ADVANCE JAVA

PANKAJ SIR ACADEMY NOTES

BATCH APRIL 2022

(LIVE LECTURES)

Prepared by:

_M$A_


PA
QUALITY ASSURED

INDEX

_M$A_

16/06/2022 (Thursday)

To develop a web page we need server to run on it, there are lot of servers available, here we are using Apache Tomcat® server.
Before creating a dynamic web project using html, let's take a little introduction of html.

INTRODUCTION OF HTML

The major points of HTML are given below:

- HTML stands for HyperText Markup Language.
- HTML is used to create web pages and web applications.
- HTML is widely used language on the web.
- We can create a static website by HTML only.
- Technically, HTML is a Markup language rather than a programming language.

Q . What is HTML?

A. HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

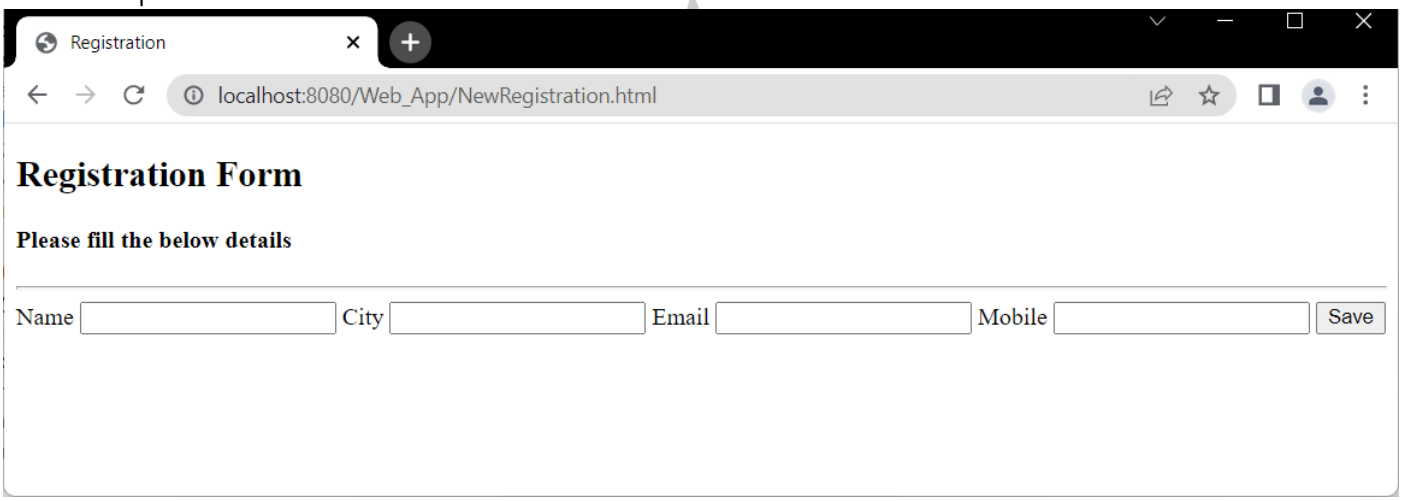
Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.
Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

Example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Registration</title>
</head>
<body>
	<h2> Registration Form </h2>
	<h4> Please fill the below details </h4>
	<hr>
	Name <input type="text" />
	City <input type="text" />
	Email <input type="text" />
	Mobile <input type="text" />
	<input type="submit" value="Save" />
</body>
</html>
```

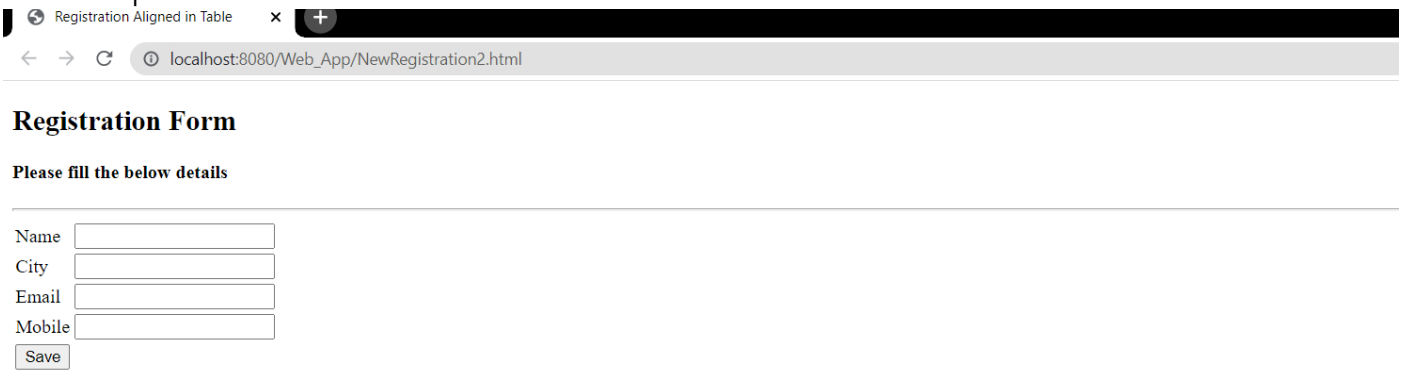
Web Output:



Example #2:

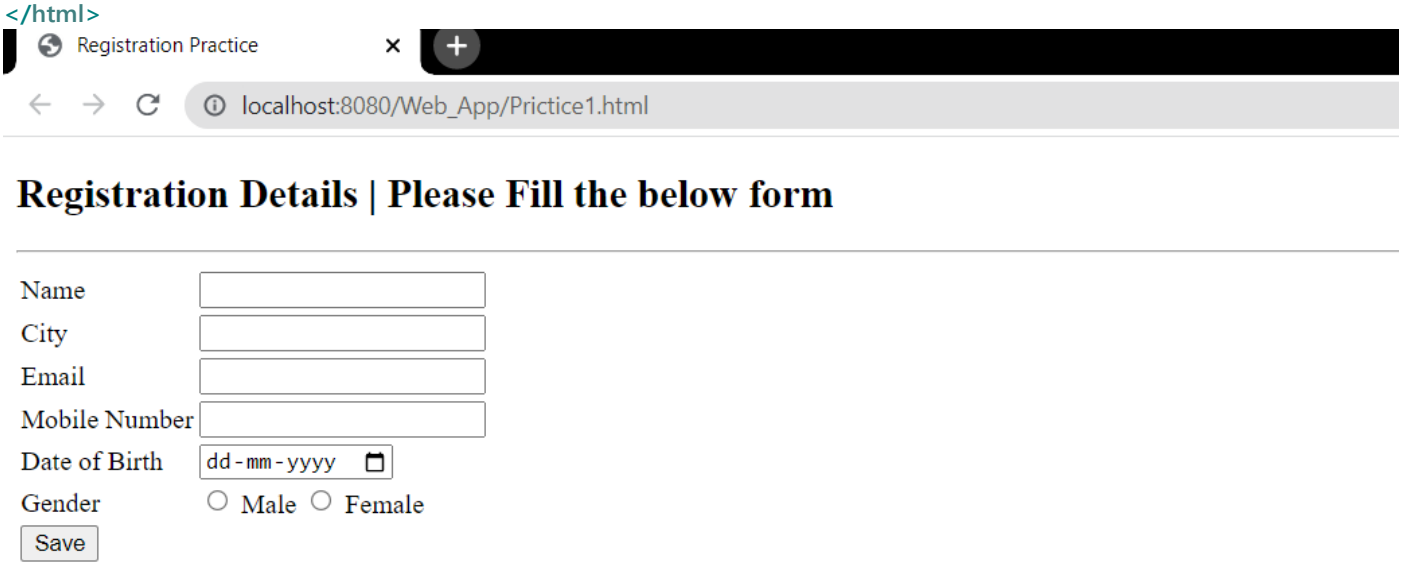
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Registration Aligned in Table</title>
</head>
<body>
	<h2> Registration Form </h2>
	<h4> Please fill the below details </h4>
	<hr>

	<table>		<!-- Table to Align our Form inputs  -->
		<tr>
			<td>Name</td>
			<td><input type="text" /></td>
		</tr>
		<tr>
			<td>City</td>
			<td><input type="text" /></td>
		</tr>
		<tr>
			<td>Email</td>
			<td><input type="text" /></td>
		</tr>
		<tr>
			<td>Mobile</td>
			<td><input type="text" /></td>
		</tr>
		<tr>
			<td><input type="submit" value="Save" /></td>
		</tr>
	</table>

</body>
</html>
```

Web output:



Example #3:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> Registration Practice</title>
</head>
<body>
	<h2> Registration Details | Please Fill the below form</h2>
	<hr>

	<table>
		<tr>
			<td>Name</td>
			<td><input type="text"/> </td>
		</tr>
		<tr>
			<td>City</td>
			<td><input type="text"/> </td>
		<tr>
			<td>Email</td>
			<td><input type="text"/> </td>
		</tr>
		<tr>
			<td>Mobile Number</td>
			<td><input type="text"/> </td>
		</tr>
		<tr>
			<td>Date of Birth</td>
			<td><input type="date"/> </td>
		</tr>
		<tr>
			<td>Gender</td>
			<td> <input type="radio" name="gender" value="male"> Male
<input type="radio" name="gender" value="female"> Female </td>
		</tr>
		<tr>
			<td><input type="submit" value="Save" /></td>
		</tr>
	</table>
</body>
</html>
```



17/06/2022 (Friday)

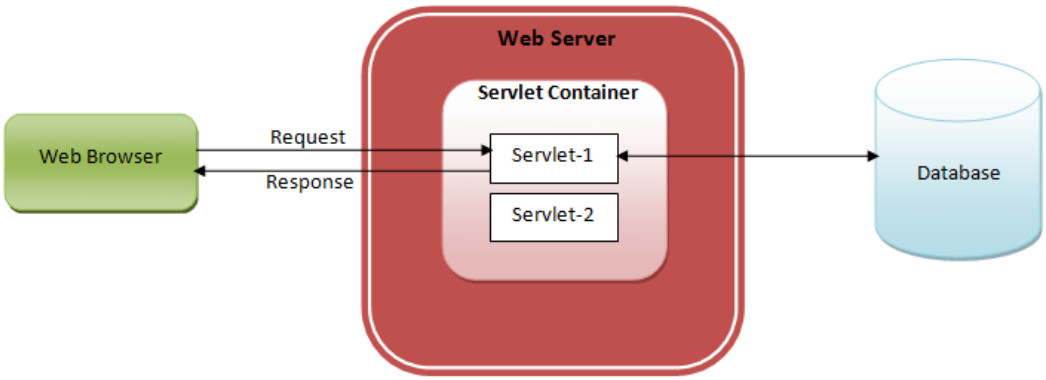
READING THE DATA FROM THE FRONTEND

Let's read the user's input data, to do that we need a servlet.

Q . What is servlet?

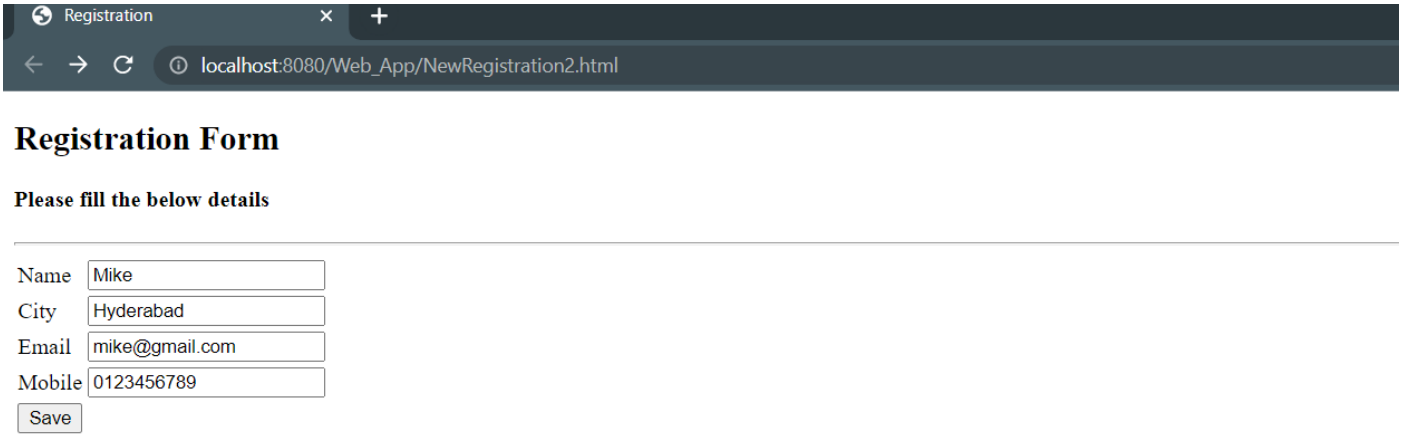
A. Servlet is a JAVA class which extends HttpServlet responsible to read the data from the frontend, it is used to develop the backend code & it helps us to read the data from the form, it has two methods in it doGet and doPost.

Java Server Architecture:



Example #1:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Registration </title>
</head>
<body>
	<h2> Registration Form </h2>
	<h4> Please fill the below details </h4>
	<hr>

	<form action="new" method="post">
		<table> <!--Form tag used to take user input and save to the database  -->
			<tr>
				<td>Name</td>
				<td><input type="text" name="firstName" /></td>
			</tr>
			<tr>
				<td>City</td>
				<td><input type="text" name="city" /></td>
			</tr>
			<tr>
				<td>Email</td>
				<td><input type="text" name="email" /></td>
			</tr>
			<tr>
				<td>Mobile</td>
				<td><input type="text" name="mobile"/></td>
			</tr>
			<tr>
				<td><input type="submit" value="Save" /></td>
			</tr>
		</table>
	</form>

</body>
</html>
```



JAVA Servlet Code, It has two methods, doGet and doPost, in html <form> tag if method="post" it will call doPost Method, if method="get' it will call doGet Method.

```
package p1;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
@WebServlet("/new") //Action attribute name should be same as here "new"
//If @WebServlet annotaion is missing, we should manually type it
public class NewRegistration2 extends HttpServlet { //It's a child class of HttpServlet.
      private static final long serialVersionUID = 1L;
    public NewRegistration2() {
        super();
    }
      protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
            System.out.println("Get");
      }
      protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
            String name = request.getParameter("firstName");
            String city = request.getParameter("city");
            String email = request.getParameter("email");
            String mobile = request.getParameter("mobile");
            System.out.println(name);
            System.out.println(city);
            System.out.println(email);
            System.out.println(mobile);

      }
}
```

Output:
Mike
Hyderabad
mike@gmail.com
0123456789

20/06/2022 (Monday)

SAVING THE USER DATA INTO THE DATABASE

Note:Before interacting with the database, we need to add the .jar file into *<Project/src/main/webapp/WEB-INF/lib>.*

Just adding this .jar file is not enough, in JAVA it happens automatically but in Web Application, we need to locate it manually by writing

`Class.forName("com.mysql.jdbc.Driver");` (Older Approach)

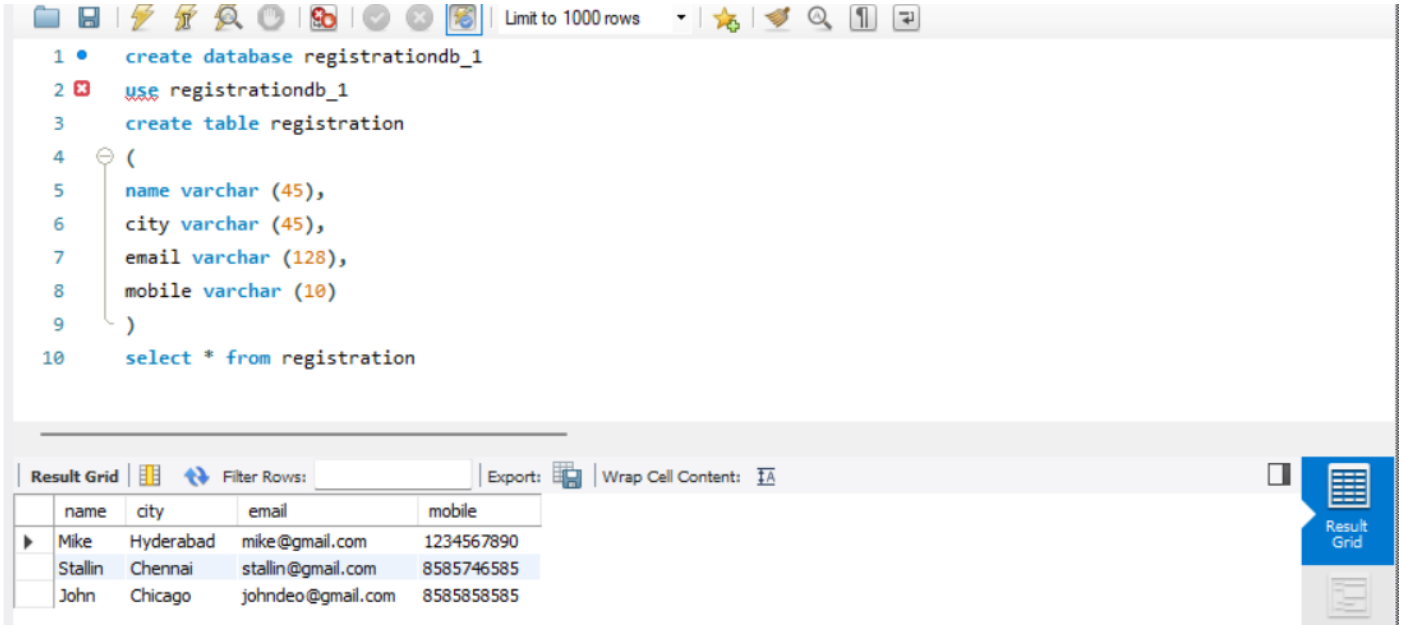
Or

`Class.forName("com.mysql.cj.jdbc.Driver");` (Latest Approach)

Example: *html code will be the same as previous, and to save the user input data into the database the below is the Java Servlet code.*

```
package p1;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
@WebServlet("/new") //A0ction attribute name should be same as here "new"
//If @WebServlet annotaion is missing, we should manually type it
public class NewRegistration extends HttpServlet { //It's a child class of HttpServlet.
private static final long serialVersionUID = 1L;
public NewRegistration() {
    super();
  }
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
}
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            String name = request.getParameter("firstName");
            String city = request.getParameter("city");
            String email = request.getParameter("email");
            String mobile = request.getParameter("mobile");

try {
Class.forName("com.mysql.cj.jdbc.Driver");
//To help TomCat Locate DriverClass after adding .jar file
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/registra-
tiondb_1", "root", "Test");
Statement stmnt = con.createStatement();
stmnt.executeUpdate("insert into registration values('"+name+"', '"+city+"', '"+email+"',
'"+mobile+"')");
//if we directly type the values ('name', 'City', 'email', 'mobile') it will save the
data as we typed,It won't save the user's input data, to save that we need to write vari-
bale names in ('"+ +"') as above.it should turn to brown (which indicates that the values
present in that variable, that value is going to DB
con.close();
} catch (Exception e) {
      e.printStackTrace();
            }
      }
}
```

We saved 3 Users Data, Let's check MySQL DataBase:



```
1  create database registrationdb_1
2  use registrationdb_1
3  create table registration
4  (
5  name varchar (45),
6  city varchar (45),
7  email varchar (128),
8  mobile varchar (10)
9  )
10 select * from registration
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

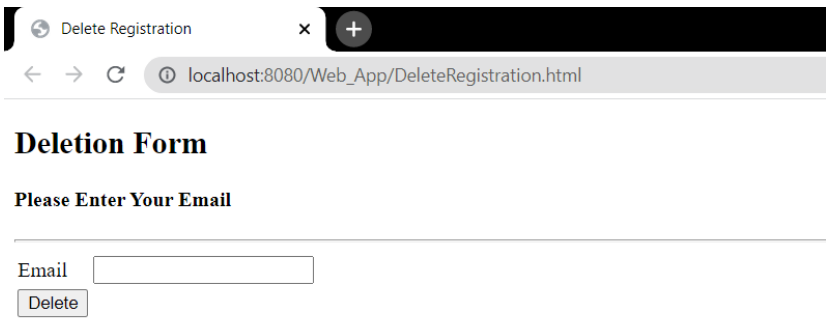
name	city	email	mobile
Mike	Hyderabad	mike@gmail.com	1234567890
Stallin	Chennai	stallin@gmail.com	8585746585
John	Chicago	johndeo@gmail.com	8585858585

DELETING THE USER DATA FROM THE DATABASE

Let's delete the data from the database using "email", here in this example we're going to delete Mike's Data.

html code:

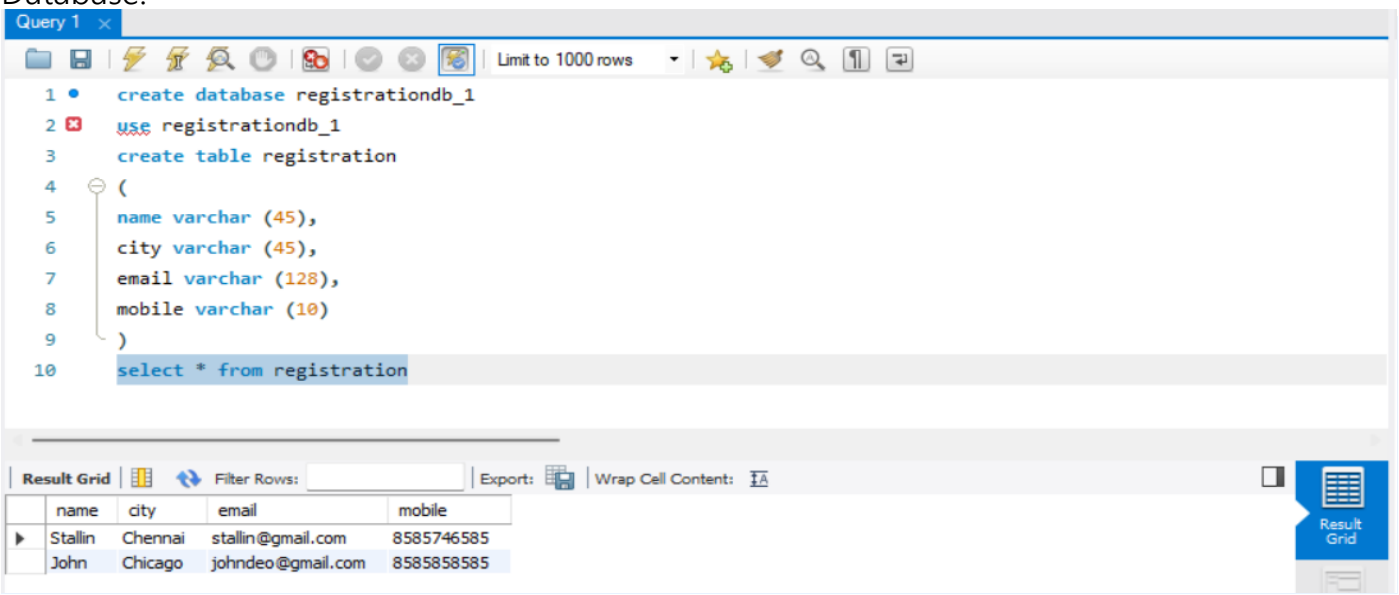
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Delete Registration </title>
</head>
<body>
	<h2> Deletion Form </h2>
	<h4> Please Enter Your Email </h4>
	<hr>

	<form action="delete" method="post">
		<table>
			<tr>
				<td>Email</td>
				<td><input type="text" name="email" /></td>
			</tr>
			<tr>
				<td><input type="submit" value="Delete" /></td>
			</tr>
		</table>
	</form>

</body>
</html>
```



JAVA Servlet Code:

```
package p1;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
@WebServlet("/delete")
public class DeleteRegistration extends HttpServlet {
private static final long serialVersionUID = 1L;
public DeleteRegistration() {
        super();
    }
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            System.out.println("Get");
      }
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            String email = request.getParameter("email");

try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/registra-
tiondb_1", "root", "Test");
      Statement stmnt = con.createStatement();
      stmnt.executeUpdate("delete from registration where email='"+email+"'");
      con.close();
      } catch (Exception e) {
            e.printStackTrace();
            }
      }
}
```

Database:

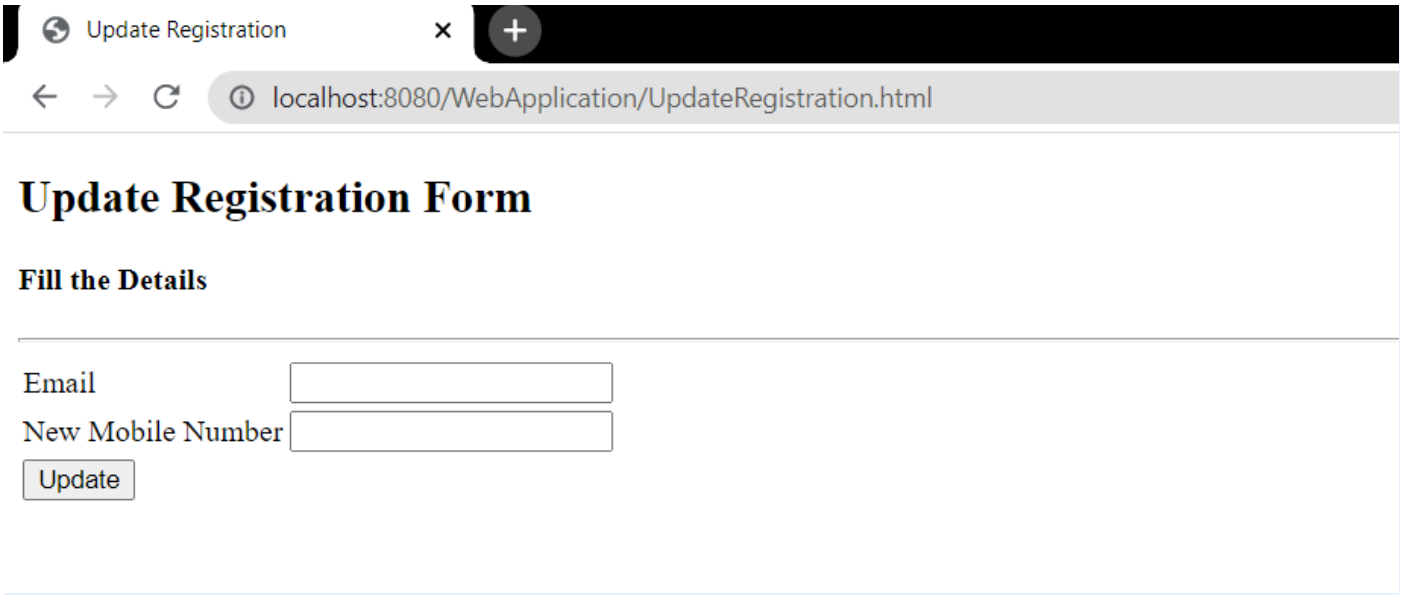


UPDATING THE USER DATA IN THE DATABASE

Let's update the Mobile Number of the John using Email.

html Code:

