	Arul	70	72	75	arul.l	al225
4	Bob	80	60	65	bob.b	bo85

Now what our particular program has to do is- When the user enters the URL then the login page should be displayed and if the user provides a valid username and password, then the servlet has to now connect to the database and the following message.

Welcome Sagar!

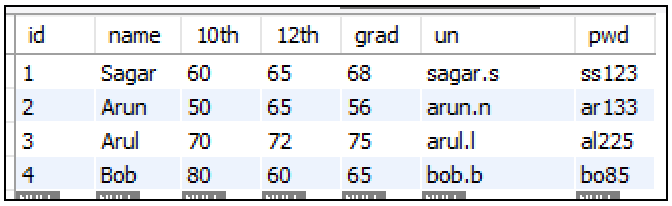
So how do we achieve this?

We will first create a servlet called Exmp.java and it must accept the request from the client and the data which was sent from the client will be available in the Request object and now the servlet must connect to the database and check if a row with the username & password present in the object is present in the database or not.

If the login details are valid then a message should be displayed. Similarly if the details are not valid then the appropriate message should be displayed.

Let us implement this in our code-

1. First create a table **‘tapstudent’** in MySQL workbench

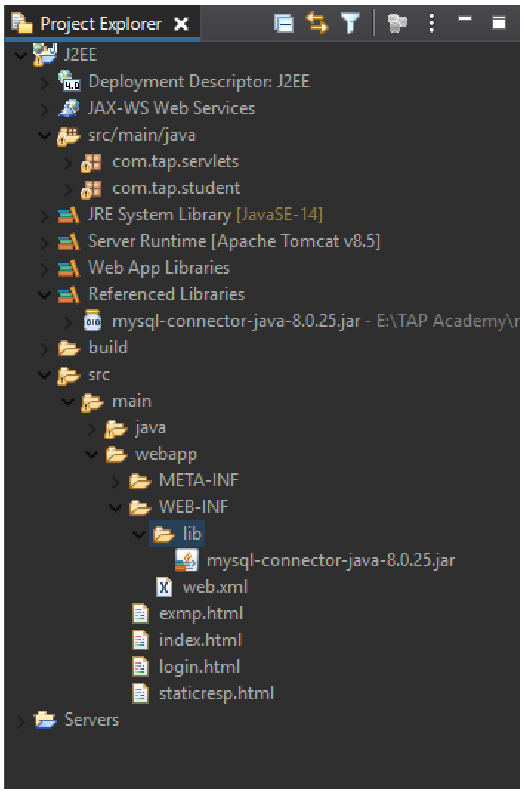


id	name	10th	12th	grad	un	pwd
1	Sagar	60	65	68	sagar.s	ss123
2	Arun	50	65	56	arun.n	ar133
3	Arul	70	72	75	arul.l	al225
4	Bob	80	60	65	bob.b	bo85

2. Now create a html file called **'login.html'** in eclipse and type in the following-

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
      <h3>Login Page</h3>
      <form action="">
            <label>Username</label>
            <input type="text" name="username"><br/>
            <label>Password</label>
            <input type="text" name="password"><br/>
            <input type="submit">
      </form>
</body>
</html>
```

3. Create a new package **"com.tap.student"** and create a class called **"Exmp.java"**
4. Add the **"mysql-connector-java-8.0.25"** jar file in the project as previously explained in JDBC class
5. Copy the **"mysql-connector-java-8.0.25"** jar file and paste it in the "**lib**" folder in **WEB-INF** folder



```
Project Explorer
J2EE
  Deployment Descriptor: J2EE
  JAX-WS Web Services
  src/main/java
    com.tap.servlets
    com.tap.student
  JRE System Library [JavaSE-14]
  Server Runtime [Apache Tomcat v8.5]
  Web App Libraries
  Referenced Libraries
    mysql-connector-java-8.0.25.jar - E:\TAP Academy\
  build
  src
    main
      java
      webapp
        META-INF
        WEB-INF
          lib
            mysql-connector-java-8.0.25.jar
          web.xml
        exmp.html
        index.html
        login.html
        staticresp.html
Servers
```

6. Now we will override the init( ), doGet( ) and destroy( ) methods

```java
package com.tap.student;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class Exmp extends HttpServlet {
	Connection con = null;
	PreparedStatement pstmt = null;
	ResultSet res = null;
	String url = "jdbc:mysql://localhost:3306/tapacademy";
	String un = "root";
	String pwd = "root";

	@Override
	public void init() throws ServletException {
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			DriverManager.getConnection(url, un, pwd);

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

	}

	@Override
	protected void doGet(HttpServletRequest req, HttpServletResponse resp)
			throws ServletException, IOException {

		String username = req.getParameter("username");
		String password = req.getParameter("password");
	}