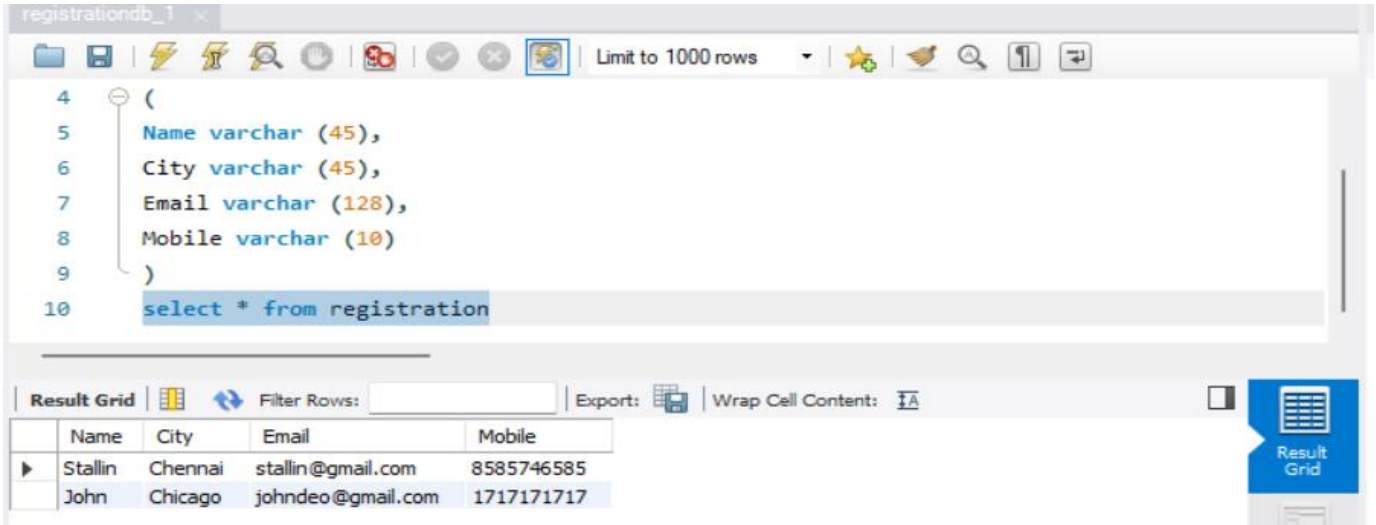
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> Update Registration </title>
</head>
<body>
	<h2> Update Registration Form </h2>
	<h4> Fill the Details </h4>
	<hr>

	<form action="update" method="post">
		<table>
			<tr>
				<td>Email</td>
				<td><input type="text" name="email" /></td>
			</tr>
			<tr>
				<td>New Mobile Number</td>
				<td><input type="text" name="mobile"/></td>
			</tr>
			<tr>
				<td><input type="submit" value="Update" /></td>
			</tr>
		</table>
	</form>

</body>
</html>
```



JAVA Servlet Code:

```
package p1;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
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;
@WebServlet("/update")
public class UpdateRegistration extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public UpdateRegistration() {
        super();
    }
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
	ServletException, IOException {
		System.out.println("Get");
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
	ServletException, IOException {
		String email = request.getParameter("email");
		String mobile = request.getParameter("mobile");

	try {
	Class.forName("com.mysql.cj.jdbc.Driver");
	Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/registra-
tiondb_1", "root", "Test");
	Statement stmnt = con.createStatement();
		stmnt.executeUpdate("UPDATE registration SET Mobile = '"+mobile+"' WHERE Email =
'"+email+"'");
	con.close();
	} catch (Exception e) {
		e.printStackTrace();
		}
	}
}
```

Database:



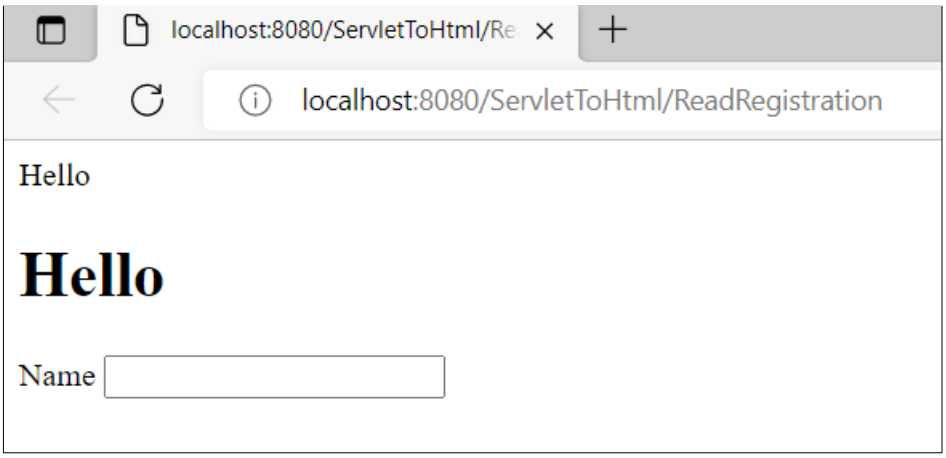
21/06/2022 (Tuesday)

WRITING CONTENT FROM SERVLET TO HTML

If we directly run the servlet on the server it will always run doGet method.

```
package p1;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

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

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
	response.setContentType("text/html"); //to set the content type as html
	PrintWriter out = response.getWriter();
	out.println("Hello");
	out.println("<h1>");
	out.println("Hello");
	out.println("</h1>");
	out.println("Name <input type='text'/>"); //String within string should be in single quotes
	}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
	System.out.println("Post");
	}
}
```

Output:



Now let's directly run the servlet and write the data from database into html in the form of table through servlet.

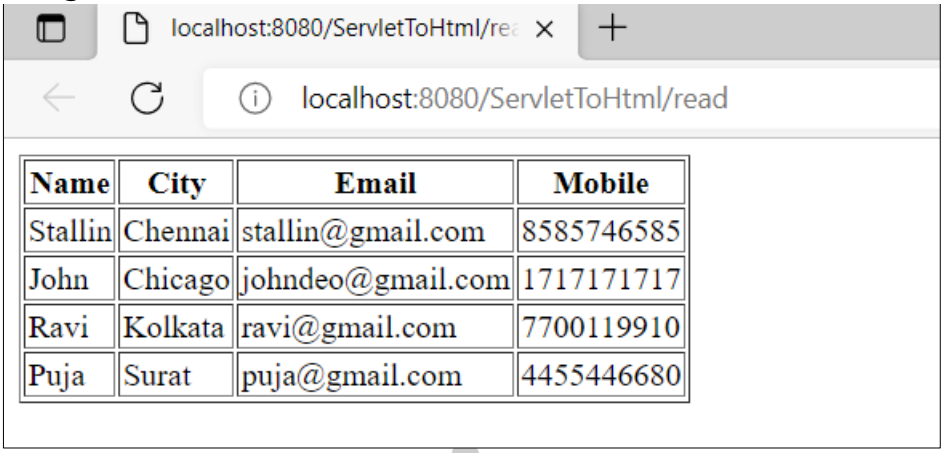
Example #2:

```
package p1;
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;

@WebServlet("/read")
public class ReadRegistration extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public ReadRegistration() {
        super();
    }
      //Dircetly running the servlet on the server will always run doGet method
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
		try {
		Class.forName("com.mysql.cj.jdbc.Driver");
		Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/registra-
tiondb_1", "root", "Test");
		Statement stmnt = con.createStatement();
		ResultSet result = stmnt.executeQuery("Select * from registration");

		response.setContentType("text/html"); //to set the content type as html
		PrintWriter out = response.getWriter();
		out.println("<table border='1'>"); //string within string should be in single quote
			out.println("<tr>");
				out.println("<th>");
					out.println("Name");
				out.println("</th>");
				out.println("<th>");
					out.println("City");
				out.println("</th>");
				out.println("<th>");
					out.println("Email");
				out.println("</th>");
				out.println("<th>");
					out.println("Mobile");
				out.println("</th>");
			out.println("</tr>");

			while(result.next()) {
			out.println("<tr>");
				out.println("<td>");
					out.println(result.getString(1)); //Column#1 in database
				out.println("</td>");
				out.println("<td>");
					out.println(result.getString(2)); //Column#2 in database
				out.println("</td>");
				out.println("<td>");
					out.println(result.getString(3)); //Column#3 in database
				out.println("</td>");
				out.println("<td>");
					out.println(result.getString(4)); //Column#4 in database
				out.println("</td>");
			out.println("</tr>");
			}
		out.println("</table>");
```

Data Present in database:

Name	City	Email	Mobile
Stallin	Chennai	stallin@gmail.com	8585746585
John	Chicago	johndeo@gmail.com	1717171717
Ravi	Kolkata	ravi@gmail.com	7700119910
Puja	Surat	puja@gmail.com	4455446680

OutPut after running the JAVA servlet code:



22/06/2022 (Wednesday)

DEVELOPMENT OF LOGIN PAGE

First of All let's create a database and store values in it, like Email and Password.

```
1   create database login_db
2   use login_db
3   create table login
4   (
5       Email varchar (128),
6       Password varchar (15)
7       )
8
9       insert into login values('msa@gmail.com', 'Test')
10      select * from login
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Email	Password
msa@gmail.com	Test

Result Grid

Development of Login Page in html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Login</title>
</head>
<body>
	<h2>Login to your Account</h2>

	 <form action="verifyLogin" method="post">
	 <table>
		<tr>
		<td>Email</td>
		<td><input type="text" name="email"/></td>
		</tr>
		<tr> <td>Password</td> <td> <input type="password" name="password"/></td> </tr>
		<tr> <td><input type="submit" value ="Login"/> </td> </tr>
	</table>
	</form>
</body>
</html>
```

Login

localhost:8080/LoginApp/Login.html

Login to your Account

Email

Password

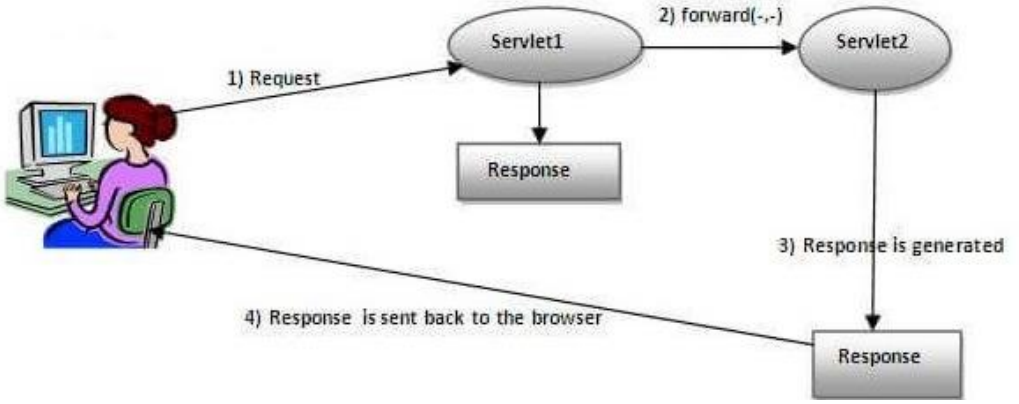
Login

Java Servlet Code to Verify Login Credentials:

```
package p1;
import java.io.IOException;
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;
@WebServlet("/verifyLogin")
public class LoginServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public LoginServlet() {
        super();
    }

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
}
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
	String email = request.getParameter("email");
	String password = request.getParameter("password");

	try {
		Class.forName("com.mysql.cj.jdbc.Driver");
		Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/login_db",
"root", "Test");
		Statement stmnt = con.createStatement();
		ResultSet result = stmnt.executeQuery("select * from login where Email='"+email+"'
and Password='"+password+"'");
		if (result.next()) {
			System.out.println("Welcome");
		}else {
			System.out.println("Invaild Email/Password");
		}

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

23/06/2022 (Thursday)

INTER-SRVLET COMMUNICATION



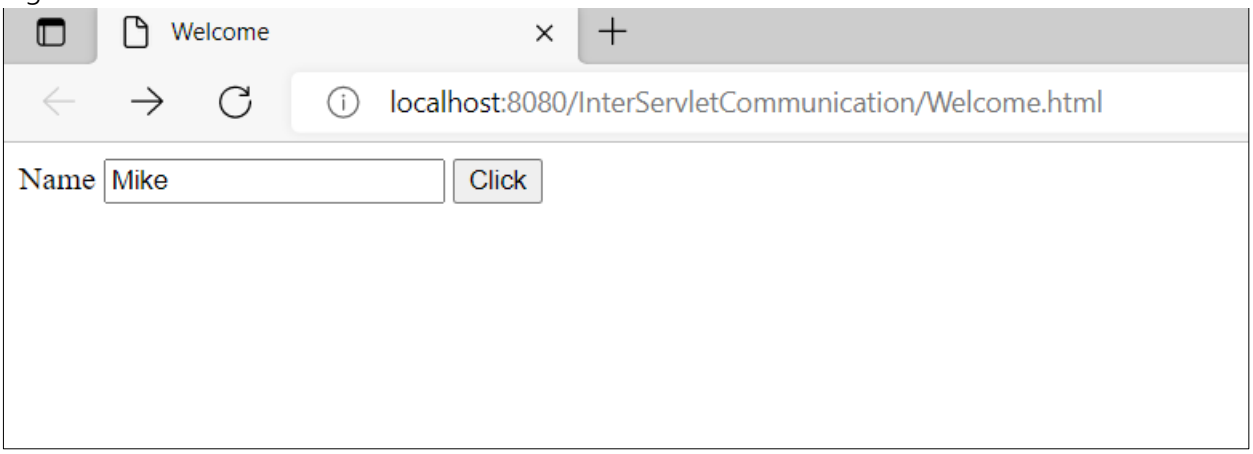
When we call one Servlet from another Servlet using request dispatcher concept then it is called as Inter-Servlet Communication, during Inter-Servlet Communication we can exchange the data between servlets using `request.setAttribute()` and `request.getAttribute()`.

Example #1:

Html code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Welcome </title>
</head>
<body>
	<form action="firstServlet" method="post">
		Name <input type="text" name="name"/>
		<input type="submit" value="Click"/>
	</form>
</body>
</html>
```

Web Page:



First Servlet Code:

```
package p1;
import java.io.IOException;
@WebServlet("/firstServlet")
public class FirstServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public FirstServlet() {
        super();
    }
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		String name = request.getParameter("name");
		request.setAttribute("firstName", name);

		RequestDispatcher rd = request.getRequestDispatcher("secondServlet");
		rd.forward(request, response);
	}
}
```

Second Servlet Code:

```
package p1;
import java.io.IOException;
@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 {
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		String name = (String)request.getAttribute("firstName");
	//return type of it was "Object" (Object is supermost class), Typecasted with "String"
		System.out.println(name);
	}
}
```

Output:
Mike

However the data stored using `request.setAttribute()` is not permanent, it cannot be used without using ReuqestDispatcher, it will be lost when server is Offline & also cannot store the data when we directly run the Servlet, To store the data permanent & use anywhere in the application we can use Session Variable.

SESSION VARIABLE

Html code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Welcome </title>
</head>
<body>
	<form action="firstServlet" method="post">
		Name <input type="text" name="name"/>
		<input type="submit" value="Click"/>
	</form>
</body>
</html>
```

FirstServlet Code:

```
package p1;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
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;
@WebServlet("/firstServlet")
public class FirstServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public FirstServlet() {
        super();
    }
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		String name = request.getParameter("name");
		request.setAttribute("firstName", name);

		HttpSession session = request.getSession(); //Session Variable
		session.setAttribute("sessionName", name);

		RequestDispatcher rd = request.getRequestDispatcher("secondServlet");
		rd.forward(request, response);
	}
}
```

SecondServlet Code:

```
package p1;
import java.io.IOException;
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;
@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 {
		String name = (String)request.getAttribute("firstName"); //TypeCasted
		System.out.println(name);
		//It will print “null” value when we run this servlet directly


		HttpSession session = request.getSession();
		String sessionVal = (String)session.getAttribute("sessionName"); //TypeCasted
		System.out.println(sessionVal);
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		String name = (String)request.getAttribute("firstName"); //TypeCasted
		System.out.println(name);

		HttpSession session = request.getSession();
		String sessionVal = (String)session.getAttribute("sessionName"); //TypeCasted
		System.out.println(sessionVal);

	}
}
```

When we run this Servlet directly the doGet method will run (as we know directly running the Servlet will run doGet method) & print the output as:

null

Pankaj

“null” because, in `request.getAttribute()` it will work only with RequestDispatcher and the data was not permanent, but in Session Variable even after restarting the server the data will still be remembered.

24/06/2022 (Friday)

JSP (JAVA SERVER PAGES)

JSP stands for Java Server Pages/JAKARTA Server Pages, Inside .jsp we can embed partial JAVA Code, It is a technology for building web applications that support dynamic content and acts as a Java servlet. It's a different option to a servlet, and it has a lot more capabilities than a servlet.

The following are the JSP scripting elements

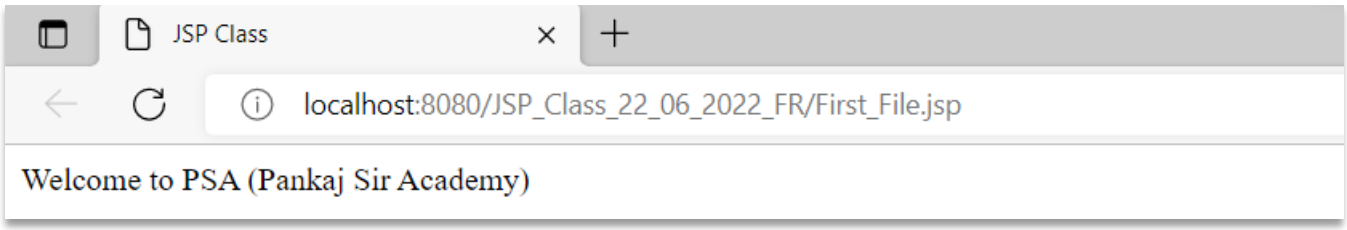
1. JSP Scriptlet Tag
2. JSP Declaration Tag
3. JSP Expression Tag
4. JSP Directive Tag

1. JSP Scriptlet Tag:

In Scriptlet Tag we c

an embed partial JAVA Code using Syntax <% %>, Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Class</title>
</head>
<body>
	<%
		request.setAttribute("msg", "Welcome to PSA");
		out.println(request.getAttribute("msg"));
		session.setAttribute("msg2", "(Pankaj Sir Academy)");
		out.println(session.getAttribute("msg2"));
	%>
</body>
</html>
```



We cannot use JAVA Access Specifiers, Methods & html codes in Scriptlet Tags.

Scriptlet tags have implicit objects like: out, request, response, session, exception, config, application, pageContext and page.

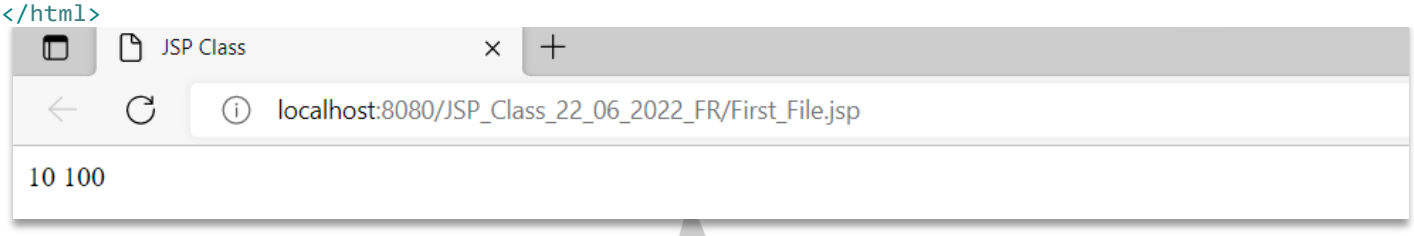
2. JSP Declaration Tag:

We can create Variables, Methods and use Access Specifiers as well in Declaration Tags.

We cannot use implicit objects and neither write html codes in the Declaration Tags.

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Class</title>
</head>
<body>
	<%! //This is a JSP Declaration Tag
		public int x = 10; //Access Specifier and variable
		public int test() {  //Access Specifier and method
			return 100;
		}
	%>

	<% //This is a JSP Scriptlet Tag
		out.println(x); //we cannot use implicit object "out" in Declaration Tag)
		out.println(test());
	%>
</body>
</html>
```



3. JSP Expression Tag

In Expression Tag we need not write `out.println()` to get the output, JSP Expression is equivalent to `out.println()`.

It can consist only one statement, and it should not be terminated with the semicolon;

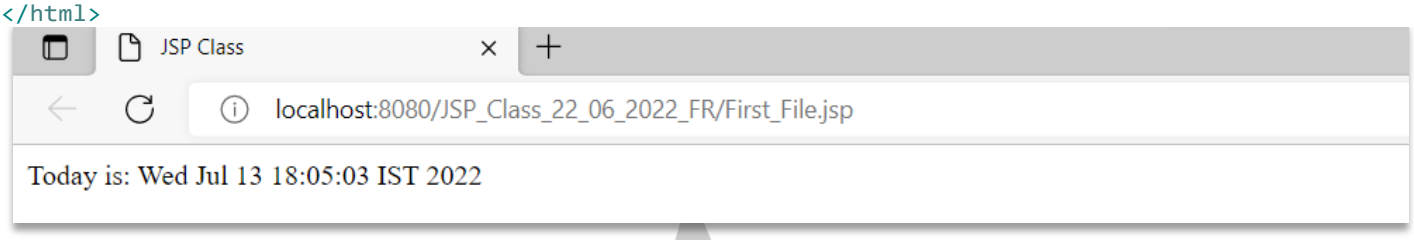
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Class</title>
</head>
<body>
	<%! //This is a JSP Declaration Tag
		public int x = 10; //Access Specifier and variable
		public int test() {  //Access Specifier and method
			return 100;
		}
	%>

	<% //This is a JSP Scriptlet Tag
		out.println(x); //we cannot use implicit object "out" in Declaration Tag
		out.println(test());
	%>

	<br> <!-- "br tag"  is used to break the line -->

	<%= //This is a JSP Expression Tag
		test() + x
	%>

</body>
</html>
```

4. JSP Page Directive Tag:

The Directive tag is used to import the packages from JAVA using "import" keyword, and we can also include a file to our web pages using "include file" keyword.

Example: *(import)*

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.Date" %>   <!-- This is the JSP Directive Tag, Imported Date Package
from JAVA -->
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Class</title>
</head>
<body>
Today is: <%= new Date() %>  <!-- The date method is imported from "java.util.Date" package -->
</body>
</html>
```

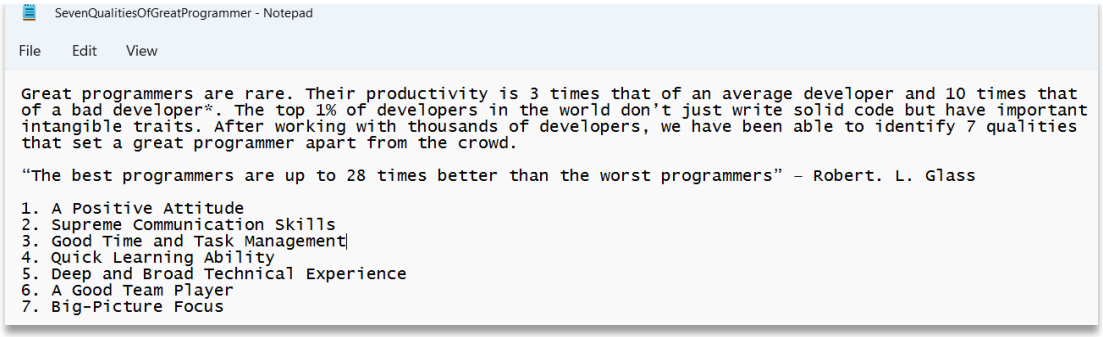


Example: *(include file, here we're including a text file)*

Html code:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.Date" %>   <!-- This is the JSP Directive Tag, Imported Date Package
from JAVA -->
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Class</title>
</head>
<body>
Today's Date and Time: <%= new Date() %>  <!-- The date method is imported from "java.util.Date"
package -->
<br>
<h2>7 Qualities That Differentiate a Great Programmer From a Good Programmer</h2>
<%@ include file = "SevenQualitiesOfGreatProgrammers.txt" %>
<!-- Make sure that the file being included should be in the same path as this .jsp file -->
</body>
</html>
```

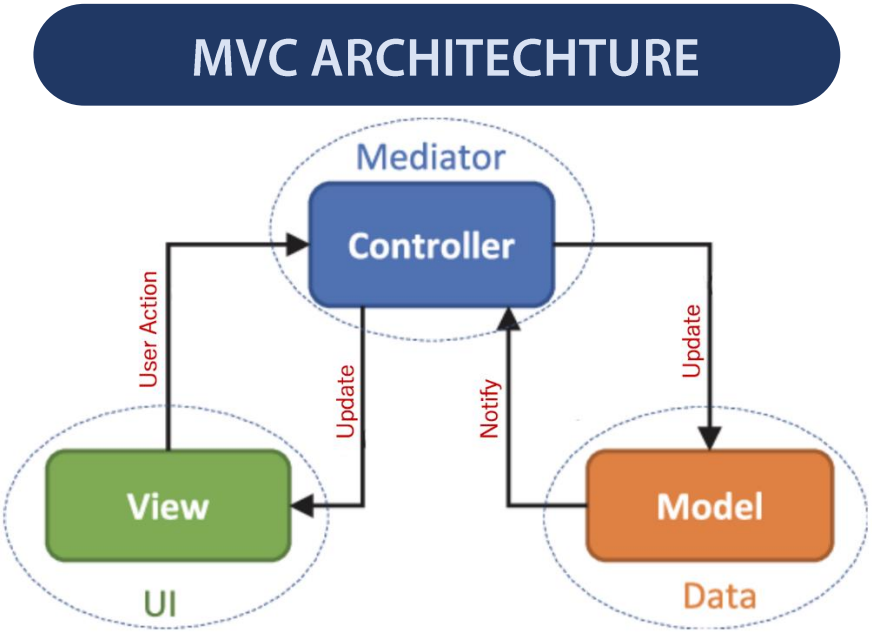
Text File:



Web Output:

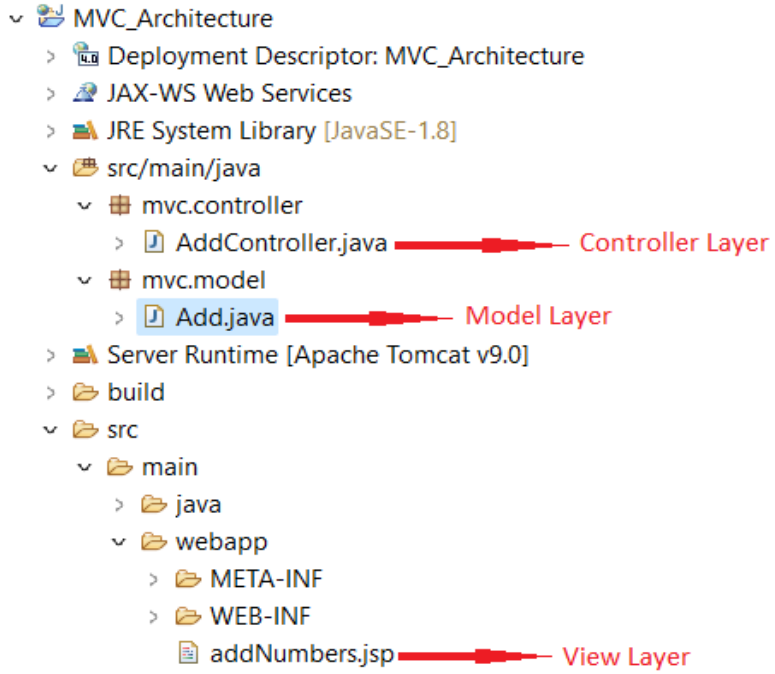

JSP Class
localhost:8080/JSP_Class_22_06_2022_FR/First_File.jsp
Today's Date and Time: Thu Jul 14 12:04:31 IST 2022
7 Qualities That Differentiate a Great Programmer From a Good Programmer
Great programmers are rare. Their productivity is 3 times that of an average developer and 10 times that of a bad developer. The top 1% of developers in the world don't just write solid code but have important intangible traits. After working with thousands of developers, we have been able to identify 7 qualities that set a great programmer apart from the crowd.
"The best programmers are up to 28 times better than the worst programmers" -Robert. L. Glass
1. A Positive Attitude
2. Supreme Communication Skills
3. Good Time and Task Management
4. Quick Learning Ability
5. Deep and Broad Technical Experience
6. A Good Team Player
7. Big-Picture Focus

27/06/2022 (Monday)



All the FrontEnd code like jsp, html etc... we develop that in View Layer. Controller Layer reads the data from View Layer and gives it to the Model Layer. And also it takes the data back from Model Layer and gives it back to View Layer, In other words it acts as a mediator layer to integrate View and Model Layer. Model Layer is responsible to perform business logic implementation, or database operations.

An Overview of the project in Eclipse, Example: Addition.



Java Class (Model Layer):

```
package mvc.model;
public class Add {
	public int addNumbers(int num1, int num2) {
		return num1+num2;
	}
}
```

Java Servlet (Controller Layer):

```
package mvc.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
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 mvc.model.Add;
@WebServlet("/addController")
public class AddController extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public AddController() {
        super();
    }
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
	}
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
	int num1 = Integer.parseInt(request.getParameter("num1"));
	int num2 = Integer.parseInt(request.getParameter("num2"));

	Add a = new Add();
	int result = a.addNumbers(num1, num2);
	request.setAttribute("res", result); //Setting the Attribute
	RequestDispatcher rd = request.getRequestDispatcher("addNumbers.jsp"); //Dispatch Request
to the jsp file
	rd.forward(request, response);
	}
}
```

jsp (view Layer):

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>MVC Example (Addition)</title>
</head>
<body>
<form action="addController" method="post">
	Number 1 <input type="text" name="num1">
	Number 2 <input type="text" name="num2">
	<input type="submit" value="Add">
</form>
<br>
<h2> The Result is <%out.println(request.getAttribute("res")); %> </h2>
</body>
</html>
```

WepPage:

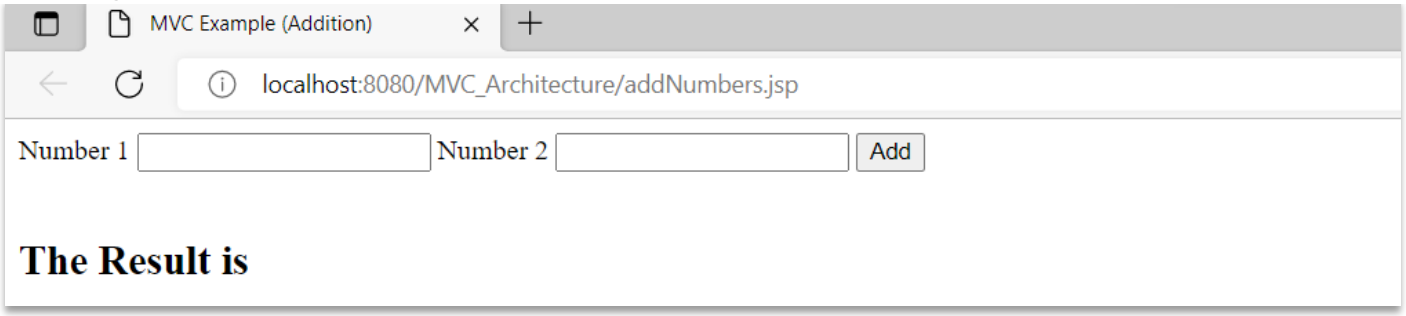
28/06/2022 (Tuesday)

MVC ARCHITECTURE (Contd...)

Earlier in the previous example while running the .jsp file for the first time, it was printing the result as "null", we can overcome the problem just by adding if condition in the JSP Scriptlet Tag. Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>MVC Example (Addition)</title>
</head>
<body>

<form action="addController" method="post">
       Number 1 <input type="text" name="num1">
       Number 2 <input type="text" name="num2">
       <input type="submit" value="Add">
</form>
<br>
<h2> The Result is
<%
//Adding this condition will stop printing "null" while running 1st time
if (request.getAttribute("res")!=null) {
       out.println(request.getAttribute("res"));
}
%> </h2>

</body>
</html>
```

Web Page:



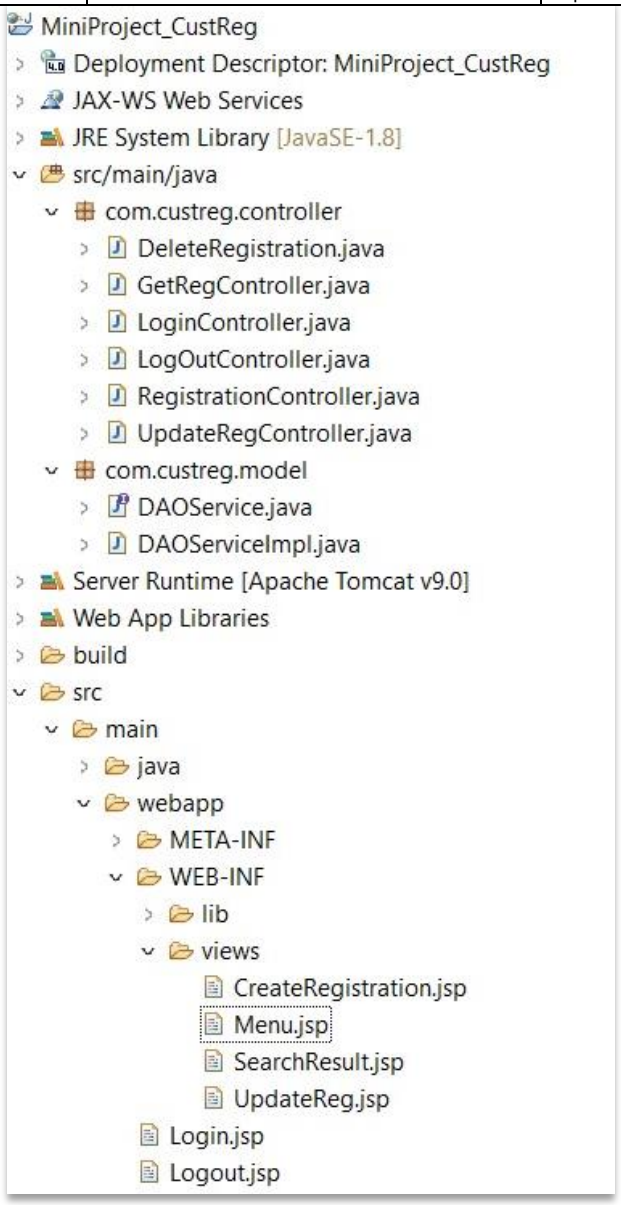
28/06/2022 (Tuesday)

MINI PROJECT

About Project: Create a Web Page that consist of Login and Logout feature, to take the Customer Registration Details like: Name, City, Email, Mobile and save it to the database and perform CRUD Operations.

MVC Architecture of the Project

View Layers	Model Layers	Controller Layers
Login.jsp	DAOService.java	DeleteRegistration.java
CreateRegistration.jsp	DAOServiceImpl.java	GetRegController.java
Menu.jsp		LoginController.java
SearchResult.jsp		LogOutController.java
UpdateReg.jsp		RegistrationController.java
Logout.jsp		UpdateRegController.java



First of all, we have stored the Login Credentials in MySQL Database:

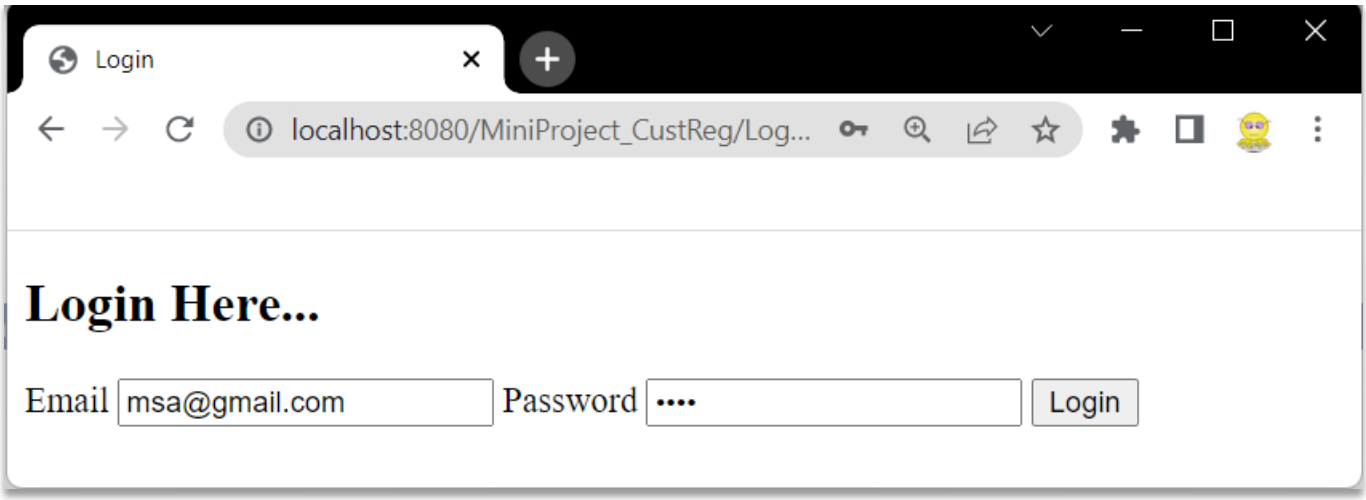
```
9    select * from login
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Email	Password
msa@gmail.com	Test

Login.jsp *(View Layer):*

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Login </title>
</head>
<body>
<h2>
Login Here...
</h2>
	<!-- It will call "LoginController.jsp" -->
	<form action="verifyLogin" method="post">
	Email <input type="email" name="email"/>
	Password <input type="password" name="password"/>
	<input type="submit" value="Login"/>
	</form>

	<!-- Error Message to be printed for Invalid Credentials -->
	<%
		if(request.getAttribute("error") != null) {
			out.println(request.getAttribute("error"));
		}
	%>
</body>
</html>
```



After clicking on "Login" button, It will verify the Credentials by calling LoginController.java `@WebServlet("/verifyLogin").`

LoginController.java: *(Controller Layer)*

```
package com.custreg.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
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;
import com.custreg.model.DAOService;
import com.custreg.model.DAOServiceImpl;

@WebServlet("/verifyLogin")
public class LoginController extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public LoginController() {
        super();
    }
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		String useremail = request.getParameter("email");
		String userpassword = request.getParameter("password");

		DAOService dao = new DAOServiceImpl(); //Connection to database
		dao.connectDB();

		boolean status = dao.verifyCredentials(useremail, userpassword);
	if(status==true) {
	HttpSession session = request.getSession(); //Session Variable to keep track of Logging in
	   session.setAttribute("email", useremail); //here
RequestDispatcher rd = request.getRequestDispatcher("WEB-INF/views/CreateRegistration.jsp");
			rd.forward(request, response); //redirects to a new page
	} else {
			request.setAttribute("error", "Invalid Username/Password");
			RequestDispatcher rd = request.getRequestDispatcher("Login.jsp");
			rd.include(request, response);
//using "include" instead of "forward", because "forward" is used to redirect to new page &
"include" will stay on the same page, actually both works same, but "include" is better here
		}
	}
}
```

This servlet will connect to the database using Model Layers (DAOService.java and DAOServiceImpl.java).

DAOService.java: *(Model Layer) - Interface*

```
package com.custreg.model;
import java.sql.ResultSet;
//DAO Stands for "Data Access Object" Service Layer
//Abstraction (Hiding of Implementation Details)
public interface DAOService {
	public void connectDB();
	public boolean verifyCredentials(String useremail, String userpassword);
	public void saveRegistration(String name, String city, String email, String mobile);
	public ResultSet listAllReg(); //import the ResultSet file
	public void deleteByEmailId(String email);
	public void updateReg(String email, String mobile);
}
```

DAOServiceImpl.java *(Model Layer) – JAVA Class*

```
package com.custreg.model;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class DAOServiceImpl implements DAOService { //Implementation of DAOService Interface

//Declaring the variables here so it can be accessed in all the below methods
	private Connection con;
	private Statement stmnt;

	@Override
	public void connectDB() {
		try {
		Class.forName("com.mysql.cj.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/customer_reg_db", "root", "Test");
			stmnt = con.createStatement();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

//Connecting to the database to verify the Login Credentials
	@Override
	public boolean verifyCredentials(String useremail, String userpassword) {
		try {
ResultSet result = stmnt.executeQuery
("select * from login where Email='"+useremail+"' and Password='"+userpassword+"'");
			return result.next();
		} catch (Exception e) {
			e.printStackTrace();
		}
		return false;
	}

//Connecting to the database to insert registration details
	@Override
	public void saveRegistration(String name, String city, String email, String mobile) {
		try {
stmnt.executeUpdate
("insert into registration values ('"+name+"','"+city+"', '"+email+"', '"+mobile+"')");
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

//Connecting to the database to view all the registrations
	@Override
	public ResultSet listAllReg() {
		try {
			ResultSet result = stmnt.executeQuery("select * from registration");
			return result;
		} catch (Exception e) {
			e.printStackTrace();
		}
		return null;
	}

//Connecting to the database to delete the registration
	@Override
	public void deleteByEmailId(String email) {
		try {
			stmnt.executeUpdate("delete from registration where Email='"+email+"'");
		} catch (SQLException e) {
			e.printStackTrace();
```

```
//Connecting to the database to delete the registration
	@Override
	public void deleteByEmailId(String email) {
		try {
			stmnt.executeUpdate("delete from registration where Email='"+email+"'");
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

//Connecting to the database to update the registrations details (Mobile Number)
	@Override
	public void updateReg(String email, String mobile) {
		try {
			stmnt.executeUpdate("UPDATE registration SET Mobile = '"+mobile+"' WHERE
Email='"+email+"'");
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

If the credentials are incorrect, it will show "Invalid Username/Password", If Correct it will redirect to the "CreateRegistration.jsp".

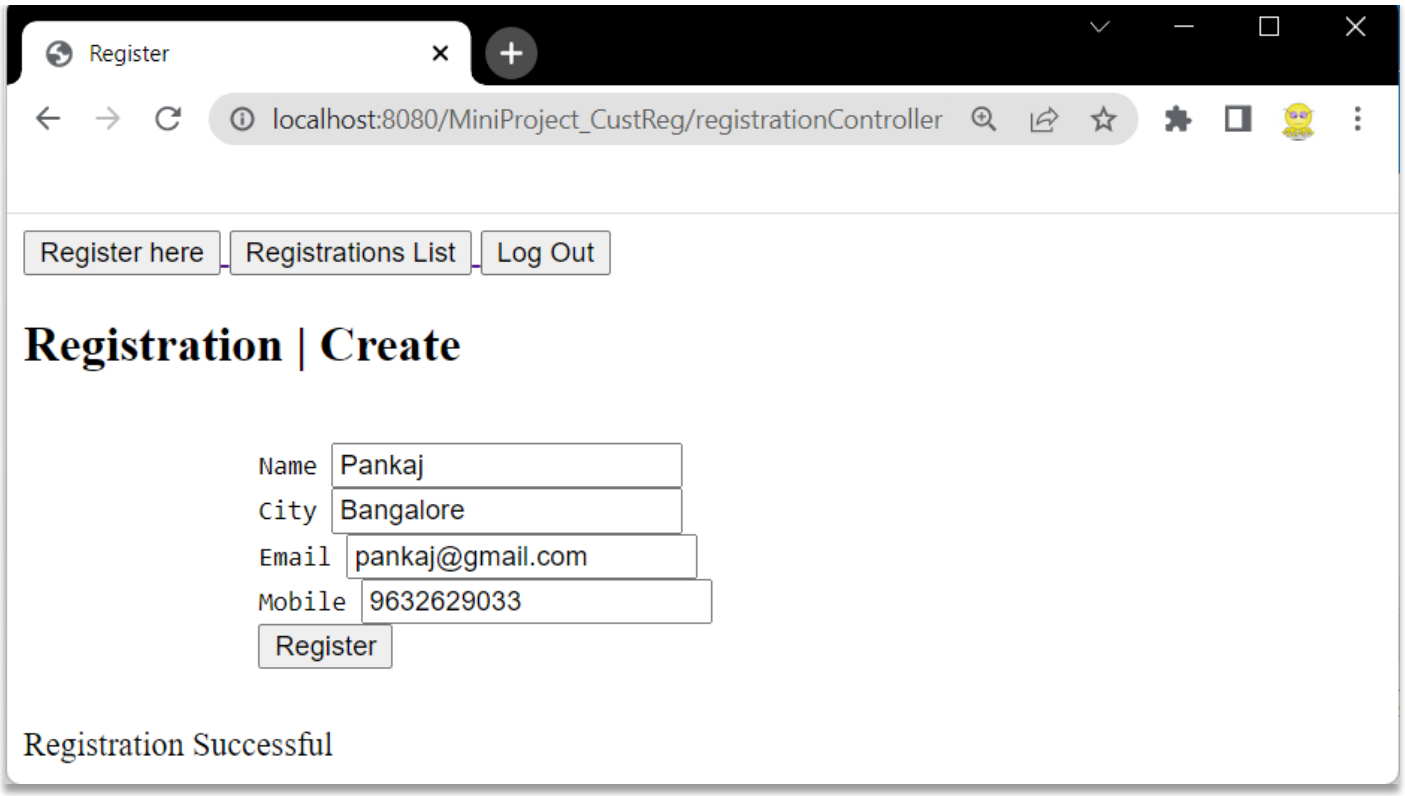
CreateRegistration.jsp *(View Layer)*

```
<!-- This file is created in "WEB-INF>view>" so it gives more security & cannot run directly -->
<!--  It runs only when servlet calls this page-->

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Menu.jsp" %>   <!--this file has a menu tag  -->
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Registration Form</title>
</head>
<body>
	<h2> Registration | Create </h2>
	<form action="registrationController" method="post">
		<pre>		<!--"pre" tag used to move the code to different line  -->
		Name <input type="text" name="name"/>
		City <input type="text" name="city"/>
		Email <input type="text" name="email"/>
		Mobile <input type="text" name="mobile"/>
		<input type="submit" value="Register"/>
		</pre>
	</form>

	<%
		if(request.getAttribute("msg") != null) {
			out.println(request.getAttribute("msg"));
		}
	%>
</body>
</html>
```

CreateRegistration.jsp file also includes "Menu.jsp"

Menu.jsp: *(View Layer)*

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Register </title>
</head>
<body>
       <a href="registrationController"> <input type="button" value="Register here"> </a> <!--
Clicking on the link will call "get" method by default -->
       <a href="getRegController"> <input type="button" value="Registrations List"> </a>
       <a href="logout"> <input type="button" value="Log Out"></a>
</body>
</html>
```



After entering the data, clicking on Register button will call "RegistrationController.java" Servlet to supply the data into the database.

RegistrationController.java: *(Controller Layer)*

```
package com.custreg.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
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 com.custreg.model.DAOService;
import com.custreg.model.DAOServiceImpl;
@WebServlet("/registrationController")
```

```
public class RegistrationController extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public RegistrationController() {
        super();
    }
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		RequestDispatcher rd = request.getRequestDispatcher("WEB-INF/views/CreateRegistra-
tion.jsp");
		rd.forward(request, response); //redirects to Create Registration page
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		HttpSession session = request.getSession(false);
		if(session.getAttribute("email")!=null) {
			String name = request.getParameter("name");
			String city = request.getParameter("city");
			String email = request.getParameter("email");
			String mobile = request.getParameter("mobile");

			DAOService dao = new DAOServiceImpl(); //Connection to database
			dao.connectDB();

//Create new method in DAOService Interface (Click & Quick Fix, & create method)
			dao.saveRegistration(name, city, email, mobile);

			request.setAttribute("msg", "Registration Successful");
			RequestDispatcher rd = request.getRequestDispatcher("WEB-INF/views/CreateReg-
istration.jsp");
			rd.include(request, response); //remains on the same page
		}else {
			RequestDispatcher rd = request.getRequestDispatcher("Login.jsp");
			rd.forward(request, response);
		}
	}
}
```

When we click on the "Registration List" button it will call the "GetRegController.java" Servlet which will fetch the data from the database and pass it to "SearchResult.jsp" and it displays the data in the form of table.

GetRegController.java: *(Controller Layer)*

```
package com.custreg.controller;
import java.io.IOException;
import java.sql.ResultSet;
import javax.servlet.RequestDispatcher;
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;
@WebServlet("/getRegController")
public class GetRegController extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public GetRegController() {
        super();
    }
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		HttpSession session = request.getSession(false);
		if(session.getAttribute("email")!=null) {
			DAOService dao = new DAOServiceImpl();
			dao.connectDB();
```

```
            ResultSet result = dao.listAllReg();
            request.setAttribute("result", result);

        RequestDispatcher rd = request.getRequestDispatcher("WEB-INF/views/SearchResult.jsp");
            rd.forward(request, response);
        }
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
        }
}
```

It will pass data to "SearchResult.jsp"

SearchResult.jsp: *(View Layer)*

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Menu.jsp" %>
<%@ page import="java.sql.ResultSet" %> <!-- This is imported from JAVA -->
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Registrations List</title>
</head>
<body>
<h3> LIST OF THE REGISTRATIONS</h3>
<table border=1>
<tr> <th>Name</th><th>City</th><th>Email</th><th>Mobile</th><th>Delete</th><th>Update</th> </tr>

<%
        ResultSet set = (ResultSet)request.getAttribute("result");
        //Import the ResultSet JAVA package, & TypeCast as (ResultSet)
        while(set.next()) {
%>

<tr>
<td><%=set.getString(1)%></td>
<td><%=set.getString(2)%></td>
<td><%=set.getString(3)%></td>
<td><%=set.getString(4)%></td>

<td><a href="delete?emailId=<%=set.getString(3)%>"><input type="submit" value="Delete"> </a></td>

<!-- "delete?emailId emailId will be stored in variable "emailId" to pass to the Servlet-->

<td><a href="update?emailId=<%=set.getString(3)%>&mobile=<%=set.getString(4)%>"><input type="sub-
mit" value="Update"> </a></td>

<!-- "update?emailId emailId will be stored in variable "emailId" and the mobile number will be
stored in variable "mobile"to pass to the Servlet-->

</tr>
<%}%>
</table>
</body>
</html>
```

We can also delete the data from the database by clicking on the delete button, delete button will call the "DeleteRegistration.java" Servlet to delete the data from the database.

DeleteRegistration.java: *(Controller Layer)*

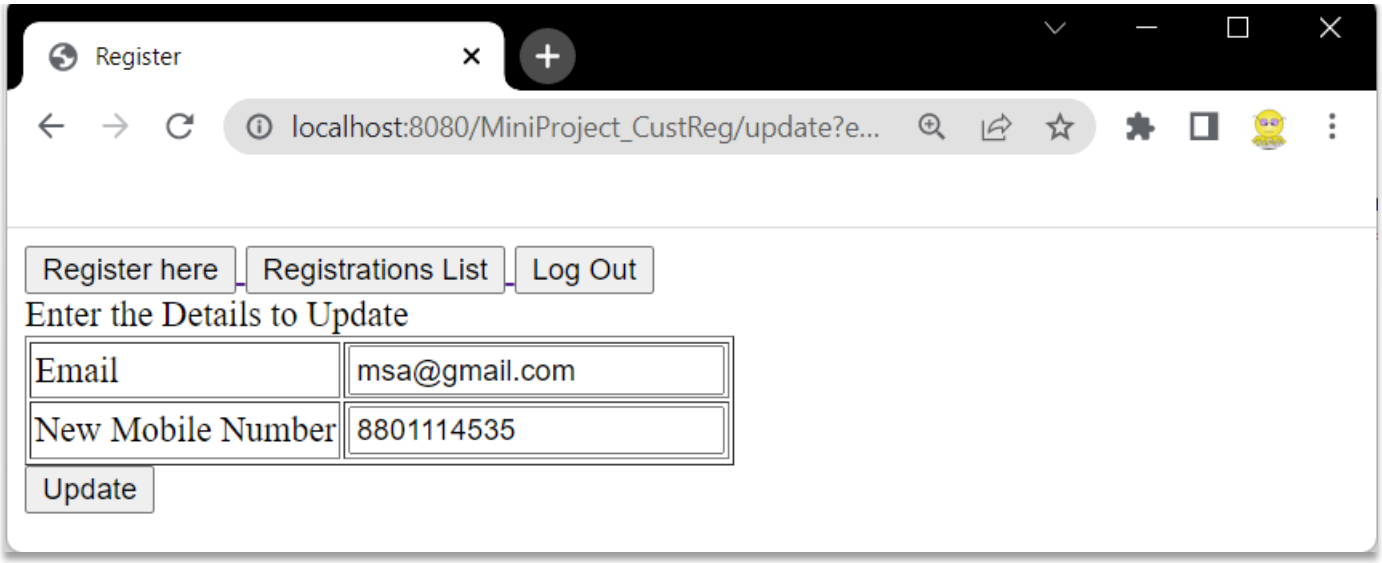
```
package com.custreg.controller;
import java.io.IOException;
import java.sql.ResultSet;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.custreg.model.DAOService;
import com.custreg.model.DAOServiceImpl;
@WebServlet("/delete")
public class DeleteRegistration extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public DeleteRegistration() {
        super();
    }

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx-
ception, IOException {
		HttpSession session = request.getSession(false);
		if(session.getAttribute("email")!=null) {
			String email = request.getParameter("emailId");
			DAOService dao = new DAOServiceImpl();
			dao.connectDB();
			dao.deleteByEmailId(email);

//To Avoid going to the blank Page & stay on same page ("Copied" from "GetRegController.java")
			ResultSet result = dao.listAllReg();
			request.setAttribute("result", result);
	RequestDispatcher rd = request.getRequestDispatcher("WEB-INF/views/SearchResult.jsp");
			rd.forward(request, response);
		 }else {
			 RequestDispatcher rd = request.getRequestDispatcher("Login.jsp");
			 rd.forward(request, response);
		    }
	}
```

```
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        }
}
```

We can also update the record by clicking on the "Update" button, here we have given the feature to update Mobile Number based on Email, Update button will be redirected to the "UpdateReg.jsp" which will call the doGet method of "UpdateRegController.java" Servlet, the Email is auto populated & read-only (which cannot be edited) and after entering new Mobile Number & submitting it calls doPost method of "UpdateRegController.java" Servlet.