	@Override
	public void destroy() {

	}

}
```

Now we have to write the logic to check for valid username and password details.
We have to write an incomplete query because the username and password are sent by the client and when the query is executed, a ResultSet is generated with one row if the username and password are valid otherwise we will get an empty ResultSet.
Let us implement this in our code-

```
@Override
      protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                  throws ServletException, IOException {

            resp.setContentType("text/html");
            PrintWriter writer =  resp.getWriter();

            String username = req.getParameter("username");
            String password = req.getParameter("password");

            try {
                  String query = "select * from tapstudent where un = ? and pwd = ?";
                  pstmt = con.prepareStatement(query);
                  pstmt.setString(1, username);
                  pstmt.setString(2, password);
                  res = pstmt.executeQuery();

                  if (res.next() == true) {
                        writer.println("<h3>Welcome "+ res.getString(2)+"!</h3>");
                  }
                  else {
                        writer.println("<h3>Invalid login details.Please try 
again!</h3>");
                  }

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

Register the Exmp.java servlet in the deployment descriptor(web.xml)

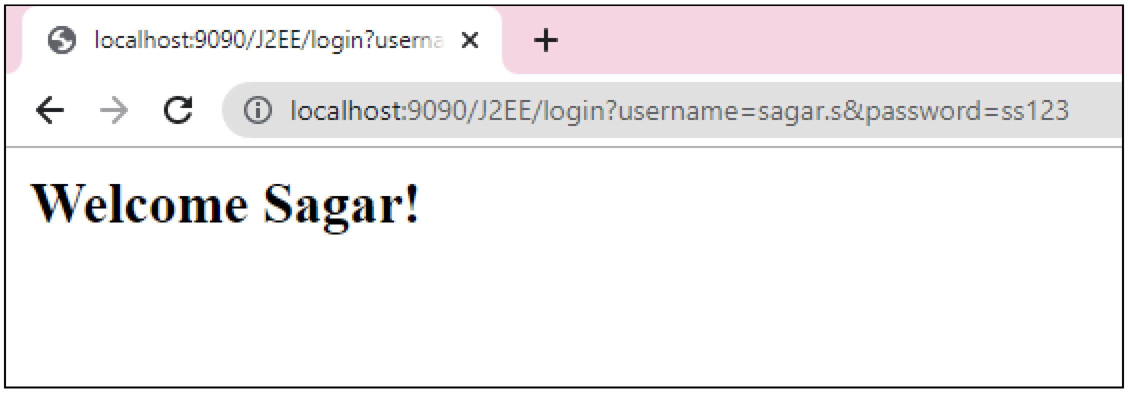
```
<servlet>
      <servlet-name>Login</servlet-name>
      <servlet-class>com.tap.student.Exmp</servlet-class>
</servlet>

<servlet-mapping>
      <servlet-name>Login</servlet-name>
      <url-pattern>/login</url-pattern>
</servlet-mapping>
```

But when we execute the code in the browser, we will not get the expected output because we did not mention what action must be taken after the form is submitted so let us do that now by mentioning the URL in the action form in **login.html**

```
<form action="login">
```

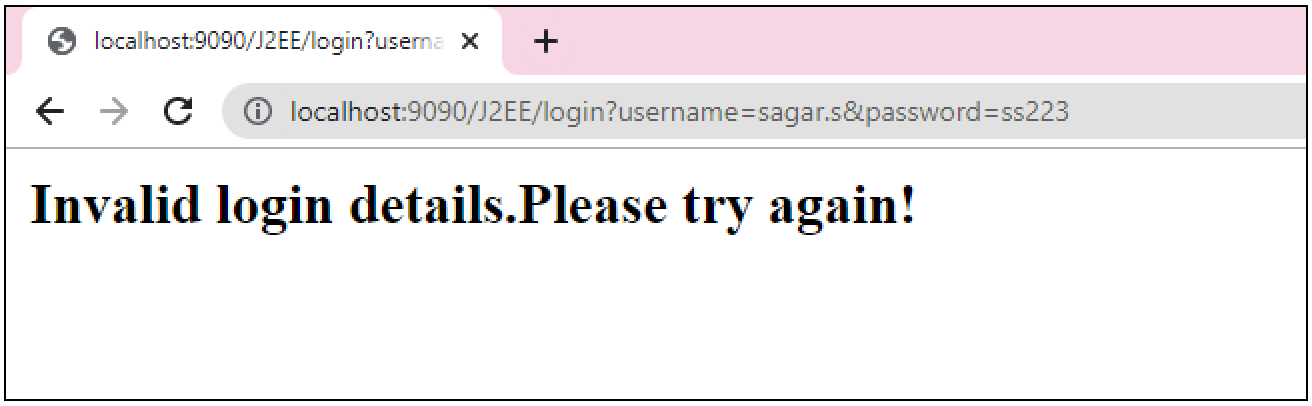
When we execute the code with valid username and password, we will the output as shown below-



localhost:9090/J2EE/login?username=sagar.s&password=ss123

Welcome Sagar!