UpdateReg.jsp: *(View Layer)*

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Menu.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Update Mobile Number </title>
</head>
<body>
        <form action="update" method="post">
        Enter the Details to Update
        <table border=1>
        <tr><td>Email</td><td><input type="text" name="email" value="<%=request.getAttrib-
ute("email") %>" readonly/></td></tr>
        <tr><td>New Mobile Number</td> <td> <input type="text" name="mobile" value="<%=re-
quest.getAttribute("mobile") %>"/></td> </tr>
        </table>
        <input type="submit" value="Update"/>
        </form>
</body>
</html>
```



UpdateRegController.java: *(Controller Layer)*

```
package com.custreg.controller;
import java.io.IOException;
import java.sql.ResultSet;
import javax.servlet.RequestDispatcher;
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;
import com.custreg.model.DAOService;
import com.custreg.model.DAOServiceImpl;

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

	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		HttpSession session = request.getSession(false);
		if(session.getAttribute("email")!=null) {
			String email = request.getParameter("emailId");
			String mobile = request.getParameter("mobile");

			request.setAttribute("email", email);
			request.setAttribute("mobile", mobile);

		RequestDispatcher rd = request.getRequestDispatcher("WEB-INF/views/UpdateReg.jsp");
		rd.forward(request, response);
		 }else {
			 RequestDispatcher rd = request.getRequestDispatcher("Login.jsp");
			 rd.forward(request, response);
		    }
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		String email = request.getParameter("email");
		String mobile = request.getParameter("mobile");

		DAOService dao = new DAOServiceImpl();
		dao.connectDB();
		dao.updateReg(email, mobile);

		//After updating show the updated result
		ResultSet result = dao.listAllReg();
		request.setAttribute("result", result);

	RequestDispatcher rd = request.getRequestDispatcher("WEB-INF/views/SearchResult.jsp");
	rd.forward(request, response);
	}
}
```

To Logout when we click on the "Logout" button, it will call the "LogOutController.java" Servlet.

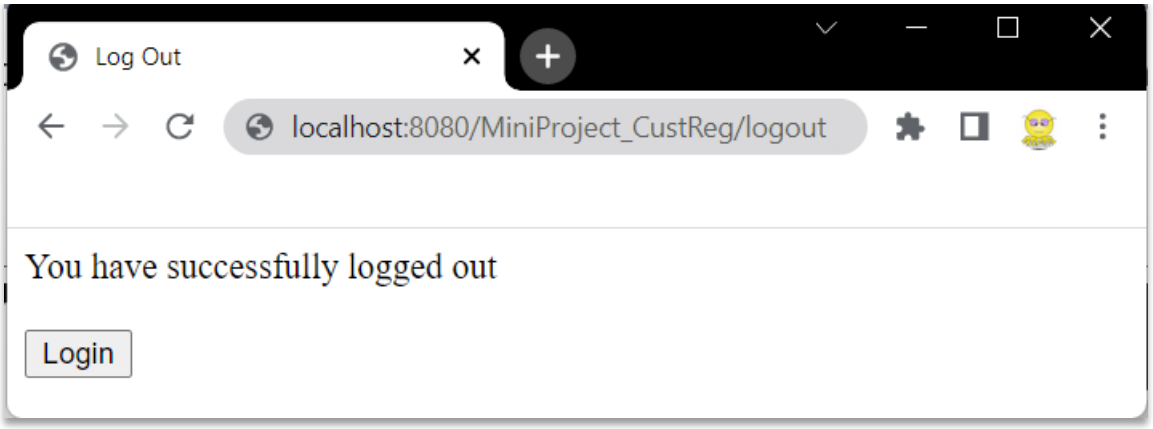
LogOutController.java: *(Controller Layer)*

```
package com.custreg.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
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;

@WebServlet("/logout")
public class LogOutController extends HttpServlet {
	private static final long serialVersionUID = 1L;
    public LogOutController() {
        super();
    }
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		//HTTP Session Class will close the session & also we put "session.getSession"
		on Update and Delete Servlets to deny the database activity after logout
		HttpSession session = request.getSession(false);
		 if(session.getAttribute("email")!=null) {
			try {
			session.invalidate();
			RequestDispatcher rd = request.getRequestDispatcher("Logout.jsp");
			rd.forward(request, response);
		} catch (Exception e) {
			e.printStackTrace();
			}
		 } else {
			RequestDispatcher rd = request.getRequestDispatcher("Login.jsp");
			rd.forward(request, response);
		 }
	}

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

}
```

This servlet will call the "Logout.jsp"

Logout.jsp: *(View Layer)*

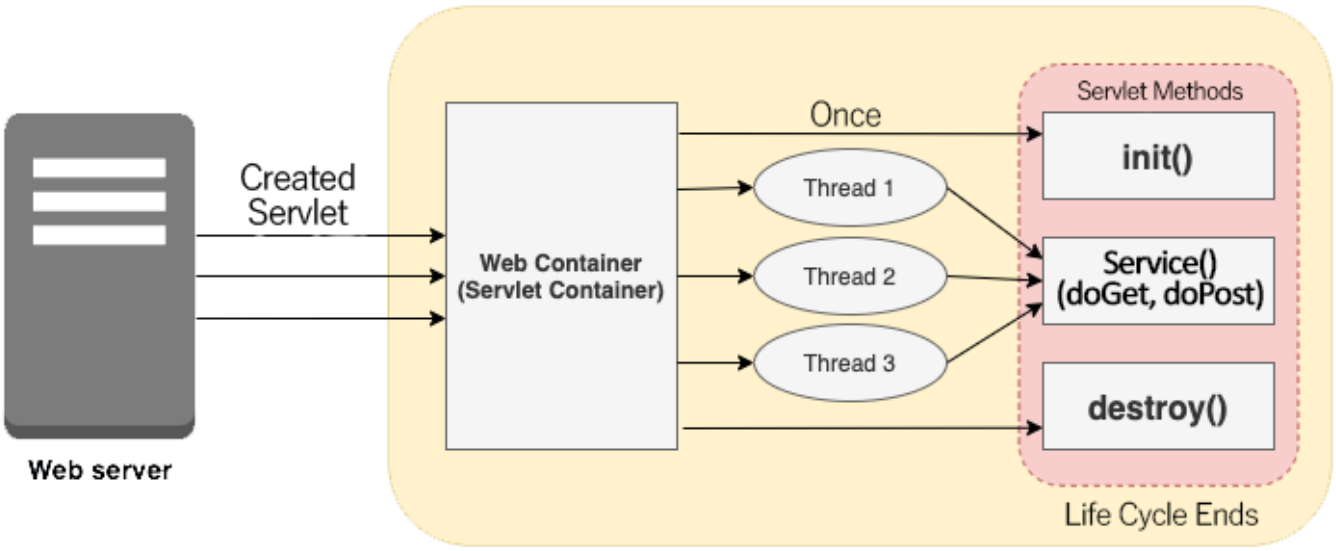
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Logged Out</title>
</head>
<body>
	You have Successfully Logged Out
	<br>
	<br>
	<a href='Login.jsp' ><input type='submit' value='Login'></a>
</body>
</html>
```



After successful Logout, the logout message "You have successfully logged out" is displayed, & below Login button is given to log back in.

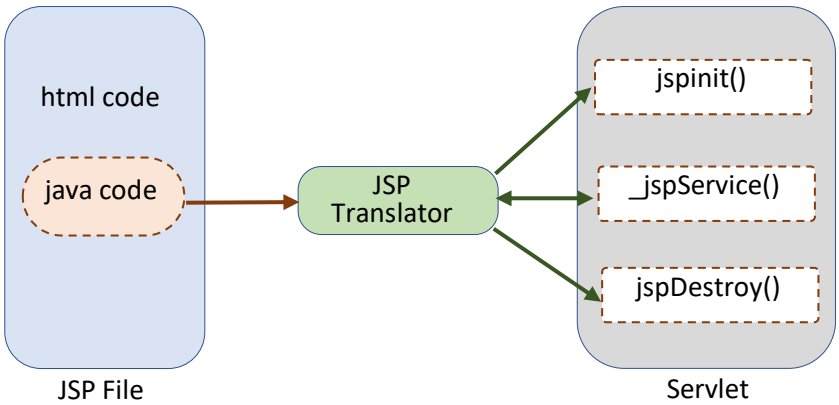
08/07/2022 (Friday)

SERVLET LIFE CYCLE



For the first time when we start Apache Tomcat® Servlet gets loaded into the Tomcat Container, Tomcat Container is an area where logical implementation is been developed to run the Servlet, when the Tomcat gets started, very first method that will run is init() method, and this method runs only once, after init() method we can call the service() methods several times, the service methods are doPost and doGet and we call that any number of times depending upon the business requirements, finally when the destroy method of servlet gets executed Servlet life cycle comes to an end.

JSP Life Cycle



Whatever java code we write in .jsp file, with the help of JSP Translator, that java code is converted into servlet, this servlet have 3 methods jspinit(), _jspService() and jspDestroy().
Jspinit() method will run only once after Tomcat is started but the _jspService() method can run any number of times depending upon business requirements, when the jspDestroy() method runs, it means the life cycle of the .jsp comes to an end.

DIFFERENCE BETWEEN doGet() and doPost() METHODS

	doPost()		doGet()
1.	Only submits data to the database/backend	1.	Only sends data from database/backend to frontend
2.	When refreshing page, it gives alert box to avoid duplicate transactions	2.	When refreshing page, it doesn't give alert box to avoid duplicate transactions
3.	Highly secured	3.	Less secured
4.	Data not exposed in URL	4.	Data is Exposed in URL

Task Assignment

Watch recorded lectures of
"Agile" and **"SQL"**
from "Purchased Courses" section
in Pankaj Sir Academy App.

12/07/2022 (Tuesday)

SERVERS

If we want to build dynamic applications using technologies like SpringBoot, Hibernate, Servlets, JSP, then we use Application Servers like Tomcat, JBoss, Glass Fish etc...

Apache Tomcat

GlassFish

JBoss by Red Hat

IBM WebSphere

ORACLE Weblogic Server

But If we want to build Static Application which doesn't have Login Logout Page, No database, No Servlets, No JSP required & want to use just html, java script, CSS like basics where no backend codes required, then we use Web Servers like NGiNX, LiteSpeed, Apache HTTP Server etc...

NGINX

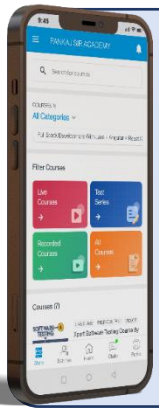
APACHE HTTP SERVER PROJECT

LiteSpeed Web Server

LIGHTTPD fly light.

SOME THEORY QUESTIONS

Q . What are Session Variables?
Q . What is Inter-Servlet Communication?
Q . What is Request Dispatcher?
Q . How do you read form data in Servlets?
Q . What is Session TimeOut?



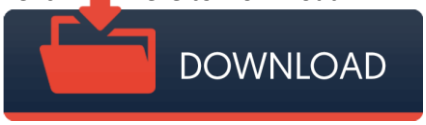
Task Assignment

Watch recorded lecture of "Session TimeOut & Prepared Statement" from "Recorded Series Part-II", Date: 10-12-2020, Duration: 1:04:08 & Next Video also in PSA App

13/07/2022 (Wednesday)

SPRING TOOL SUIT

Click here to Download



Description on Website: *Spring Tools 4 is the next generation of Spring tooling for your favorite coding environment. Largely rebuilt from scratch, it provides world-class support for developing Spring-based enterprise applications, whether you prefer Eclipse, Visual Studio Code, or Theia IDE.*

Click the above Download button to Download Spring Tool Suit 4.8.1 (For Windows)

Maven helps us to download Project dependencies, To download Maven Dependencies Click Here to go to Maven Repository.

Copy the required File's Maven Code and paste into the project's "pom.xml" file under `<dependencies>` tag, The file will automatically be downloaded.

J-UNIT TESTING

JUnit is a unit testing open-source framework for the Java programming language. Java Developers use this framework to write and execute automated tests. In Java, there are test cases that have to be re-executed every time a new code is added. This is done to make sure that nothing in the code is broken. Junit Test is also called as "White Box Testing", "Glass Box Testing", "Transparent Testing", "Open Box Testing".

Annotations for Junit testing:

1. @Test: It is used to specify the test method.
2. @BeforeAll: It is used to specify that method will be called only once, before starting all the test cases.
3. @AfterAll: It is used to specify that method will be called only once, after finishing all the test cases.
4. @BeforeEach: It is used to specify that method will be called before each test case.
5. @AfterEach: It is used to specify that method will be called after each test case.
6. @Ignore: It is used to ignore the test case.

Example:

```
package p1;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class A {
```

```
	@Test
	public void test1() {
		System.out.println("From Test 1");
	}

	@Test
	public void test2() {
		System.out.println("From Test 2");
	}

	@BeforeEach
	public void beforeTest() {
		System.out.println("Before Test");
	}

	@AfterEach
	public void afterTest() {
		System.out.println("AfterTest");
	}

	@BeforeAll //Static Method
	public static void beforeAll() {
		System.out.println("Before All");
	}

	@AfterAll //Static Method
	public static void afterAll() {
		System.out.println("After All");

	}
}
```

Output:

```
Before All
Before Test
From Test 1
AfterTest
Before Test
From Test 2
AfterTest
After All
```

ASSERT CLASS

JUnit provides the Assert class to check the certain conditions. Assert class methods compare the output value to the expected value.

Example #1: *(Class Add)*

```
package p1;
public class Add {
	public int addNumbers (int x, int y) {
		return x+y;
	}
}
```

(testAddClass)

```
package p1;
import org.junit.Test;
import junit.framework.Assert;
public class TestClass {

	@Test
	public void testAddClass() {
		Add a = new Add();
		int actualResullt = a.addNumbers(10, 20);
		int expectedResult = 30;

		Assert.assertEquals(expectedResult, actualResullt);
		//The above method is deprecated (outdated) but still it will work
	}
}
```

JUnit Output:

Example #2: *(testAddClass)*

```
package p1;
import org.junit.Test;
import junit.framework.Assert;
public class TestClass {

	@Test
	public void testAddClass() {
		Add a = new Add();
		int actualResullt = a.addNumbers(10, 20);
		int expectedResult = 60;

		Assert.assertEquals(expectedResult, actualResullt);
		//The above method compared the Expected and Actual but the result was different
	}
}
```

JUnit Output:
junit.framework.AssertionFailedError: expected:<**60**> but was:<**30**>

14/07/2022 (Thursday)

JPA (JAVA PERSISTENCE API)

JPA (Java Persistence API) is also called as "Jakarta Persistence API", The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems.

As JPA is just a specification, we don't write any SQL Query here, ORM (Object Relational Mapping) tools like Hibernate, TopLink and iBatis does itself.

Example: Let's create a table in database with name "Student".

Table Name : Student

Id	Name	Course	Fee

Create a class with same name as table in DB, & create variable equal to the no. of columns in that table & make getters & setters for that variables. This class is called as "Entity Class".

Now create the object of the entity class and put all the data in that object like id, name, course and fee, now JPA takes the data of the object and puts it into the table in database & the ORM which implements this in is called Hibernate (We're using Hibernate).

Q . What is Hibernate?

A. Hibernate is an ORM (Object Relational Mapping) tool. JPA defines the concept of ORM, but implementation of that is done in Hibernate, Eclipse Link etc..

As a beginner when working on Spring Boot Projects, follow the below steps:

1. Create Database Schema (Tables)
2. Create Entity Class
3. Configure "application.properties" file (with DB details)
4. Create Repository Layer (Interface)
5. Perform JUnit Testing

The first 3 steps belongs to Hibernate *(anything imported from "javax.persistence" package belongs to Hibernate)*, 4th Step belong to Spring Boot *(Repository layer has lot of built-in methods using which we can perform CRUD Operations with DB)*.

Key points to remember in Spring Boot:

- Object is called as "bean" in Spring Boot
- @Autowired annotation is used for creating beans (Objects) in Spring Boot, it's called as "DI" (Dependency injection), injecting the object address into reference variable.

Example: Let's create a project named "demo_crud" to perform CRUD Operations on Student's data.

Step #1: Create Database Schema (Tables)

```
create database demo_crud
use demo_crud
create table student
(
ID int primary key auto_increment,
Name varchar (45),
Course varchar (45),
Fee int
)

select * from student
```

Here primary key means ID should be unique and auto_increment declares value should be increased automatically.

Step #2: Create Entity Class

```
package com.demo_crud.entities;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity //This annotation describes that this is a Special class whose object data will be stored in DB table, Entity=Table in DB
public class Student {
	//variable name should match with the column names in DB
	@Id //It declares that this is a Primary Key
	@GeneratedValue(strategy = GenerationType.IDENTITY) //It declares that the Id has auto increment in its values
	private long id; //Using Long" instead of "int" here because, Long can store more data than int
	private String name;
	private String course;
	private int fee;

		public long getId() {
			return id;
		}
		public void setId(long id) {
			this.id = id;
		}
		public String getName() {
			return name;
		}
		public void setName(String name) {
			this.name = name;
		}
		public String getCourse() {
			return course;
		}
```

```
	public void setCourse(String course) {
		this.course = course;
	}
	public int getFee() {
		return fee;
	}
	public void setFee(int fee) {
		this.fee = fee;
	}
}
```

Step #3: Configure "application.properties" file (with DB details)

```
spring.datasource.url=jdbc:mysql://localhost:3306/demo_crud
spring.datasource.username=root
spring.datasource.password=Test
```

Step #4: Create Repository Layer (Interface)

```
package com.demo_crud.repository;
import org.springframework.data.repository.CrudRepository;
import com.demo_crud.entities.Student;

public interface StudentRepository extends CrudRepository<Student, Long> {
	//Perform CRUD Operations on "Student" Class and Primary key wrapper class name, it wont take "long" so apply wrapper class and put "Long"
}
```

Step #4: Perform JUnit Testing

```
package com.demo_crud;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.demo_crud.entities.Student;
import com.demo_crud.repository.StudentRepository;

@SpringBootTest
class DemoCrudApplicationTests {

	@Autowired //Dependency Injection
	private StudentRepository studentRepo;  //Ref. variable of Interface

	@Test
	void saveOneStudent() {
		Student s1 = new Student();
		s1.setName("Mike");
		s1.setCourse("Development");
		s1.setFee(1000);

		studentRepo.save(s1);
	}
}
```

Now we have successfully saved the data of one student in MySQL DB without writing Query.

```
11      select * from student
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	Name	Course	Fee
1	Mike	Development	1000
NULL	NULL	NULL	NULL

Let's save one more student's data.

```
package com.demo_crud;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.demo_crud.entities.Student;
import com.demo_crud.repository.StudentRepository;

@SpringBootTest
class DemoCrudApplicationTests {

	@Autowired //Dependency Injection
	private StudentRepository studentRepo;  //Ref. variable of Interface

	@Test
	void saveOneStudent() {
		Student s1 = new Student();
		s1.setName("Stallin");
		s1.setCourse("Testing");
		s1.setFee(800);

		studentRepo.save(s1);
	}
}
```



```
11      select * from student
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	Name	Course	Fee
1	Mike	Development	1000
2	Stallin	Testing	800
NULL	NULL	NULL	NULL

We can also delete the data of any student using ID#, but the deleted ID# will not be assigned to the next record, once deleted, permanently deleted, because we're using primary key for IDs, so every entry has unique ID#, so the next save method will insert ID#3 after deleting Stallin's record which is assigned ID#2.

```java
package com.demo_crud;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.demo_crud.entities.Student;
import com.demo_crud.repository.StudentRepository;

@SpringBootTest
class DemoCrudApplicationTests {

	@Autowired //Dependency Injection
	private StudentRepository studentRepo;  //Ref. variable of Interface

	@Test
	void saveOneStudent() {
		Student s1 = new Student();
		s1.setName("Stallin");
		s1.setCourse("Testing");
		s1.setFee(800);

		studentRepo.save(s1);
	}

	@Test
	void deleteOneStudent() {
		studentRepo.deleteById(2L); //Deleting ID#2 (Stallin's Record)
		//Just select delete method name and run, if we run directly the above save
method will also run and again same data will be saved with next ID number
	}
}
```

```
11	select * from student
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	Name	Course	Fee
1	Mike	Development	1000
NULL	NULL	NULL	NULL

15/07/2022 (Friday)

Before going forward in Student Project's CRUD Operations, let's learn about Optional Class.

OPTIONAL CLASS

Java 8 introduced a new public final class "Optional" in java.util package. It is used to deal with NullPointerException in java application. It provides the methods to easily check whether a variable has null value or not, this is an alternative way of handling NullPointerException other than Try Catch Block, the commonly used methods of Java Optional class are:

Optional.ofNullable(): It returns a Non-empty Optional if the given object has a value, otherwiseit returns an empty Optional.

isPresent(): It is used check whether the particular Optional object is empty or no-empty.

ifPresent(): It only executes if the given Optional object is non-empty.

Example: *(with NullPointerException)*

```
package p1;
public class A {
	int x =10;
	static A a1; //a1 is null reference variable
	public static void main(String[] args) {
		System.out.println(a1.x);
	}
}
```

Output:

```
Exception in thread "main" java.lang.NullPointerException
	at p1.A.main(A.java:6)
```

(with Optional Class)

```
package p1;
import java.util.Optional;
public class A {
	int x =10;
	static A a1; //a1 is null reference variable
	public static void main(String[] args) {
		Optional<A> val = Optional.ofNullable(a1);
		System.out.println(val);
		System.out.println(val.isPresent()); //if value present true, otherwise false
	}
}
```

Output:

```
Optional.empty
false
```

(Applying if condition, If value is present, it prints value, otherwise else part)

```
package p1;
import java.util.Optional;
public class A {
	int x =10;
	static A a1 = new A(); //a1 is not null
	public static void main(String[] args) {
		Optional<A> val = Optional.ofNullable(a1);
		if(val.isPresent()) {
			System.out.println(a1.x);
		}else {
			System.out.println("No Value Present");
		}
	}
}
```

Output:

```
10
```

(here a1 is null, so else part will run)

```
package p1;
import java.util.Optional;
public class A {
	int x =10;
	static A a1; //a1 is null
	public static void main(String[] args) {
		Optional<A> val = Optional.ofNullable(a1);
		if(val.isPresent()) {
			System.out.println(a1.x);
		}else {
			System.out.println("No Value Present");
		}
	}
}
```

Output:

```
No Value Present
```

JPA (JAVA PERSISTENCE API) (Contd...)

Now let's Read one student data by ID#1 (ID#1 is present in database)

```
package com.demo_crud;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.demo_crud.entities.Student;
import com.demo_crud.repository.StudentRepository;

@SpringBootTest
class DemoCrudApplicationTests {

	@Autowired //Dependency Injection
	private StudentRepository studentRepo;  //Ref. variable of Interface

	@Test
	void saveOneStudent() {
		Student s1 = new Student();
		s1.setName("Sameer");
		s1.setCourse("Development");
		s1.setFee(5000);
		studentRepo.save(s1);
	}

	@Test
	void deleteOneStudent() {
		studentRepo.deleteById(2L); //Deleting ID#2 (Stallin's Record)
	}

	@Test
	void getOneStudent() {
		Optional<Student> findById = studentRepo.findById(1L); //ID#1 is present
		if(findById.isPresent()){ //it will put the return data in optional object
			Student student = findById.get();
			//"get" will convert it to student(entity) object
			System.out.println(student.getId());
			System.out.println(student.getName());
			System.out.println(student.getCourse());
			System.out.println(student.getFee());
		}else {
			System.out.println("No Record Found");
		}
	}
}
```

Output:

```
1
Mike
Development
1000
```

Let's find by ID#10 (ID#10 is not present in database

```
package com.demo_crud;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.demo_crud.entities.Student;
import com.demo_crud.repository.StudentRepository;

@SpringBootTest
class DemoCrudApplicationTests {

	@Autowired //Dependency Injection
	private StudentRepository studentRepo;  //Ref. variable of Interface

	@Test
	void saveOneStudent() {
		Student s1 = new Student();
		s1.setName("Sameer");
		s1.setCourse("Development");
		s1.setFee(5000);
		studentRepo.save(s1);
	}

	@Test
	void deleteOneStudent() {
		studentRepo.deleteById(2L); //Deleting ID#2 (Stallin's Record)
	}

	@Test
	void getOneStudent() {
		Optional<Student> findById = studentRepo.findById(10L); //ID#10 isn't present
		if(findById.isPresent()){ //it will put the return data in optional object
			Student student = findById.get();
			//"get" will convert it to student(entity) object
			System.out.println(student.getId());
			System.out.println(student.getName());
			System.out.println(student.getCourse());
			System.out.println(student.getFee());
		}else {
			System.out.println("No Record Found");
		}
	}
}
```

Output:

```
No Record Found
```

If we wouldn't have used Optional Class here, as there was no data with ID#10 was present in DB, it would have resulted "NullPointerException", because findById is null, & with null ref variable if we call get method it is "NullPointerException".

Now let's update the Record. *Updating Mike's (ID#1) data.*
Database before updating:

```
11    select * from student
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	Name	Course	Fee
1	Mike	Development	1000
3	Stallin	Testing	800
4	Sameer	Development	5000
NULL	NULL	NULL	NULL

Example: *(Mike wants to change the Course from development to Testing)*

```
package com.demo_crud;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.demo_crud.entities.Student;
import com.demo_crud.repository.StudentRepository;

@SpringBootTest
class DemoCrudApplicationTests {

	@Autowired //Dependency Injection
	private StudentRepository studentRepo;  //Ref. variable of Interface

	@Test
	void updateOneStudent() { //Updating ID#1
		Optional<Student> findById = studentRepo.findById(1L);
		if(findById.isPresent()){
			Student student = findById.get();
			student.setCourse("Testing");
			student.setFee(800);
			studentRepo.save(student);
		}else {
			System.out.println("No Record Found");
		}
	}
}
```

Mike's data has been successfully updated, database:

```
11    select * from student
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	Name	Course	Fee
1	Mike	Testing	800
3	Stallin	Testing	800
4	Sameer	Development	5000
NULL	NULL	NULL	NULL

Let's find All Student data.

```
package com.demo_crud;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.demo_crud.entities.Student;
import com.demo_crud.repository.StudentRepository;

@SpringBootTest
class DemoCrudApplicationTests {

	@Autowired //Dependency Injection
	private StudentRepository studentRepo;  //Ref. variable of Interface

	@Test
	void getAllStudent() {
	Iterable<Student> findAll = studentRepo.findAll();
	 for (Student s : findAll) {
	 System.out.println(s.getId()+" "+s.getName()+" "+s.getCourse()+" "+s.getFee());
	 }
    }
}
```

Output:

```
1 Mike Testing 800
3 Stallin Testing 800
4 Sameer Development 5000
```

Assignment

Do the research and write the above findAll method and iterate using **Lambda Expression**.

Answer:

```
@Test
void getAllStudent() {
	Iterable<Student> findAll = studentRepo.findAll();
	findAll.forEach( (s)  -> { //Using Lambda Expression
	System.out.println(s.getId()+" "+s.getName()+" "+s.getCourse()+" "+s.getFee()); });
}
```

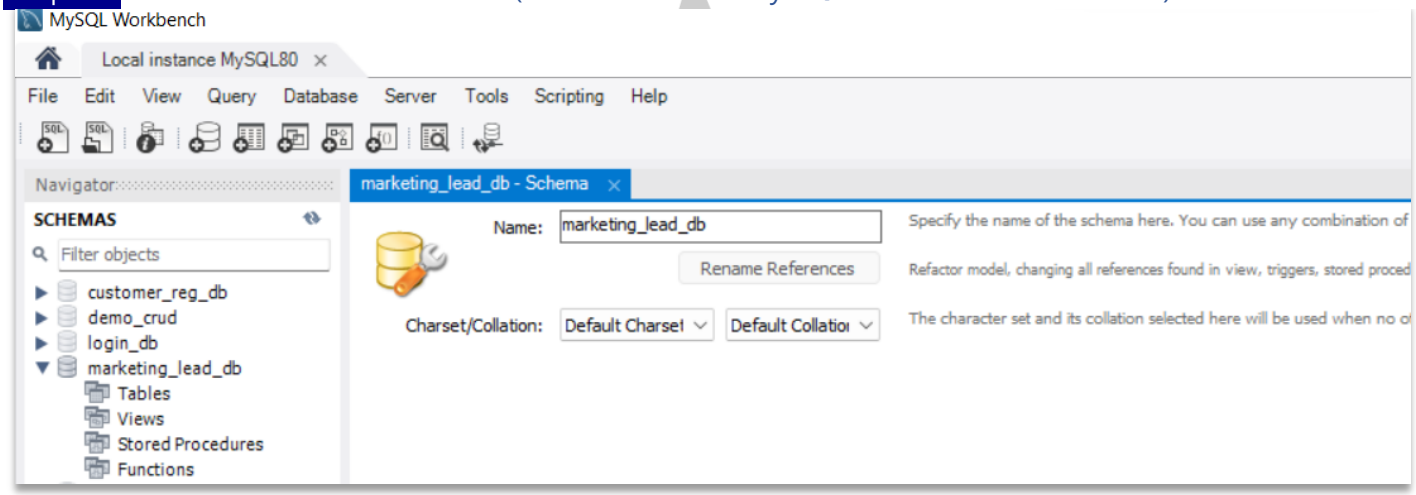
18/07/2022 (Monday)

WEB DEVELOPMENT (PROJECT)

Here we are going to develop a project for "Marketing Lead", *("Lead" is a marketing terminology, A potential customer enquiring about the product is called as lead).*
For this project on Spring Boot, we are following the below steps:

1. Create Database Schema (Tables) using JAVA Code
2. Create Entity Class
3. Configure "application.properties" file (with DB details)
4. Create Repository Layer (Interface)
5. Configure Jasper Tomcat
6. View Layer
7. Controller Layer
8. Service Layer

Step #1: Create Database Schema (Local instance MySQL>Create new Schema)



Step #2: Create Entity Class

```
package com.marketing.entities;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "leads") //Annotation should be used when table name needed in DB is dif-
ferent than class name
public class Lead {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY) //primary & auto increment
	private long id;
```

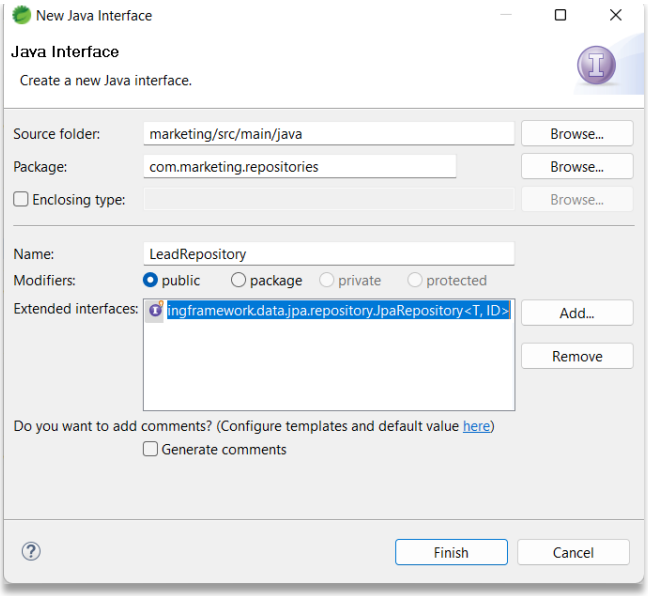
```
	@Column(name = "first_name", nullable=false)
	//Annotation should be used when column name needed in DB is different than method
	name, it shouldn't have null value
	private String firstName;


	@Column(name = "last_name", nullable=false)
	private String lastName;

	@Column(nullable=false, unique=true) //no null value & should be unique, to avoid
	duplicate data in database
	private String email;

	@Column(nullable=false, unique=true) //nullable=false means the data is mandatory
	private long mobile;


	public long getId() {
		return id;
	}

	public void setId(long id) {
		this.id = id;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public long getMobile() {
		return mobile;
	}

	public void setMobile(long mobile) {
		this.mobile = mobile;
	}
}
```

Step #3: Configure "application.properties" file (with DB details)

```
spring.datasource.url=jdbc:mysql://localhost:3306/marketing_lead_db
spring.datasource.username=root
spring.datasource.password=Test

#This line helps to create table in database using entity class,
#After running once on "create" don't forget to change it to "update".
#Otherwise on every server restart the table will be dropped off
spring.jpa.hibernate.ddl-auto=update

#Path of the jsp file and its extension
spring.mvc.view.prefix=/WEB-INF/jsps/
spring.mvc.view.suffix=.jsp
```

Step #4: Create Repository Layer (Interface)

Created the Repository Layer (interface) with extended interface "JpaRepository" instead of "CurdRepository" because it's an update of CRUD & can perform more operations than CrudRepository.

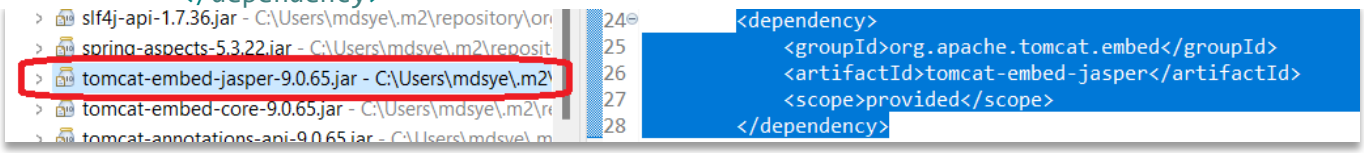



```
package com.marketing.repositories;
import org.springframework.data.jpa.repository.JpaRepository;
import com.marketing.entities.Lead;
public interface LeadRepository extends JpaRepository<Lead, Long> { //Lead is Entity Class
}
```

Step #5: Configure Jasper Tomcat

Dependency tag to configure Jasper Tomcat:

```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>
```



18/07/2022 (Monday)

Step #5: View Layers (Lead)

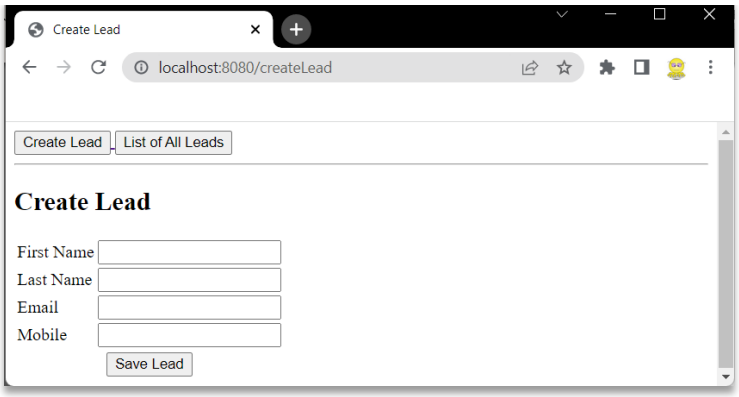
Create folders manually in src>main>webapp > WEB-INF > jsps.

JSP Page (View Layer) "create_lead.jsp":

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Menu.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Create Lead </title>
</head>
<body>
<h2> Create Lead </h2>
<form action="saveLead" method="post">
<table>
<tr><td>First Name</td><td><input type="text" name="firstName"/></td></tr>
<tr><td>Last Name</td><td><input type="text" name="lastName"/></td></tr>
<tr><td>Email</td><td><input type="text" name="email"/></td></tr>
<tr><td>Mobile</td><td><input type="text" name="mobile"/></td></tr>
<tr></tr>
<tr><td style="text-align:center" colspan="2"><input type="submit" value="Save Lead"/></td></tr>
</table>

</form>
${msg}
</body>
</html>
```

JSP Page (View Layer) "Menu.jsp":

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
</head>
<body>
      <a href="createLead"> <input type="button" value="Create Lead"> </a>
      <a href="listAll"> <input type="button" value="List of All Leads"> </a>
      <hr>
</body>
</html>
```



19/07/2022 (Tuesday)

After creating "create_lead.jsp" and "Menu.jsp" let's go ahead with Controller Layers.

Step #6: Controller Layers

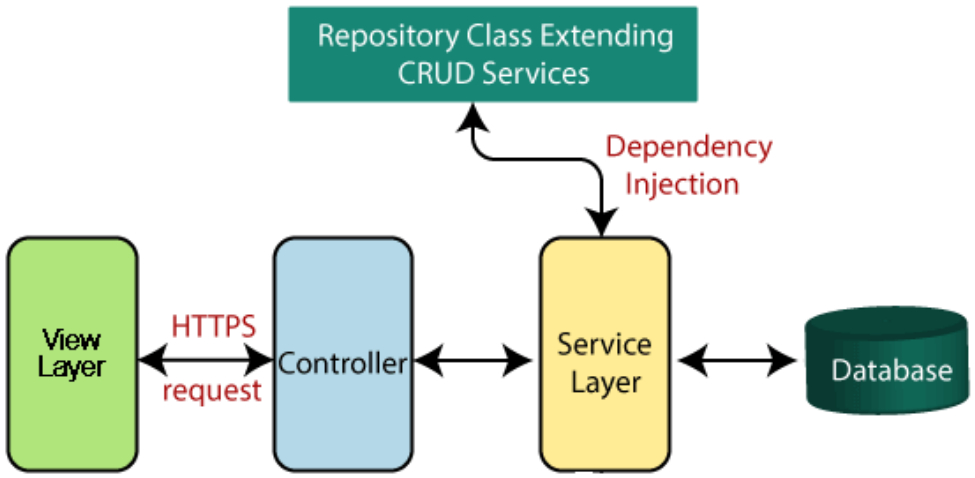
Lead Controller Page (Controller Layer) "LeadController.java":

```
package com.marketing.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.marketing.dto.LeadData;
import com.marketing.entities.Lead;
import com.marketing.services.LeadService;

@Controller
public class LeadController {

	@Autowired //Dependency Injection
	private LeadService leadService;

	//handler method, To navigate to "create_lead.jsp", because only controller can access this
	@RequestMapping("/createLead") //acts like @WebServlet
	public String viewCreateLeadPage() {
		return "create_lead"; //acts like RequestDispatcher
		//path of this jsp file given in "application.properties"
	}
```


In this Spring Boot project also, we follow MVC Architecture.

Q. Explain the technical flow of your project?

A.



20/07/2022 (Wednesday)

Step #7: Services Layers

Service Layer: Interface (LeadService.java)

```
package com.marketing.services;
import java.util.List;
import com.marketing.entities.Lead;

public interface LeadService {
	public void saveLead(Lead l);
	public List<Lead> listLeads();
	public void deleteLeadById(long id);
	public Lead getOneLead(long id);
	}
```

Service Layer: Class (LeadServiceImpl.java)

```
package com.marketing.services;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.marketing.entities.Lead;
import com.marketing.repositories.LeadRepository;

@Service
public class LeadServiceImpl implements LeadService {

	@Autowired //Dependency Injection
	private LeadRepository leadRepo;

	@Override
	public void saveLead(Lead l) {
		leadRepo.save(l); //To save data in to the database
	}

	@Override
	public List<Lead> listLeads() {
		List<Lead> leads = leadRepo.findAll(); //To read data in the database
		return leads;
	}

	@Override
	public void deleteLeadById(long id) {
		leadRepo.deleteById(id); //To delete data from the database
	}

	@Override
	public Lead getOneLead(long id) {
		Optional<Lead> findById = leadRepo.findById(id); //findrecord by ID# in the database
		Lead lead = findById.get();
		return lead;
	}
}
```

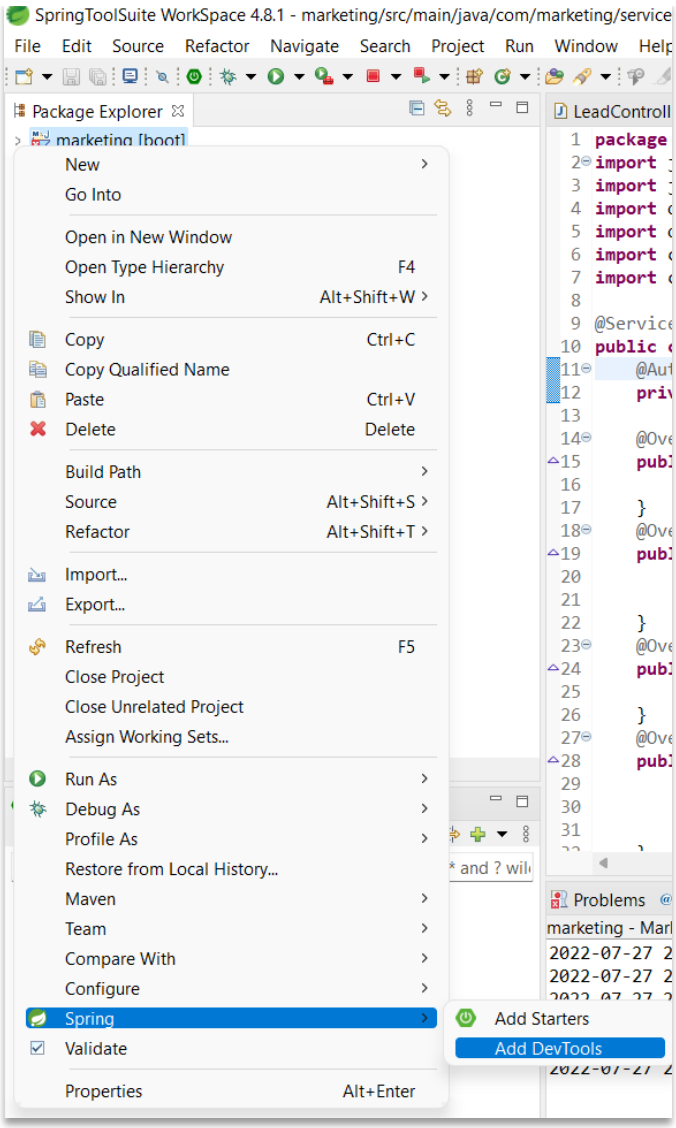
In Create Lead JSP Page, after putting all the form details, clicking on the "Save Lead" button will call `@RequestMapping("/saveLead")` from "LeadController.java".

There are multiple ways to transfer object data from "View Layer" to database.

Method #1: *To Save data from View Layer to the database (LeadController.java)*

```
//Method #1
@RequestMapping("/saveLead")
public String saveOneLead(@ModelAttribute("lead") Lead l, ModelMap model) {
//ModelAttribute("lead")-"lead" object stores the data & ref variable is "l"
//"Lead" is a entity class and "l" is ref. variable
// ModelMap is same as "Request.set & getAttribute" to put back message in View Layer
	leadService.saveLead(l);
	model.addAttribute("msg", "Lead Saved Successfully");
	return "create_lead";
```

Note: Every time restarting the servers manually is painful, let's make it automatic so that whenever we make any changes to our project, server should automatically restart (Live Loading). To do that Right Click on the project, & in cascade menu select Spring>Add DevTools.



21/07/2022 (Thursday)

Method #2: *To Save data from View Layer to the database (LeadController.java)*

```
//Method#2: it will make the method argument lengthy if working with huge no. of fields, Recommended for less no. of fields
	@RequestMapping("/saveLead")
	public String saveOneLead(@RequestParam("name") String fName, @RequestParam("LastName") String lName, @RequestParam("emailId") String mail,  @RequestParam("mobileNumber") long mobile) {
		Lead l = new Lead(); //lead Object is needed to save data in DB
		l.setFirstName(fName);
		l.setLastName(lName);
		l.setEmail(mail);
		l.setMobile(mobile);
		leadService.saveLead(l);
		return "create_lead";
		}
```

JSP Page of Method #2:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Create Lead 2 </title>
</head>
<body>
	<h2> Registration | Create </h2>
	<form action="saveLead" method="post">
		<pre>
	<!-- Method#2: In name attribute we're giving different names other than entity class   -->
		First Name <input type="text" name="name"/>
		Last Name <input type="text" name="LastName"/>
		Email <input type="text" name="emailId"/>
		Mobile <input type="text" name="mobileNumber"/>
		<input type="submit" value="Save Lead"/>
		</pre>
	</form>
</body>
</html>
```

Method #3: *To Save data from View Layer to the database (LeadController.java)*

```
//Method#3: DTO- Data Transfer Object
	@RequestMapping("/saveLead")
	public String saveOneLead(LeadData data, ModelMap model) {
	//Press Ctr+1, Create "LeadData" class in dto package
		Lead ld = new Lead(); //lead Object is needed to save data in DB
		ld.setFirstName(data.getFirstName());
		ld.setLastName(data.getLastName());
		ld.setEmail(data.getEmail());
		ld.setMobile(data.getMobile());

		leadService.saveLead(ld);
		model.addAttribute("msg", "Lead Saved Successfully");
		return "create_lead";
	}
```

LeadData Class for Method #3 in DTO Package:

```
package com.marketing.dto;
public class LeadData { //This is an ordinary JAVA Class
	//Create variables matching to .jsp file from attribute names
	private long id;
	private String firstName;
	private String lastName;
	private String email;
	private long mobile;
	//Getters and Setters
	public long getId() { return id; }
	public void setId(long id) { this.id = id; }
	public String getFirstName() { return firstName; }
	public void setFirstName(String firstName) { this.firstName = firstName; }
	public String getLastName() { return lastName; }
	public void setLastName(String lastName) { this.lastName = lastName; }
	public String getEmail() { return email; }
	public void setEmail(String email) { this.email = email; }
	public long getMobile() { return mobile; }
	public void setMobile(long phone) { this.mobile = phone; }
}
```

JSP Page of Method #3:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Create Lead </title>
</head>
<body>
	<h2> Registration | Create </h2>
	<form action="saveLead" method="post">
		<pre>
		First Name <input type="text" name="firstName"/>
		<!-- name attribute should exactly match with variable name in DTO class   -->
		Last Name <input type="text" name="lastName"/>
		Email <input type="text" name="email"/>
		Mobile <input type="text" name="phone"/>
		<input type="submit" value="Save Lead"/>
		</pre>
	</form>
	${msg}
</body>
</html>
```

Database:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	email	first_name	last_name	mobile
1	mdsyedaalam@gmail.com	Md Syed	Aalam	8801114535
2	pankaj@pankajsiracademy.com	Pankaj p	Mutha	9632629045
4	johndeo@gmail.com	John	Deo	1144553700
5	mike@gmail.com	Mike	Tyson	8585746585
6	stallin@gmail.com	Stallin	S	7878787878
NULL	NULL	NULL	NULL	NULL

22/07/2022 (Friday)

Before going further in project let's first look at "JSTL"

JSTL (Jakarta Standard Tag Library)

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

Advantage of JSTL:

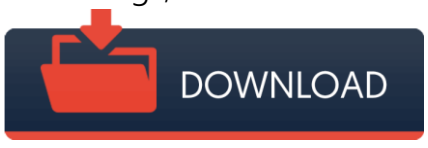
- Fast Development JSTL provides many tags that simplify the JSP.
- Code Reusability We can use the JSTL tags on various pages.
- No need to use scriptlet tag, It avoids the use of scriptlet tag.
- We can write JAVA code with JSTL tags.

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.

Source

The JSTL.jar file is needed run the JSTL Tags, Click below to download JSTL_1.2.jar



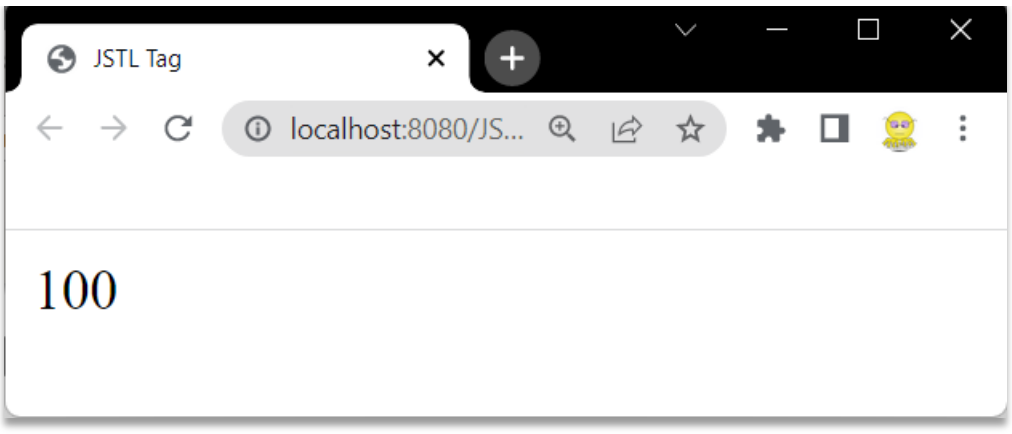
After downloading paste the .jar file into scr>main>webapp>WEB-INF>lib.

JSTL Core Tags:

JSTL Core Tag	Description
<c:out>	To write something in JSP page, we can use EL also with this tag
<c:import>	Same as <jsp:include> or include directive
<c:redirect>	redirect request to another resource
<c:set>	To set the variable value in given scope.
<c:remove>	To remove the variable from given scope
<c:catch>	To catch the exception and wrap it into an object.
<c:if>	Simple conditional logic, used with EL and we can use it to process the exception from <c:catch>
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <c:when> and <c:otherwise>
<c:when>	Subtag of <c:choose> that includes its body if its condition evalutes to 'true'.
<c:otherwise>	Subtag of <c:choose> that includes its body if its condition evalutes to 'false'.
<c:forEach>	for iteration over a collection
<c:forTokens>	for iteration over tokens separated by a delimiter.
<c:param>	used with <c:import> to pass parameters
<c:url>	to create a URL with optional query string parameters

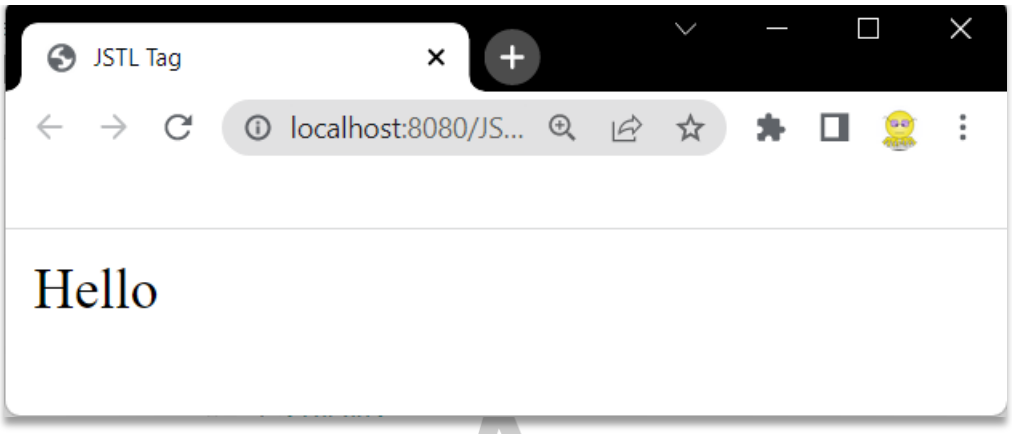
Example #1:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core"%> <!-- JSTL Directive Tag -->
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> JSTL Tag </title>
</head>
<body>
      <c:set var="val" value="100"></c:set>
      <c:out value="${val}"></c:out>
</body>
</html>
```



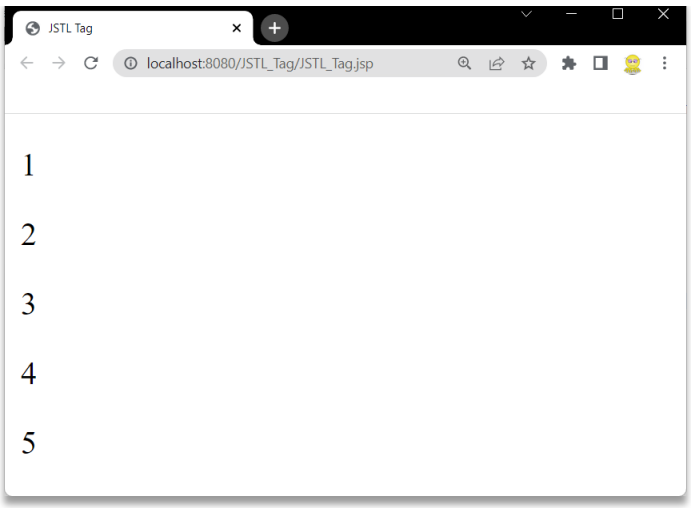
Example #2:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core"%> <!-- JSTL Directive Tag -->
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> JSTL Tag </title>
</head>
<body>
	<c:set var="val" value="Hello"></c:set>
	<c:out value="${val}"></c:out>
</body>
</html>
```



Example #3:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core"%> <!-- JSTL Directive Tag -->
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> JSTL Tag </title>
</head>
<body>
	<c:forEach var="j" begin="1" end="5">
		<p>${j}</p>
	</c:forEach>
</body>
</html>
```



Let's get back to our Web Development Project (Marketing Lead)

WEB DEVELOPMENT PROJECT (contd...)

By clicking on the "list of All Leads" button in the Menu will call `@RequestMapping("/listAll")`

To read all the data from the database. (LeadController.java)

```
@RequestMapping("/listAll")
public String listAllLeads(ModelMap model) {
	List<Lead> leads = leadService.listLeads();
	model.addAttribute("lds", leads);
	return "leadSearchResult";
```

Q. Difference between CRUD Repository and JPA Repository.?

A. In JPA Repository findAll method returns "List" and CRUD Repository returns iterable.

Add the dependency tag to download & use JSTL 1.2 in Spring Boot.