With invalid details-



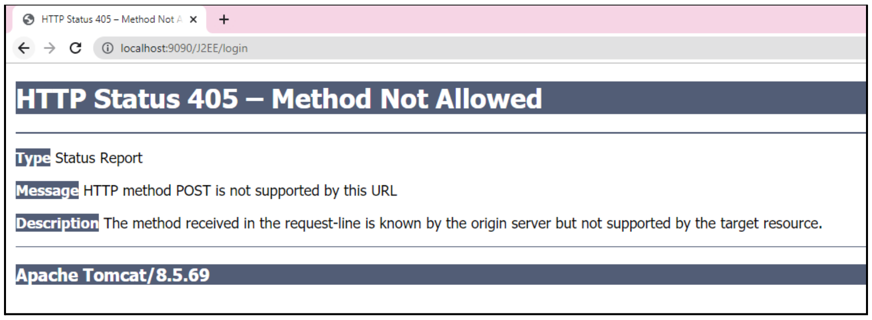
localhost:9090/J2EE/login?username=sagar.s&password=ss223

Invalid login details.Please try again!

But we can clearly see in the query string that the password is visible and this should not happen and to avoid this we must now change the request from **Get to Post** request because whenever data is sent to the browser it should not be exposed in the Query String.
So when we click on the submit button, the request must be changed from Get to Post.
This can be achieved by doing some changes in the html form as shown -
There is an attribute called method and default value of it is **"get"** and we have to change it to **"post"**

```
<form action="login" method="post">
```

And when we execute this, we can see that QueryString didn't get generated but we got an error-



localhost:9090/J2EE/login

HTTP Status 405 – Method Not Allowed

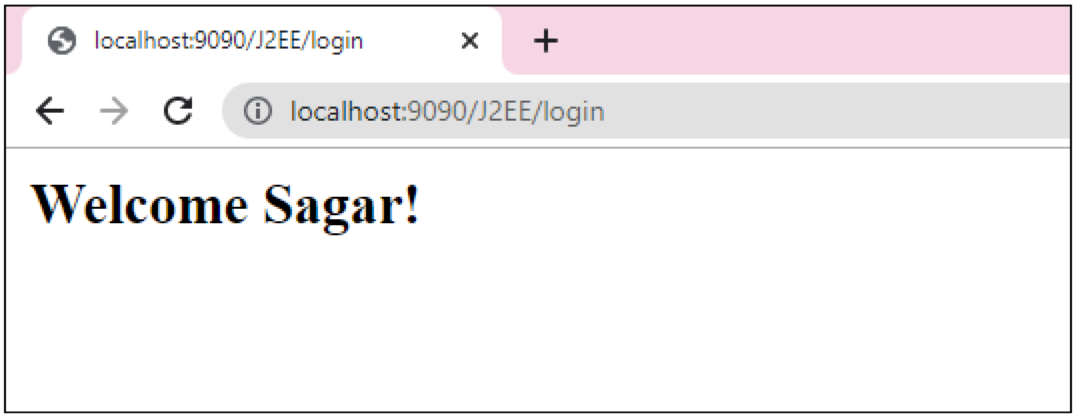
Type Status Report

Message HTTP method POST is not supported by this URL

Description The method received in the request-line is known by the origin server but not supported by the target resource.

Apache Tomcat/8.5.69

Since we have used the doGet() service method in our code but we are requesting a Post request in the form so to overcome this error, we have to change the service method from doGet( ) to doPost( ).
We can clearly see that now the user credentials are not visible in the query string and we got the expected output-



Welcome Sagar!

So whenever we are requesting a lot of data from the client and if that data should not be exposed, then we have to use the **doPost( )** service method.
Now all that is remaining is to deallocate all the resources by using the destroy( )

```
@Override
      public void destroy() {
            try {
                  res.close();
                  pstmt.close();
                  con.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
```

Remember : It is always a good programming practice to deallocate the resources.

Differences between Get and Post

	GET	POST
1.	Get is the default methodology	Post is not the default methodology
2.	Get is used to fetch information from the browser	Post is used to provide data to the browser
3.	Query String is visible	Query String is not visible
4.	Amount of data is limited to 2KB	No data limit
5.	Supports ASCII characters	Supports non-ASCII characters
6.	Get operation is safe to repeat(Idempotent)	Post operation is not safe to repeat(Non-Idempotent)
7.	Supports caching	Does not support caching