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version><!--$NO-MVN-MAN-VER$-->
</dependency>
```

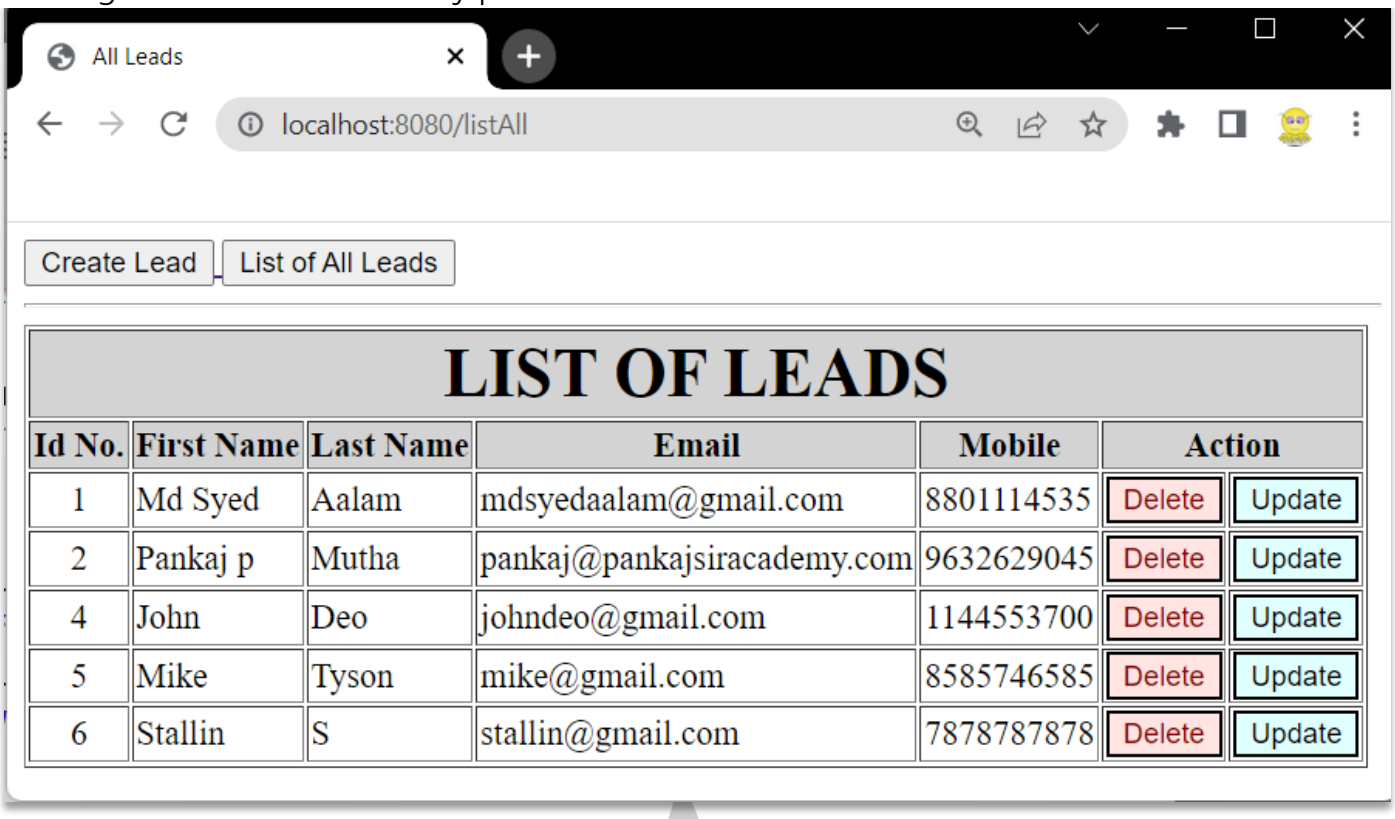
After adding the dependency jar, don't rely on auto start, re-start server manually to configure JSTL jar completely.

"listAll" method from "LeadController.java" will return the "leadSearchResult.jsp".

List of all the Leads JSP Page (leadSearchResult.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Menu.jsp" %>
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core"%> <!-- JSTL Directive Tag -->
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>All Leads</title>
</head>
<body>
<table border=1>
<tr><th colspan="7" bgcolor="lightgrey"><font size="6"> LIST OF LEADS </font></th></tr>
<tr  bgcolor="lightgrey">
<th>Id No.</th><th>First Name</th><th>Last Name</th><th>Email</th><th>Mobile</th><th col-
span="2">Action</th>
</tr>
<c:forEach var="leads" items="${lds}">
<tr><td style="text-align:center"> ${leads.id} </td>
<td> ${leads.firstName} </td>
<td>${leads.lastName}</td>
<td>${leads.email}</td>
<td>${leads.mobile}</td>
<td><a href="delete?id=${leads.id}">
<input style="background-color:mistyrose; color:darkred" type="submit" value="Delete">
</a></td>
<td><a href="update?id=${leads.id}">
<input style="background-color:lightcyan; color:darblue" type="submit" value="Update">
</a></td>
</tr>
</c:forEach>
</table>
</body>
</html>
```

WebPage of "leadSearchResult.jsp":



25/07/2022 (Monday)

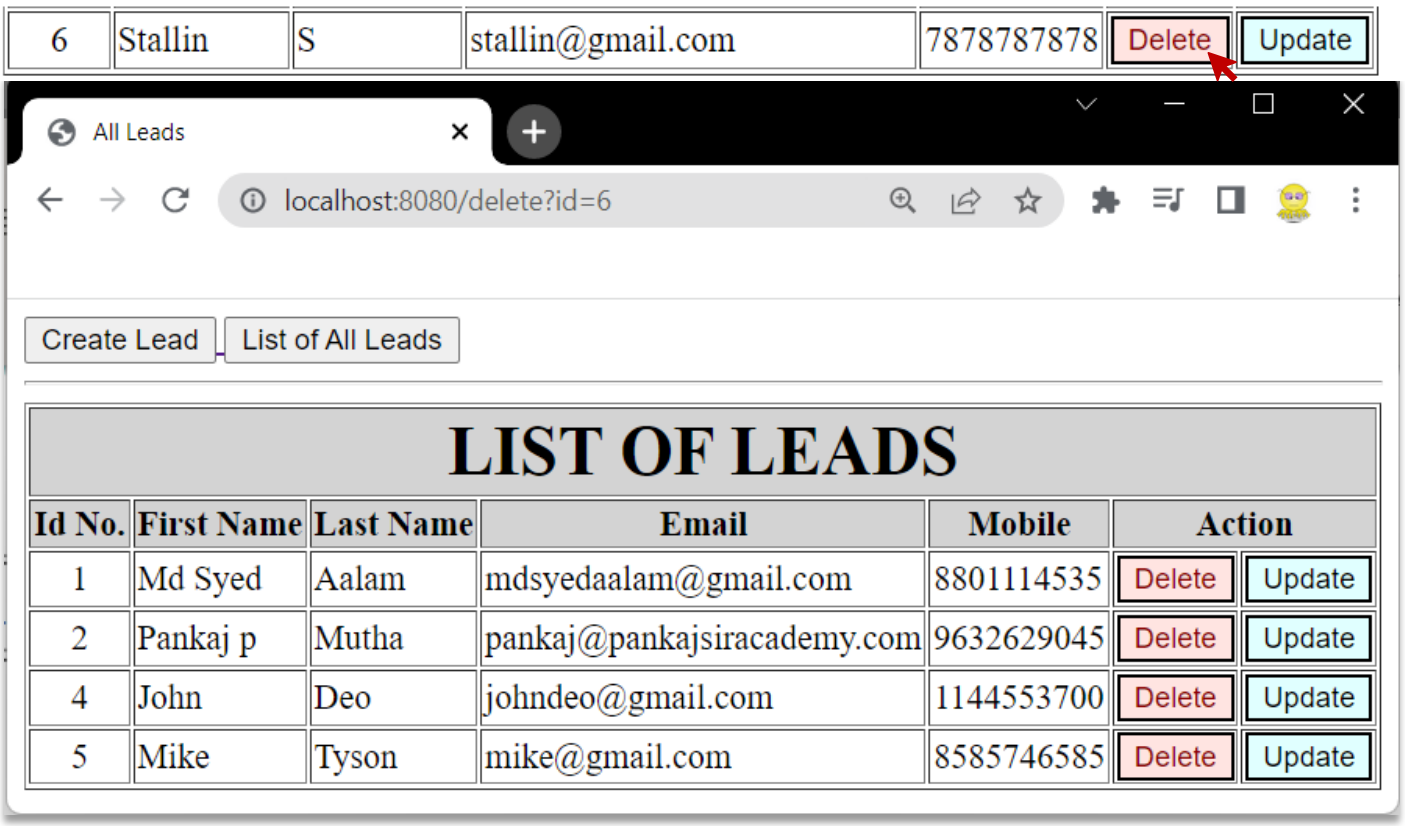
DELETING A RECORD FROM DATABASE

To delete a lead by clicking on the delete button will call `@RequestMapping("/delete")` from "LeadController.java".

To delete the data from the database. (LeadController.java)

```
@RequestMapping("/delete")
public String deleteOneLead(@RequestParam ("id") long id, ModelMap model) {
	leadService.deleteLeadById(id);

	List<Lead> leads = leadService.listLeads();
	model.addAttribute("lds", leads);
	return "leadSearchResult"; //After delete stay on the same page
}
```

Let's delete Stallin's data:



UPDATING A RECORD IN THE DATABASE

To update the record, clicking on the update button will call `@RequestMapping("/update")` from "LeadController.java".

To get the particular id data in the database. (LeadController.java)

```
@RequestMapping("/update")
public String updateOneLead(@RequestParam ("id") long id, ModelMap model) {
	Lead lead = leadService.getOneLead(id);
	model.addAttribute("leadUpd", lead);
	return "updateLead";
	}
```

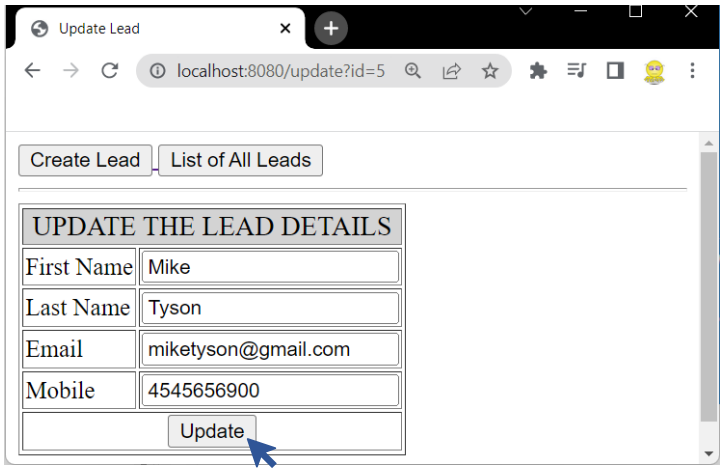
This will redirect to "updateLead.jsp" with that particular record.

JSP Page (updateLead.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Menu.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Update Lead </title>
</head>
<body>
<form action="updateLead" method="post">
<table border=1>
<tr>
	<td style="text-align:center" colspan="2" bgcolor="lightgrey">
	<input type="hidden" name="id" value="${leadUpd.id}"><font size="4"> UPDATE THE LEAD DE-
	TAILS </font></td>
</tr>
<tr>
	<td>First Name</td>
	<td><input type="text" name="firstName" value="${leadUpd.firstName}"/></td>
</tr>
<tr>
	<td>Last Name</td>
	<td><input type="text" name="lastName" value="${leadUpd.lastName}"/></td>
</tr>
<tr>
	<td>Email</td>
	<td><input type="text" name="email" value="${leadUpd.email}"/></td>
</tr>
<tr>
	<td>Mobile</td>
	<td><input type="text" name="mobile" value="${leadUpd.mobile}"/></td>
</tr>
<tr>
	<td style="text-align:center" colspan="2"><input type="submit" value="Update"/></td>
</tr>
</table>
</form>
</body>
</html>
```

Let's update Mike's Email and Mobile Number:

5	Mike	Tyson	mike@gmail.com	8585746585	Delete	Update



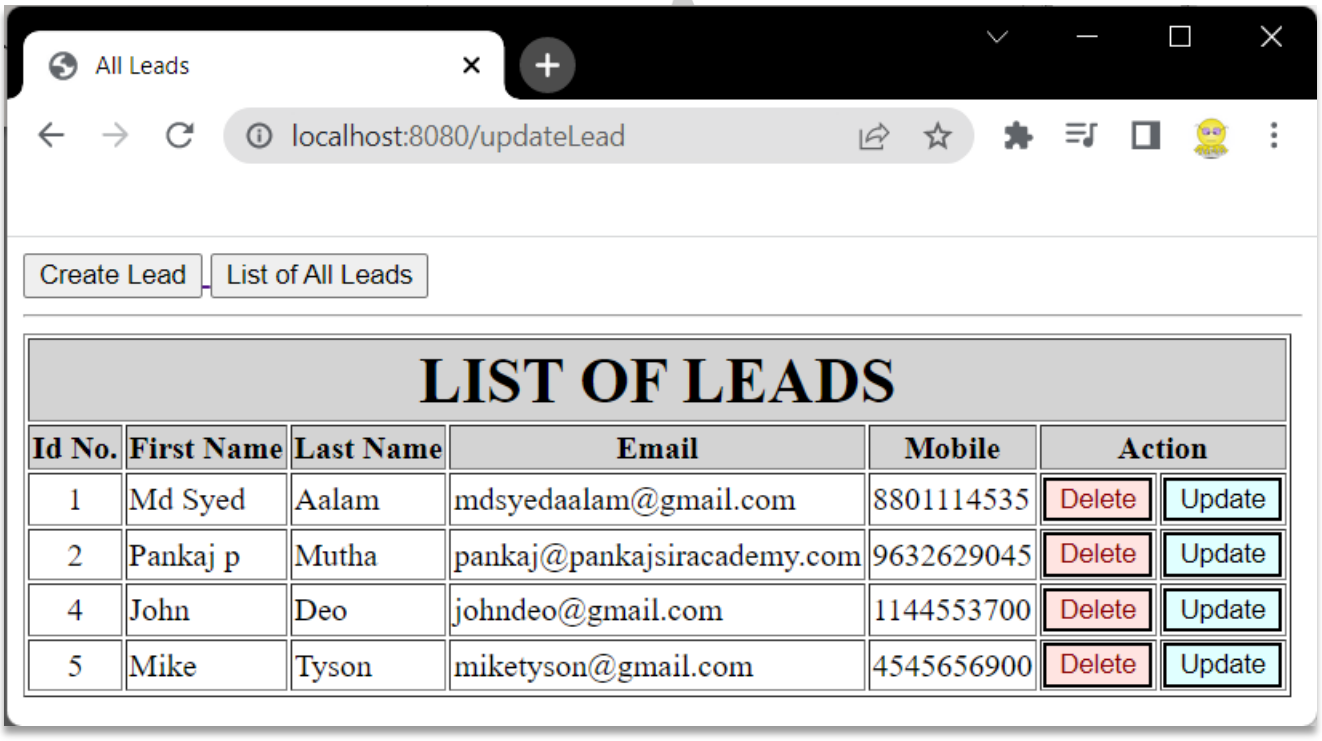
26/07/2022 (Tuesday)

After clicking on update, it will call `@RequestMapping("/updateLead")` in "LeadController.java".

To update the data in the database. (LeadController.java)

```
	@RequestMapping("/updateLead")
	//Using DTO Method to update Lead data as we don't have entity class here
	public String updateOneLeadData(LeadData data, ModelMap model) {
		Lead ld = new Lead();
		ld.setId(data.getId());
		ld.setFirstName(data.getFirstName());
		ld.setLastName(data.getLastName());
		ld.setEmail(data.getEmail());
		ld.setMobile(data.getMobile());
		leadService.saveLead(ld);
		List<Lead> leads = leadService.listLeads();
		model.addAttribute("lds", leads);
		return "leadSearchResult";
	}
}
```

This will redirect back to the "leadSearchResult.jsp" with the updated details.

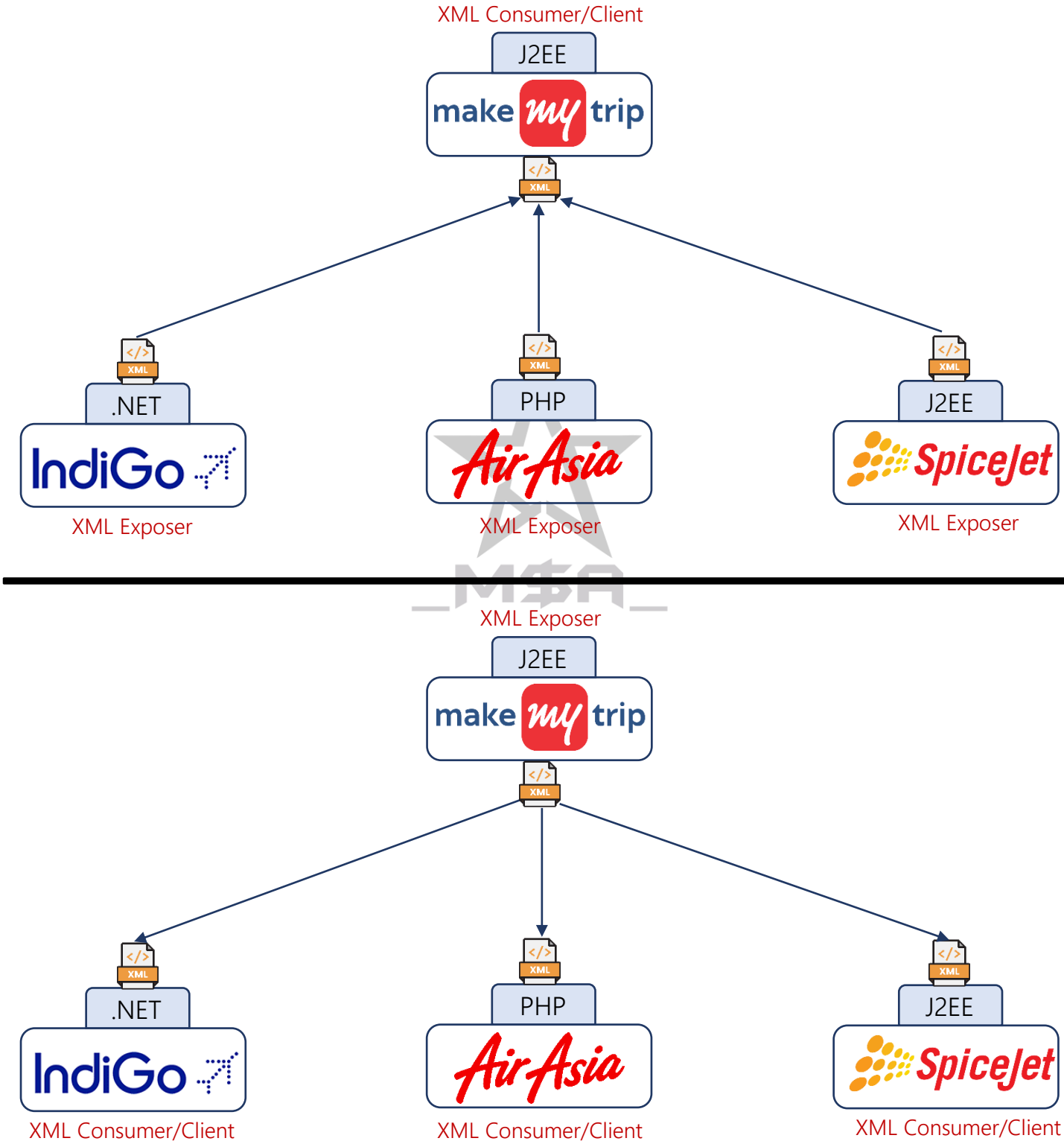


.

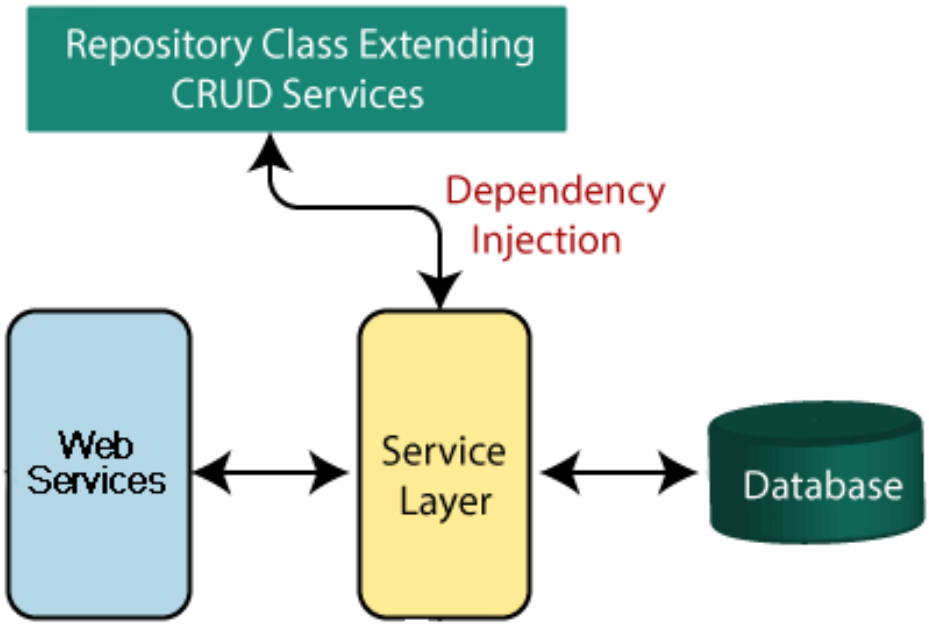
28/07/2022 (Thursday)

WEB SERVICES

It helps us to integrate heterogenous and homogenous applications.



Technical Flow of Web Services Layer:

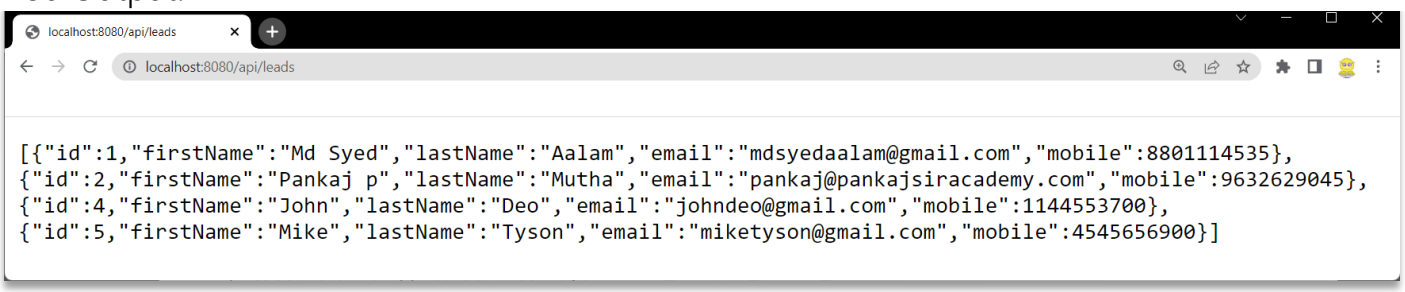


Example:

Create a class named "LeadRestController.java" in Controller layer of project "marketing"

```
package com.marketing.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.marketing.entities.Lead;
import com.marketing.services.LeadService;

@RestController //Web Service Layer
@RequestMapping("/api/leads")
public class LeadRestController {

	@Autowired //to Interact with DB
	private LeadService leadService;

	@GetMapping
	public List <Lead> getAllleads() {
		List<Lead> leads = leadService.listLeads();
		return leads;
	}
}
```

Web Output:



29/07/2022 (Friday)

Web Services is nothing but developing a url to interact with the database.
There are two ways we can implement web services:

1. SOAP: Here we exchange the data between the applications using XML file, implementation of SOAP web services is complex, because we need to parse programmatically XML file.
2. REST: In Rest web services we exchange the data between applications using JSON Object (JAVA Script Object Notation), Implementation of Rest services is easy. However Rest services also supports XML files.

Now a days we're using JSON in place of XML.
JSON stands for JavaScript object notation. JSON has been derived from javascript, where javascript is a programming language. It was originally created to hold the structured data that could be used in javascript. JSON became so popular that it is used for data for all kinds of applications. It is the most popular way of sending the data for Web APIs.

Basic data types supported by json are:
Strings: Characters that are enclosed in single or double quotation marks.
Number: A number could be integer or decimal, positive or negative.
Booleans: The Boolean value could be either true or false without any quotation marks.
Null: Here, null means nothing without any quotation marks.
In addition to basic data types, json has arrays and objects.

Q. Difference between JSON and XML.?

JSON	XML
JSON stands for javascript object notation.	XML stands for an extensible markup language.
The extension of json file is .json.	The extension of xml file is .xml.
The internet media type is application/json.	The internet media type is application/xml or text/xml.
The type of format in JSON is data interchange.	The type of format in XML is a markup language.
It is extended from javascript.	It is extended from SGML.
It is open source means that we do not have to pay anything to use JSON.	It is also open source.
The object created in JSON has some type.	XML data does not have any type.
The data types supported by JSON are strings, numbers, Booleans, null, array.	XML data is in a string format.
It does not have any capacity to display the data.	XML is a markup language, so it has the capacity to display the content.
JSON has no tags.	XML data is represented in tags, i.e., start tag and end tag.

	XML file is larger. If we want to represent the data in XML then it would create a larger file as compared to JSON.
JSON is quicker to read and write.	XML file takes time to read and write because the learning curve is higher.
JSON can use arrays to represent the data.	XML does not contain the concept of arrays.
It can be parsed by a standard javascript function. It has to be parsed before use.	XML data which is used to interchange the data, must be parsed with respective to their programming language to use that.
It can be easily parsed and little bit code is required to parse the data.	It is difficult to parse.
File size is smaller as compared to XML.	File size is larger.
JSON is data-oriented.	XML is document-oriented.
It is less secure than XML.	It is more secure than JSON.

JSON Example:

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML Example:

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

CRUD OPERATIONS IN WEB SERVICES

In order to test the Web Services CRUD operations, we're using a software named "Postman". Download it from here (for Windows 64-Bit).

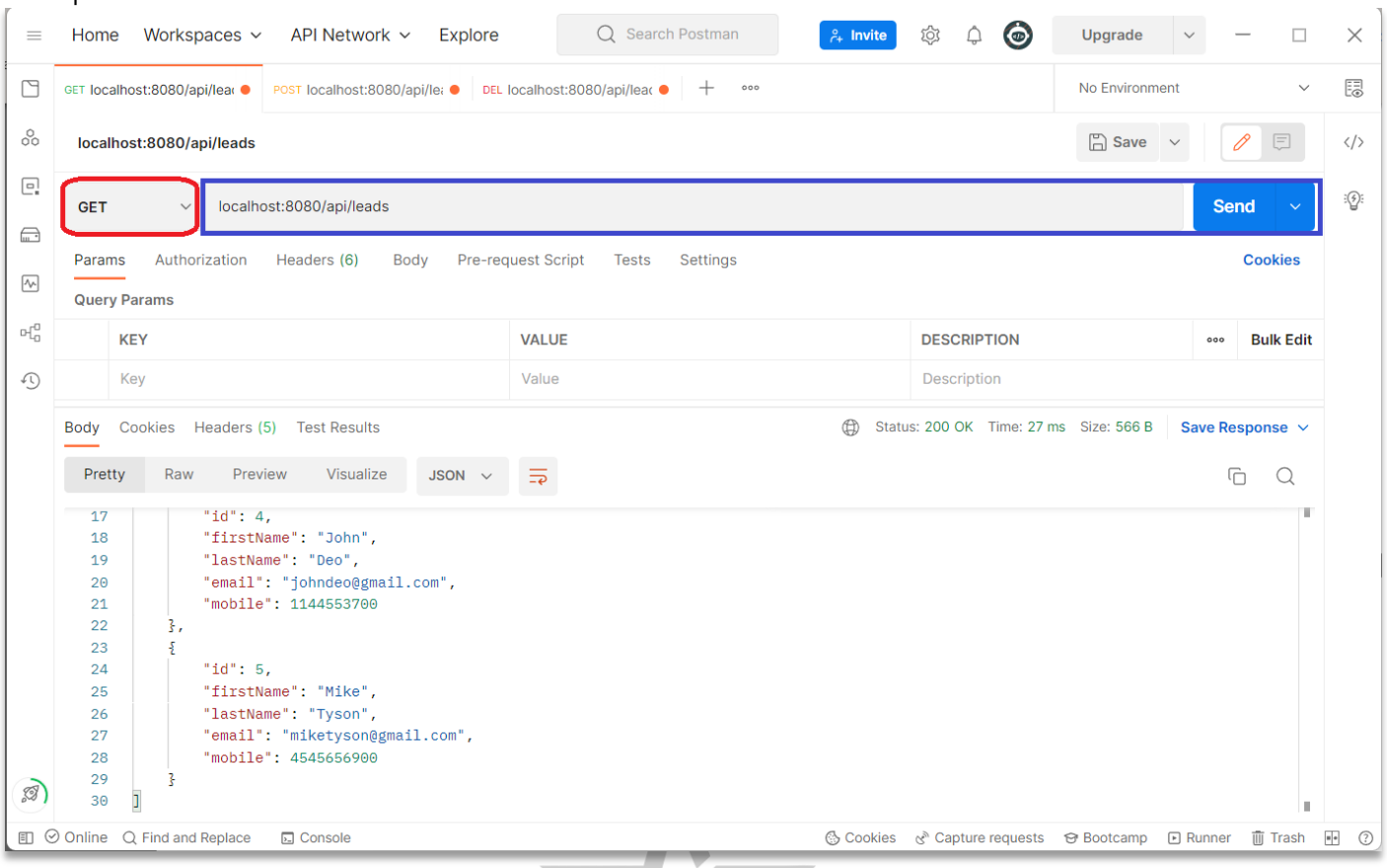
Click here to Download

Reading the Data:

To read the data using web services we just need to enter the url of the api using "GET" method in Postman, here in this case we are using (*localhost:8080/api/leads*) .

The result will be shown in the form of JASON Object.

Example:



Saving the Data:

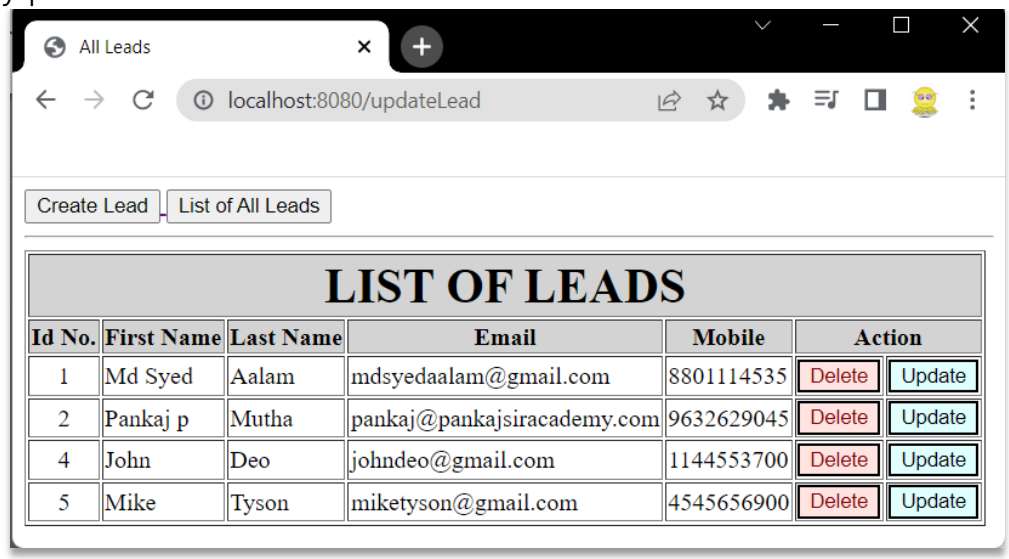
To save the data using web services we need to create a method in "LeadRestController.java"

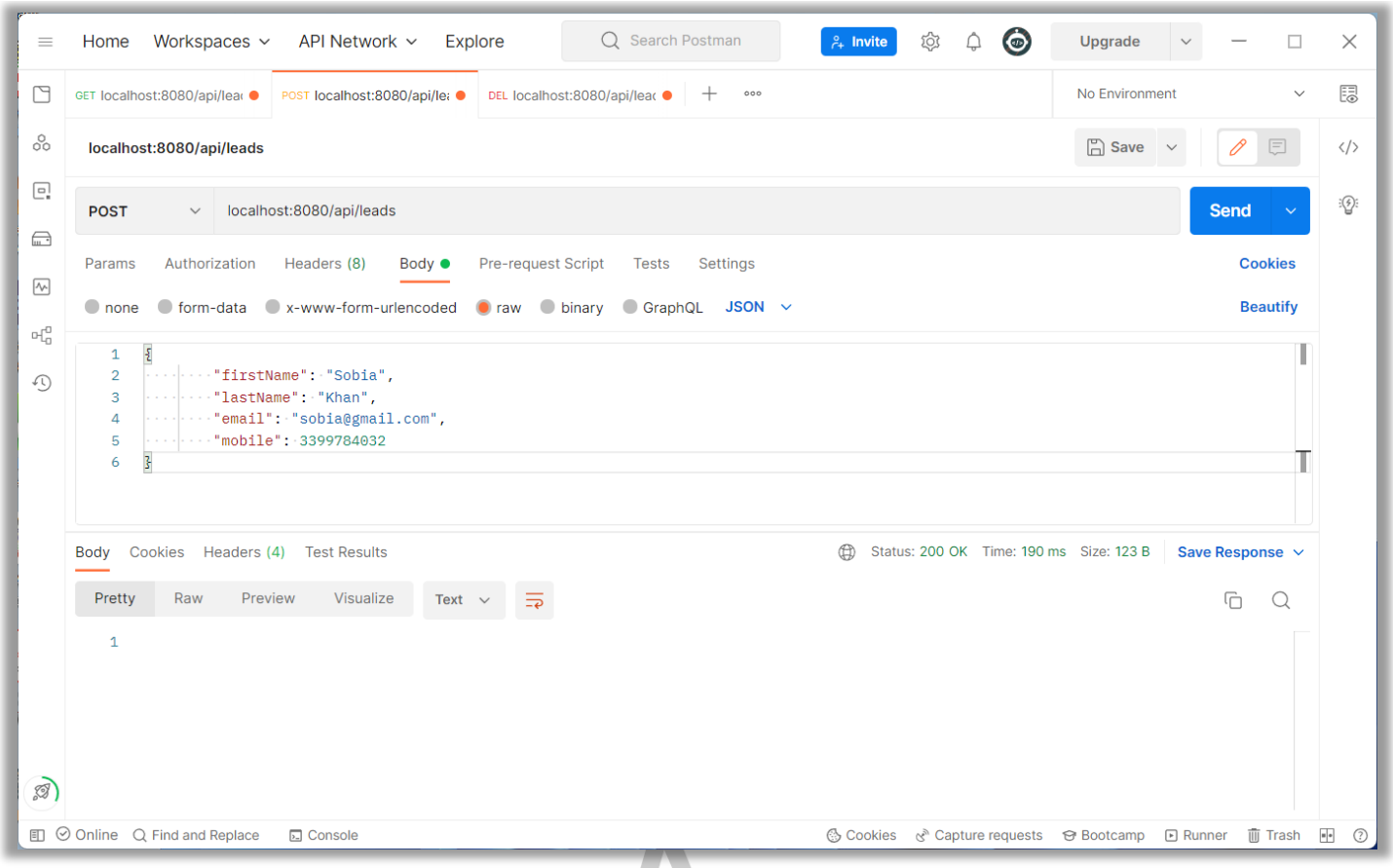
```
@PostMapping //used to save data in the database
public void saveOneLead(@RequestBody Lead lead) {
	//@RequestBody to read the data from the JSON Object & put it into Lead's Object
	leadService.saveLead(lead);
```

Enter the url of the api using "POST" method in Postman. Go to Body>Raw>JASON and input the data as per the JASON object.

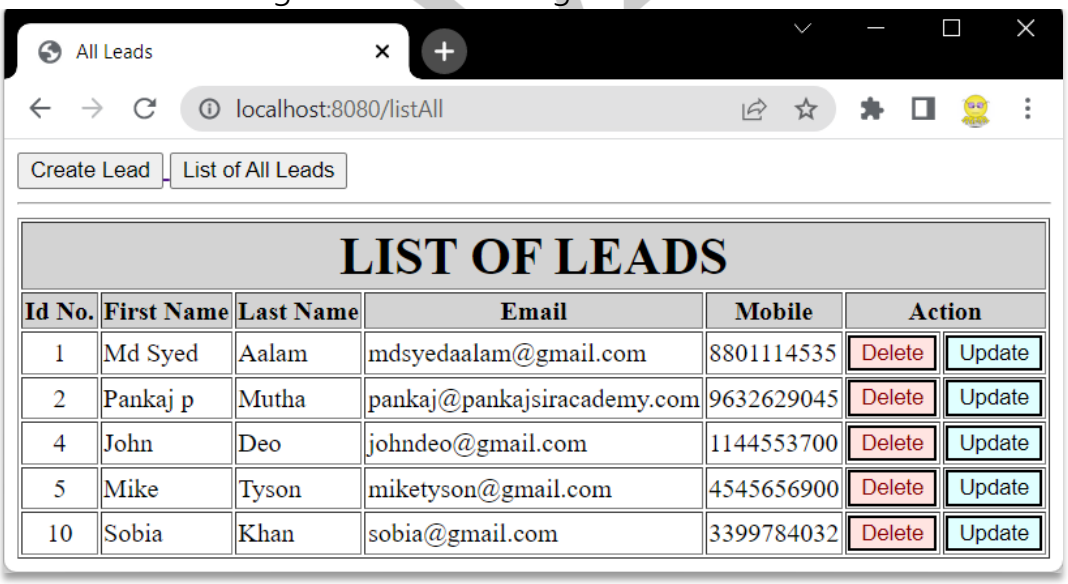
Here we're going to save Sobia's data into the database.

Data currently present in the database:





List of all the leads after saving Sobia's data using Postman:

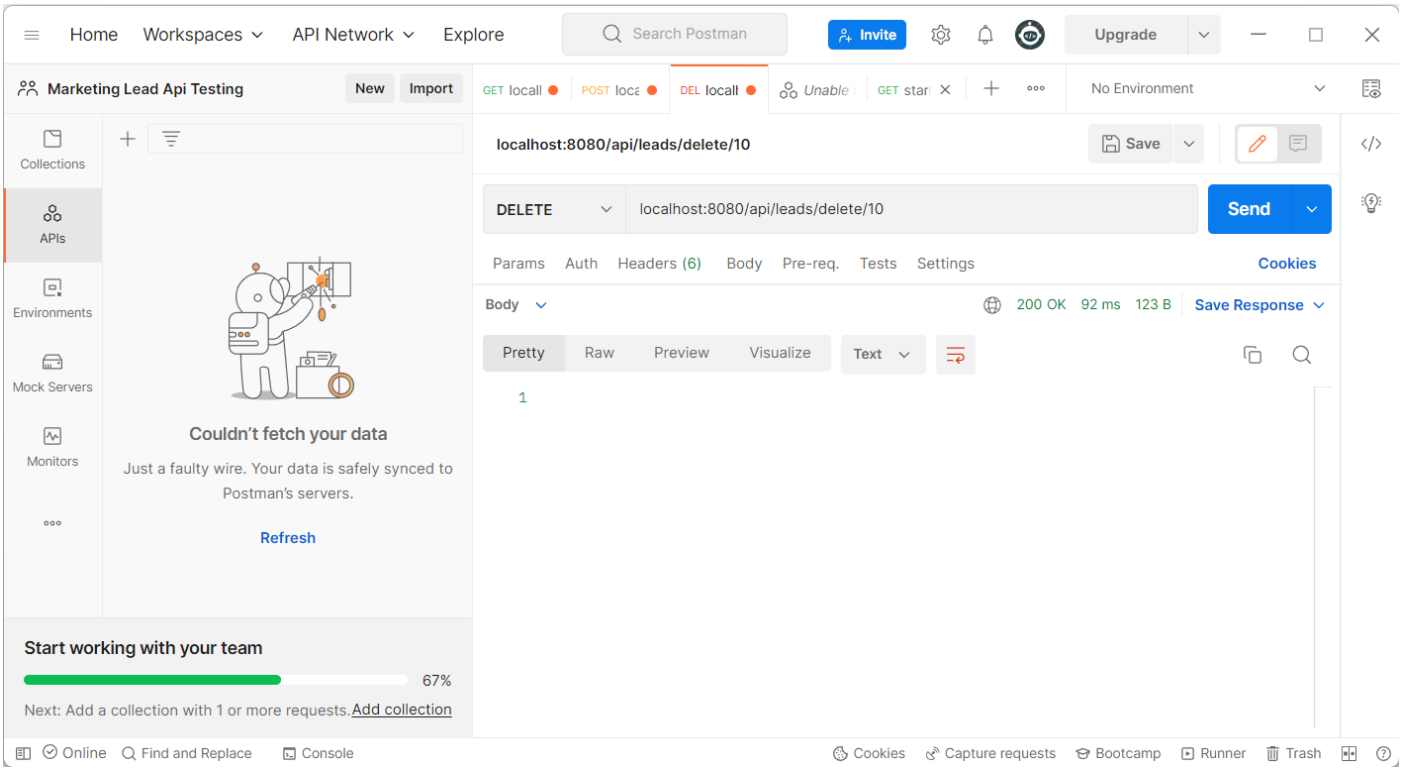


Deleting One Data:

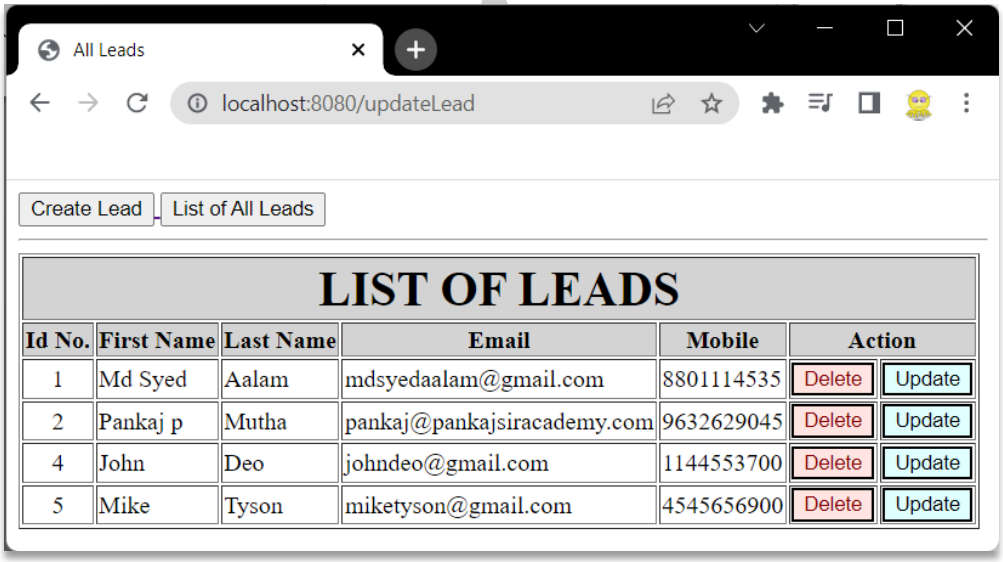
To delete the data using web services we need to create a method in "LeadRestController.java"

```
@DeleteMapping("/delete/{id}")
//("/delete/{id}") so url will be localhost:8080/api/leads/delete/id#
public void deleteOneLead(@PathVariable("id") long id) {
	leadService.deleteLeadById(id);
}
```

Enter the url of the api using "DELETE" method in Postman. Here we're going to delete Sobia's data (id# 10), so ur will be "localhost:8080/api/leads/delete/10"



List of all the leads after deleting Sobia's data using Postman:



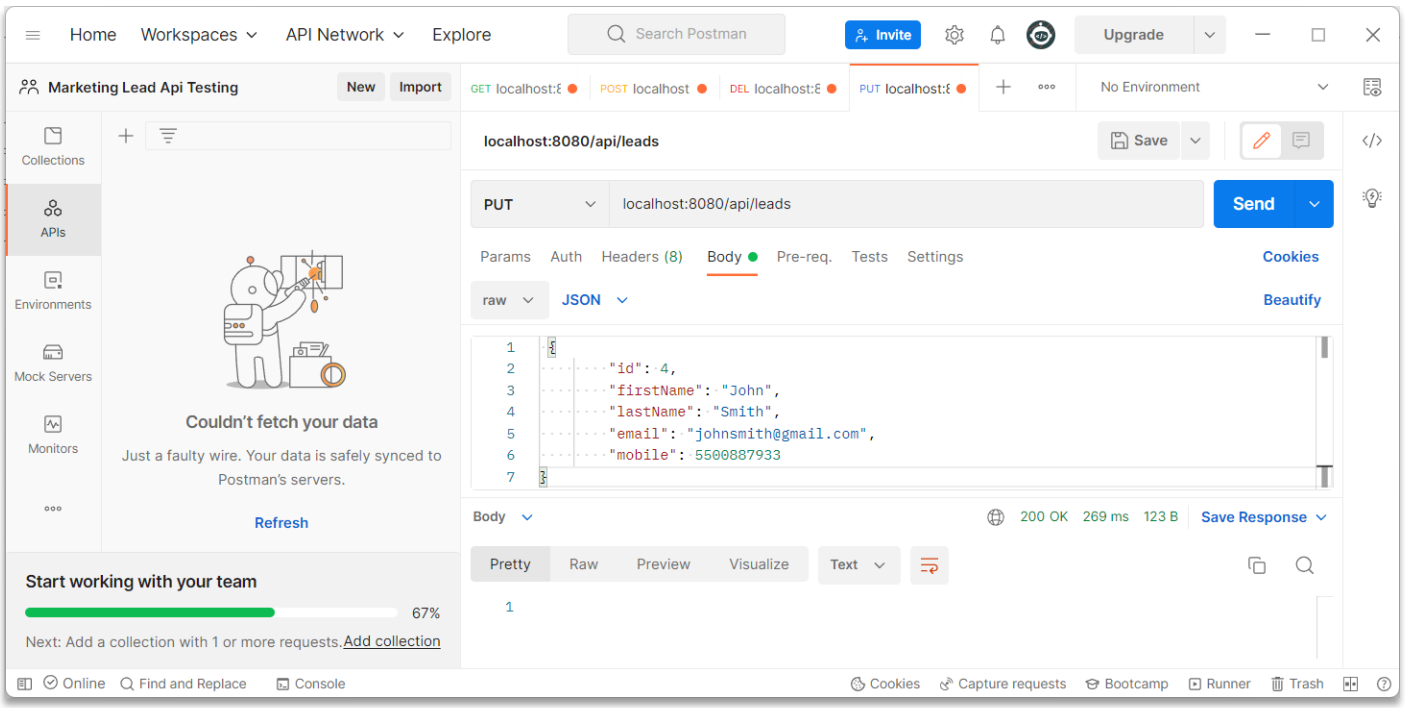
01/08/2022 (Monday)

Updating One Data:

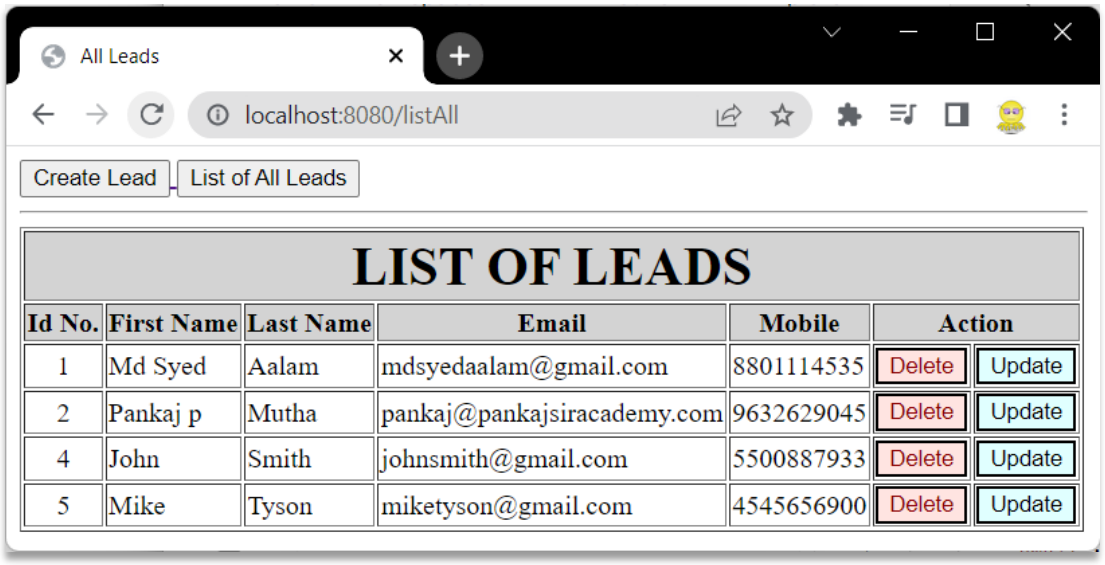
To update the data using web services we need to create a method in "LeadRestController.java"

```
@PutMapping //used to update the data in the database
public void updateOneLead(@RequestBody Lead lead) {
	//@RequestBody to read the data from the JSON Object & put it into Lead's Object
	leadService.saveLead(lead);
}
```

Enter the url of the api using "PUT" method in Postman. url will be same as Saving the lead. To update firstly get all the leads data as JSON object, pick one whose data needs to be updated, Go to Body>Raw>JASON and paste & modify the data, then click on Send. Here we're updating the ID#4: John's Last Name, Email & Mobile Number, ID Cannot be changed as it's a Primary Key.

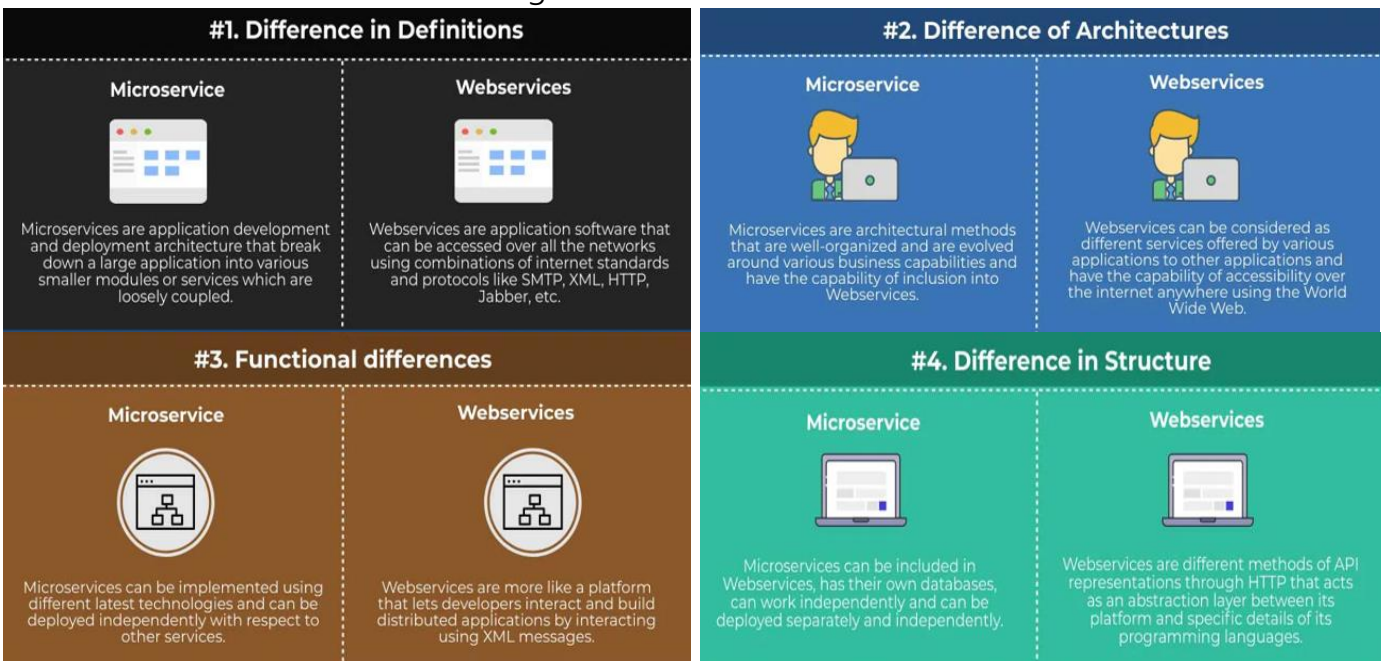


List of the all the leads after updating John's (ID#4) data:



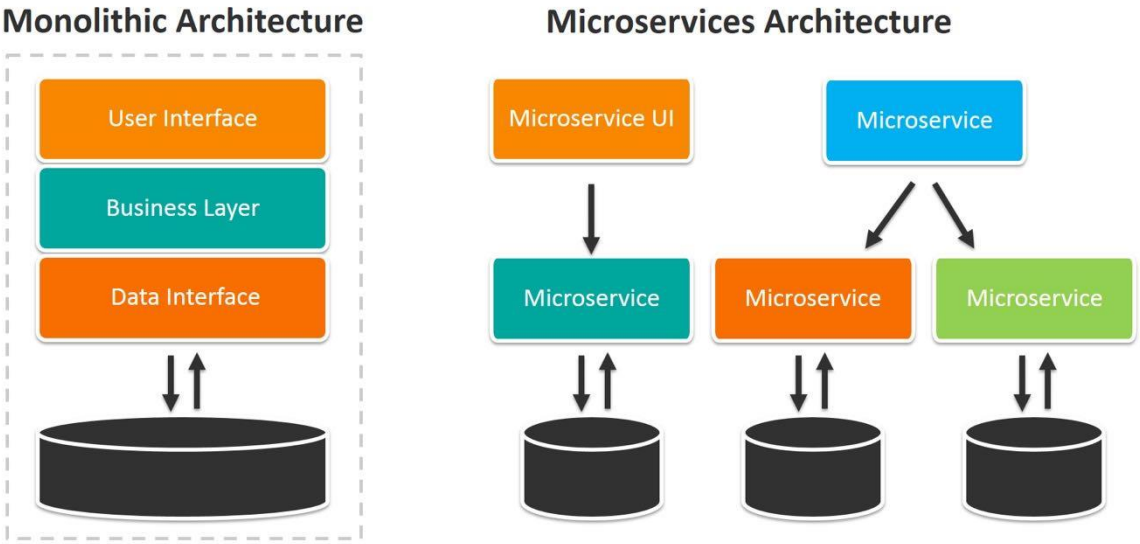
Q. What is the difference between Micro Services and Web Services.?

A. In Microservices we break bigger applications into smaller mini projects & then will establish communication between them using Web Services.



Q. What is Monolithic Application?

A. In Monolithic architecture, all the features and coding is done in single place, and all the systems store and consume data generally from a single location. In this approach, there is a high level of coupling and maintenance is difficult.



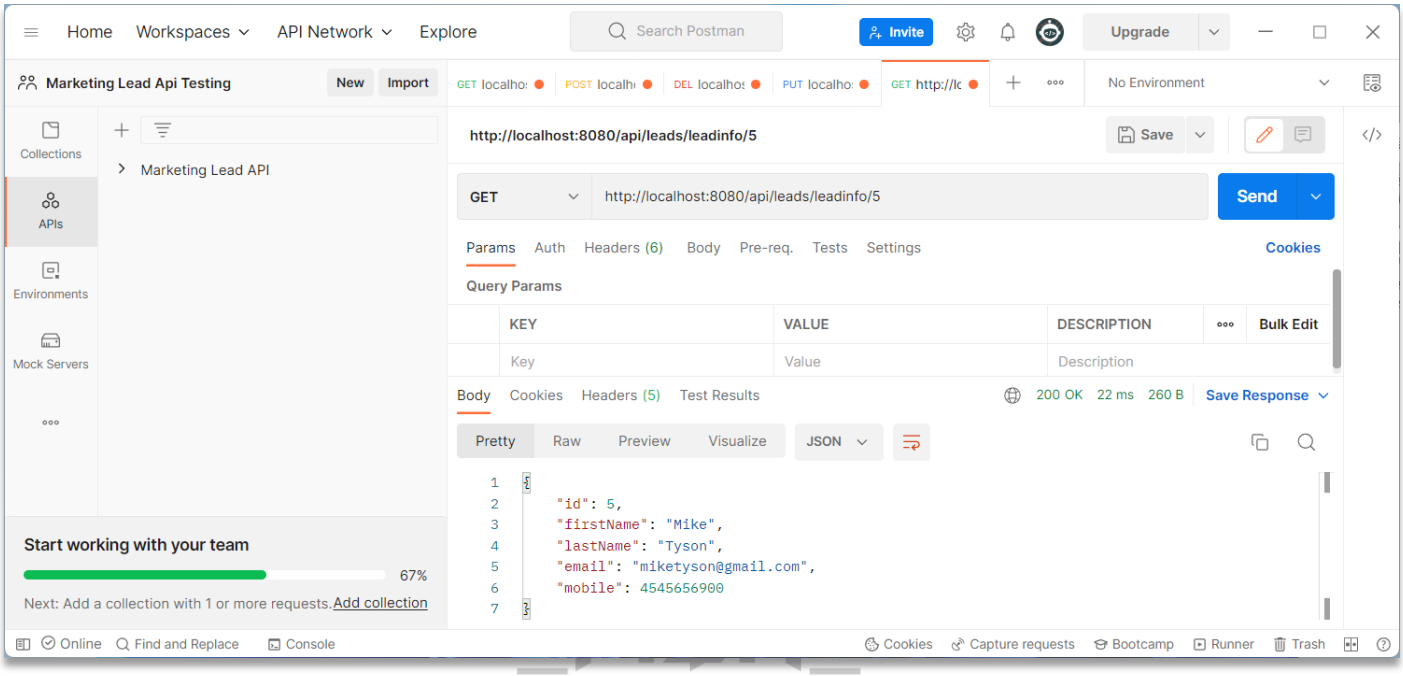
02/08/2022 (Tuesday)

Getting One Lead Data using Id#:

To get one lead data using web services we need to create a method in "LeadRestController.java"

```
@GetMapping("/leadinfo/{id}") //url will be localhost:8080/api/leads/leadinfo/id#
public Lead getOneLead(@PathVariable("id") long id) {
	Lead lead = leadService.getOneLead(id);
	return lead;
}
```

Enter the url of the api using "GET" method in Postman. Here we're going to get Mike's data (id# 5), so ur will be "localhost:8080/api/leads/leadinfo/5"



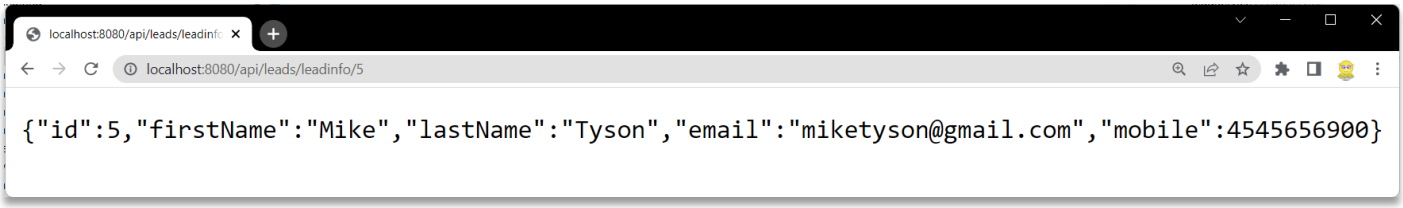
All the CRUD Operations done using Postman also called as "Exposing of the data using Web Services".

Now Let's "Consume the data using Web Services" & build one project to search the lead data using ID# & that project will run on another server port.

To build this part of the project we need the above built api (url):

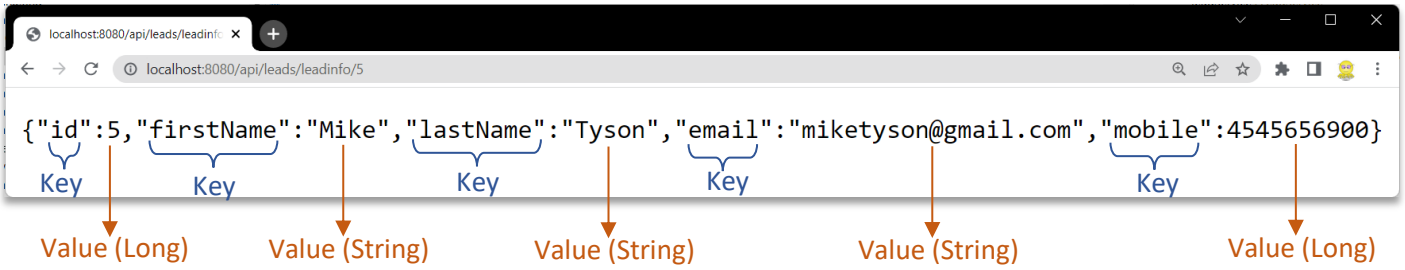
"localhost:8080/api/leads/leadinfo/5"

Let's check this url in the browser:



Here we get the JSON object, based on that we'll start our project by creating the dto package and a JAVA class matching to these variables because we have to copy the data from JSON object to JAVA object.

03/08/2022 (Wednesday)



Project name: "search_lead"

url: localhost:9090/search

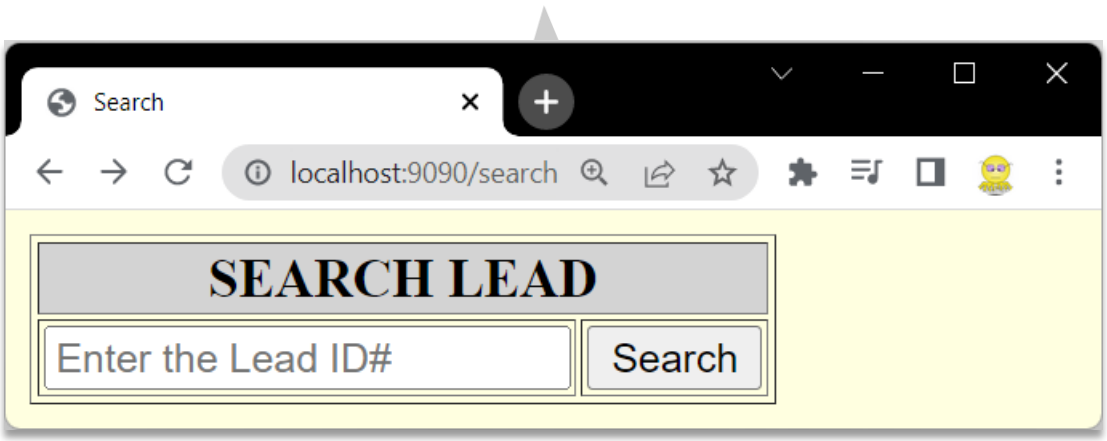
```
package com.search_lead.dto;
public class Lead {
	private long id;
	private String firstName;
	private String lastName;
	private String email;
	private long mobile;
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public long getMobile() {
		return mobile;
	}
	public void setMobile(long mobile) {
		this.mobile = mobile;
	}
}
```

Created the "com.search_lead.dto" package and a "Lead.java" JAVA class matching to the JSON Object because we have to copy the data from JSON object to JAVA object.

05/08/2022 (Friday)

Search Page (View Layer) "search_lead.jsp"

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Search</title>
</head>
<body bgcolor="lightyellow">
<form action="searchLead" method="post">
<table border=1>
<tr><th colspan="2" bgcolor="lightgrey"><font size="4"> SEARCH LEAD </font></th></tr>
      <tr>
            <td> <input type="text" name="id" placeholder="Enter the Lead ID#"/> </td>
            <td> <input type="submit" value="Search" /> </td>
      </tr>
</table>
</form>
</body>
</html>
```



Controller Layer: "SearchLeadController.java"

```
package com.search_lead.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.search_lead.dto.Lead;
import com.search_lead.services.SearchLeadService;
@Controller
public class SearchLeadController {
      @Autowired
      private SearchLeadService leadService;
      @RequestMapping("/search")
      public String viewSearchPage() {
            return "search_lead";
      }
      @RequestMapping("/searchLead")
      public String searchLeadById(@RequestParam("id") long id, ModelMap model) {
            Lead lead = leadService.getLeadById(id);
            model.addAttribute("l", lead);
            return "leadinfo_result";
      }
}
```

Service Layer (Interface): "SearchLeadService.java"

```
package com.search_lead.services;
import com.search_lead.dto.Lead;
public interface SearchLeadService {
      public Lead getLeadById(long id);
}
```

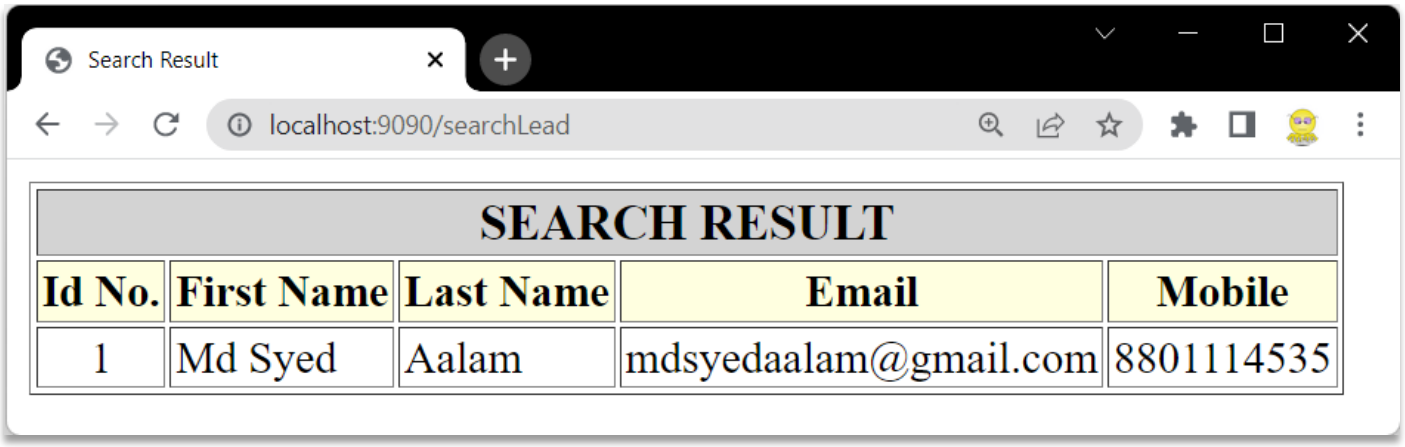
Service Layer (Class): "SearchLeadServiceImpl.java"

```
package com.search_lead.services;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import com.search_lead.dto.Lead;

@Service
public class SearchLeadServiceImpl implements SearchLeadService {

@Override
public Lead getLeadById(long id) {
RestTemplate rt = new RestTemplate(); //Built-in Class for consuming Web Services
Lead lead = rt.getForObject("http://localhost:8080/api/leads/leadinfo/"+id, Lead.class);
//take the json object & save in Lead Class
return lead;
}
}
```

Result Page (View Layer) "leadinfo_result.jsp"

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Search Result </title>
</head>
<body>
<table border=1>
<tr><th colspan="6" bgcolor="lightgrey"><font size="4"> SEARCH RESULT </font></th></tr>
<tr  bgcolor="LightYellow">
<th>Id No.</th><th>First Name</th><th>Last Name</th><th>Email</th><th>Mobile</th></tr>
<tr>
<td style="text-align:center"> ${l.id} </td>
<td>${l.firstName}</td>
<td>${l.lastName}</td>
<td>${l.email}</td>
<td>${l.mobile}</td>
</tr>
</table>
</body>
</html>
```

Search Result Page: (Browser)



Search Lead Application Properties.

```
#Path of the jsp file and its extension
spring.mvc.view.prefix=/WEB-INF/jsps/
spring.mvc.view.suffix=.jsp

#Server Port
server.port=9090
```

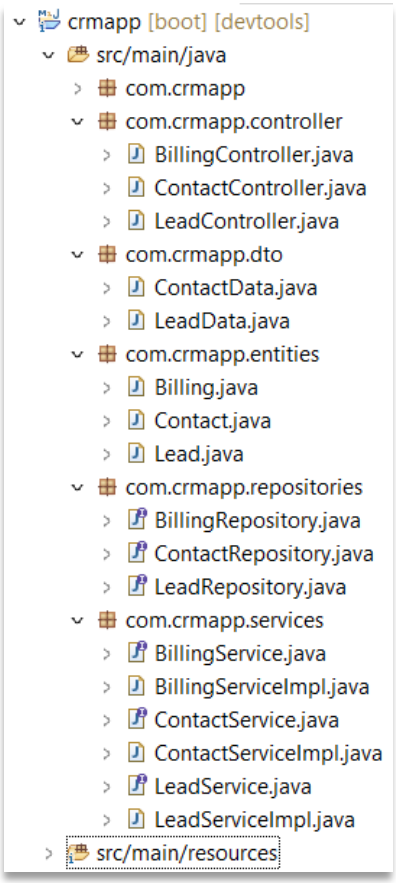
POM.xml (Dependency)

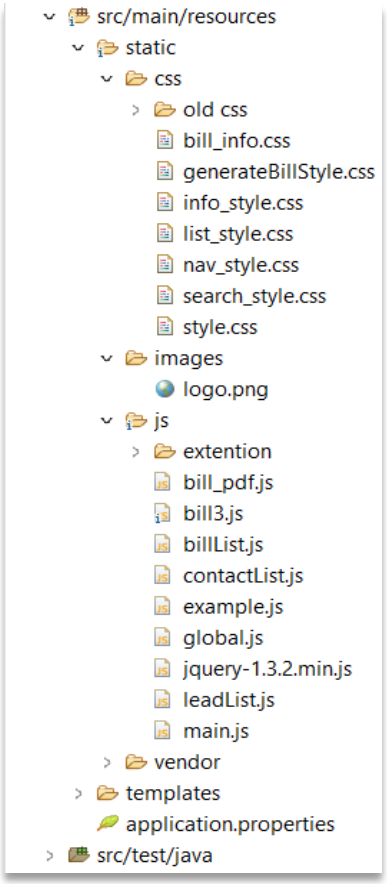
```
	<dependency>
		<groupId>org.apache.tomcat.embed</groupId>
		<artifactId>tomcat-embed-jasper</artifactId>
		<scope>provided</scope>
	</dependency>
```

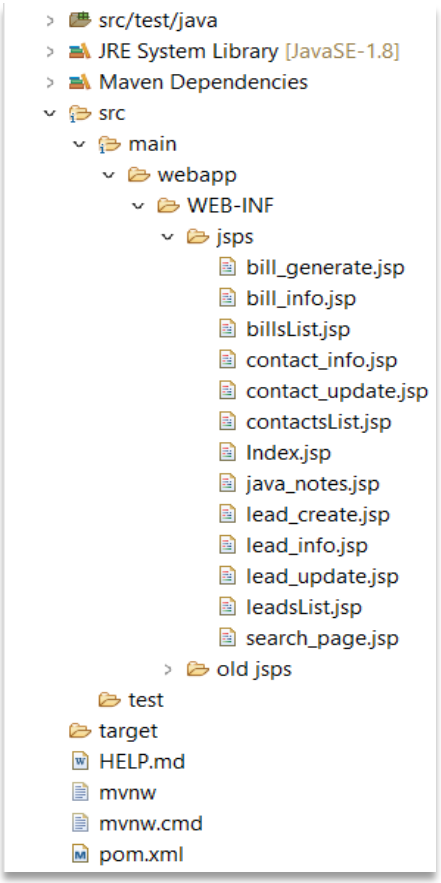
09/08/2022 (Tuesday)

CRM APP (Major Project)

PROJECT STRUCTURE







Note: Not pasting the CSS Style Codes here to restrict the elongation of the pages.

Controller Layers

Lead Controller: "LeadController.java"

```
package com.crmapp.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.crmapp.dto.LeadData;
import com.crmapp.entities.Contact;
```

```java
import com.crmapp.entities.Lead;
import com.crmapp.services.ContactService;
import com.crmapp.services.LeadService;

@Controller
public class LeadController {

	@Autowired
	private LeadService leadService;

	@Autowired
	private ContactService contactService;

	//Home Page
	@RequestMapping("/")
	public String viewCreateLeadPage() {
		return "lead_create";
	}

	//Save the Lead and display the Lead Information
	@RequestMapping("/saveLead")
	public String SaveLead(@ModelAttribute("lead") Lead lead, ModelMap model) {
		leadService.saveOneLead(lead);
		model.addAttribute("lead", lead);
		return "lead_info";
	}

	//Convert Lead to Contact, after clicking on Convert button in Lead info page
	@RequestMapping("/convertLead")
	public String convertLead(@RequestParam("id") long id, ModelMap model) {
		Lead lead = leadService.getOneLeadById(id);
		Contact contact = new Contact();
		contact.setFirstName(lead.getFirstName());
		contact.setLastName(lead.getLastName());
		contact.setEmail(lead.getEmail());
		contact.setMobile(lead.getMobile());
		contactService.saveOneContact(contact); //Save lead in contact

		leadService.deleteOneLeadById(id); //after saving in contact, delete lead
		List <Contact> contacts = contactService.listContacts();
		model.addAttribute("contact", contacts);
		return "contactsList";
	}

	//List of All Leads
	@RequestMapping("leadsList")
	public String lisAllLeads(ModelMap model) {
		List <Lead> leads = leadService.listLeads();
		model.addAttribute("leads", leads);
		return "leadsList";

	}
```

```
	//After clicking on the ID in List of Leads, Open lead info
	@RequestMapping("/leadInfo")
	public String leadInfo(@RequestParam("id") long id,ModelMap model) {
		Lead lead = leadService.getOneLeadById(id);
		model.addAttribute("lead", lead);
		return "lead_info";
	}

	//Go to Search Page
	@RequestMapping("/search")
	public String viewSearchPage() {
		return "search_page";
	}

	//After entering the lead id, show lead info
	@RequestMapping("/searchLead")
	public String searchLeadById(@RequestParam("id") long id, ModelMap model) {
		Lead lead = leadService.getOneLeadById(id);
		model.addAttribute("lead", lead);
		return "lead_info";
	}

	//Delete lead after clicking on delete button in lead info page
	@RequestMapping("/delete")
	public String deleteOneLead(@RequestParam("id") long id,ModelMap model) {
		leadService.deleteOneLeadById(id);
		model.addAttribute("msg", "Lead Deleted Successfully.!!!");

		List <Lead> leads = leadService.listLeads();
		model.addAttribute("leads", leads);
		return "leadsList";
	}

	//Go to update lead page, after clicking on update button in lead info page
	@RequestMapping("/update")
	public String updatePage(@RequestParam("id") long id, ModelMap model) {
		Lead lead = leadService.getOneLeadById(id);
		model.addAttribute("leadUpd", lead);
		return "lead_update";
	}

	//Update the Lead info and save in database
	@RequestMapping("/updateLead") //Used DTO Method to update lead details
	public String updateLead(LeadData data, @ModelAttribute("lead") Lead lead, ModelMap
model) {
		Lead ld = new Lead();
		ld.setId(data.getId());
		ld.setFirstName(data.getFirstName());
		ld.setLastName(data.getLastName());
		ld.setEmail(data.getEmail());
		ld.setMobile(data.getMobile());
		ld.setSource(data.getSource());
		leadService.saveOneLead(ld);
		model.addAttribute("msg", "Lead Details Successfully Updated.!!");

		//After update, show updated lead info
		model.addAttribute("lead", lead);
		return "lead_info";
	}
}
```

Contact Controller: "ContactController.java"

```
package com.crmapp.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.crmapp.dto.ContactData;
import com.crmapp.entities.Contact;
import com.crmapp.services.ContactService;

@Controller
public class ContactController {

	@Autowired
	private ContactService contactService;

	//List of All Contacts
	@RequestMapping("/contactsList")
	public String lisAllContacts(ModelMap model) {
		List <Contact> contact = contactService.listContacts();
		model.addAttribute("contact", contact);
		return "contactsList";
	}

	//After clicking on the ID or Name in List of Contacts, Open contact info
	@RequestMapping("/contactInfo")
	public String contactInfo(@RequestParam("id") long id,ModelMap model) {
		Contact contact = contactService.getOneContactById(id);
		model.addAttribute("contact", contact);
		return "contact_info";
	}

	//Delete contact
	@RequestMapping("/deleteContact")
	public String deleteOneContact(@RequestParam("id") long id,ModelMap model) {
		contactService.deleteOneContactById(id);
		model.addAttribute("msg", "Contact Deleted Successfully.!!!");

		List <Contact> contact = contactService.listContacts();
		model.addAttribute("contact", contact);
		return "contactsList";
	}

	//Go to update contact page
	@RequestMapping("/updateContact")
	public String updatePage(@RequestParam("id") long id, ModelMap model) {
		Contact contact = contactService.getOneContactById(id);
		model.addAttribute("contactUpd", contact);
		return "contact_update";
	}
```

```
//Update the info and save in database
	@RequestMapping("/updateContactInfo") //Used DTO Method to update lead details
	public String updateContact(ContactData data, @ModelAttribute("contact") Contact
contact, ModelMap model) {
		Contact ct = new Contact();
		ct.setId(data.getId());
		ct.setFirstName(data.getFirstName());
		ct.setLastName(data.getLastName());
		ct.setEmail(data.getEmail());
		ct.setMobile(data.getMobile());
		contactService.saveOneContact(ct);
		model.addAttribute("msg", "Contact Details Successfully Updated.!!");

		//After update, show updated contact info
		model.addAttribute("contact", contact);
		return "contact_info";
	}

	//After entering the contact id in search page, show contact info
	@RequestMapping("/searchContact")
	public String searchContactById(@RequestParam("id") long id, ModelMap model) {
		Contact contact = contactService.getOneContactById(id);
		model.addAttribute("contact", contact);
		return "contact_info";
	}
}
```

Billing Controller: "BillingController.java"

```
package com.crmapp.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.crmapp.entities.Billing;
import com.crmapp.entities.Contact;
import com.crmapp.services.BillingService;
import com.crmapp.services.ContactService;

@Controller
public class BillingController {

	@Autowired
	private BillingService billingService;

	@Autowired
	private ContactService contactService;

	//Go to billing page
	@RequestMapping("/billingPage")
	public String billingPage(@RequestParam("id") long id, ModelMap model) {
	Contact contact = contactService.getOneContactById(id);
	model.addAttribute("contact", contact);
		return "bill_generate";
	}
```

```
	//Save the bill in database & show invoice
	@RequestMapping("/generateBill")
	public String generateBill(@ModelAttribute("bill") Billing bill, ModelMap model) {
		billingService.saveBill(bill);
		model.addAttribute("bill", bill);
		return "bill_info";
	}

	//To read PSA JAVA notes online, link in index
	@RequestMapping("/javanotes")
	public String javaNotes() {
		return "java_notes";
	}

	//show list of all the bills
	@RequestMapping("/billsList")
	public String billsList(ModelMap model) {
		List <Billing> bills = billingService.listAllBills();
		model.addAttribute("bills", bills);
		return "billsList";
	}

	//After clicking on "View" show invoice
	@RequestMapping("/billInfo")
	public String getOneBill(@RequestParam("invoiceNo") long invoiceNo, ModelMap model)
{
		Billing bill = billingService.getOneBill(invoiceNo);
		model.addAttribute("bill", bill);
		return "bill_info";
	}

	//Delete the invoice
	@RequestMapping("/deleteBill")
	public String deleteOneBill(@RequestParam("invoiceNo") long invoiceNo, ModelMap
model) {
		billingService.deleteOneBill(invoiceNo);

		List <Billing> bills = billingService.listAllBills();
		model.addAttribute("bills", bills);
		return "billsList";
	}
}
```

DTO Layers

Lead DTO (Data Transfer Object) Class: "LeadData.java"

```
package com.crmapp.dto;
public class LeadData {
	private long id;
	private String firstName;
	private String lastName;
	private String email;
	private long mobile;
	private String source;
```

```
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public long getMobile() {
		return mobile;
	}
	public void setMobile(long mobile) {
		this.mobile = mobile;
	}
	public String getSource() {
		return source;
	}
	public void setSource(String source) {
		this.source = source;
	}
}
```

Contact DTO (Data Transfer Object) Class: "ContactData.java"

```
package com.crmapp.dto;
public class ContactData {
	private long id;
	private String firstName;
	private String lastName;
	private String email;
	private long mobile;
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
```

```
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public long getMobile() {
		return mobile;
	}
	public void setMobile(long mobile) {
		this.mobile = mobile;
	}
}
```

Entity Layers

Lead Entity Class: "Lead.java"

```
package com.crmapp.entities;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="crmlead")
public class Lead {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private long id;

	@Column(name = "first_name", nullable=false, length = 45)
	private String firstName;

	@Column(name = "last_name", nullable=false, length = 45)
	private String lastName;

	@Column(name = "email", nullable=false, unique=true, length = 128)
	private String email;

	@Column(name = "mobile", nullable=false, unique=true, length = 10)
	private long mobile;

	@Column(name = "source", nullable=false)
	private String source;

	public long getId() {
		return id;
	}
```

```
	public void setId(long id) {
		this.id = id;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public long getMobile() {
		return mobile;
	}

	public void setMobile(long mobile) {
		this.mobile = mobile;
	}

	public String getSource() {
		return source;
	}

	public void setSource(String source) {
		this.source = source;
	}
}
```

Contact Entity Class: "Contact.java"

```
package com.crmapp.entities;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
```

```java
@Entity
@Table(name="crmcontacts")
public class Contact {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private long id;

	@Column(name = "first_name", nullable=false, length = 45)
	private String firstName;

	@Column(name = "last_name", nullable=false, length = 45)
	private String lastName;

	@Column(name = "email", nullable=false, unique=true, length = 128)
	private String email;

	@Column(name = "mobile", nullable=false, unique=true, length = 10)
	private long mobile;

	public long getId() {
		return id;
	}

	public void setId(long id) {
		this.id = id;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public long getMobile() {
		return mobile;
	}

	public void setMobile(long mobile) {
		this.mobile = mobile;
	}
}
```

Billing Entity Class: "Billing.java"

```
package com.crmapp.entities;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "billing")
public class Billing {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "invoice_no")
	private long invoiceNo;
	@Column(name = "first_name")
	private String firstName;
	@Column(name = "last_name")
	private String lastName;
	@Column(unique=true)
	private String email;
	private String date;
	private long mobile;
	private String product;
	private int price;

	public long getInvoiceNo() {
		return invoiceNo;
	}
	public void setInvoiceNo(long invoiceNo) {
		this.invoiceNo = invoiceNo;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public long getMobile() {
		return mobile;
	}
```

```
	public void setMobile(long mobile) {
		this.mobile = mobile;
	}
	public String getProduct() {
		return product;
	}
	public void setProduct(String product) {
		this.product = product;
	}
	public int getPrice() {
		return price;
	}
	public void setPrice(int price) {
		this.price = price;
	}
	public String getDate() {
		return date;
	}
	public void setDate(String date) {
		this.date = date;
	}
}
```

Repository Layers

Lead Repository (Interface): "LeadRepository.java"

```
package com.crmapp.repositories;
import org.springframework.data.jpa.repository.JpaRepository;
import com.crmapp.entities.Lead;
public interface LeadRepository extends JpaRepository<Lead, Long> {
}
```

Contact Repository (Interface): "ContactRepository.java"

```
package com.crmapp.repositories;
import org.springframework.data.jpa.repository.JpaRepository;
import com.crmapp.entities.Contact;
public interface ContactRepository extends JpaRepository<Contact, Long> {
}
```

Billing Repository (Interface): "BillingRepository.java"

```
package com.crmapp.repositories;
import org.springframework.data.jpa.repository.JpaRepository;
import com.crmapp.entities.Billing;
public interface BillingRepository extends JpaRepository<Billing, Long> {
}
```

Service Layers

Lead Service (Interface): "LeadService.java"

```
package com.crmapp.services;
import java.util.List;
import com.crmapp.entities.Lead;
public interface LeadService {
	public void saveOneLead(Lead lead);
	public Lead getOneLeadById(long id);
	public void deleteOneLeadById(long id);
	public List<Lead> listLeads();
}
```

Lead Service Implementation (Class): "LeadServiceImpl.java"

```
package com.crmapp.services;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.crmapp.entities.Lead;
import com.crmapp.repositories.LeadRepository;
@Service
public class LeadServiceImpl implements LeadService {

	@Autowired
	private LeadRepository leadRepo;

	@Override
	public void saveOneLead(Lead lead) {
		leadRepo.save(lead);
	}
	@Override
	public Lead getOneLeadById(long id) {
		Optional<Lead> findById = leadRepo.findById(id);
		Lead lead = findById.get();
		return lead;
	}
	@Override
	public void deleteOneLeadById(long id) {
		leadRepo.deleteById(id);
	}
	@Override
	public List<Lead> listLeads() {
		List<Lead> leads = leadRepo.findAll();
		return leads;
	}
}
```

Contact Service (Interface): "ContactService.java"

```
package com.crmapp.services;
import java.util.List;
import com.crmapp.entities.Contact;
public interface ContactService {
	public void saveOneContact (Contact contact);
	public List<Contact> listContacts();
	public Contact getOneContactById(long id);
	public void deleteOneContactById(long id);
}
```

Contact Service Implementation (Class): "ContactServiceImpl.java"

```
package com.crmapp.services;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.crmapp.entities.Contact;
import com.crmapp.repositories.ContactRepository;

@Service
public class ContactServiceImpl implements ContactService {

	@Autowired
	private ContactRepository contactRepo;

	@Override
	public void saveOneContact(Contact contact) {
		contactRepo.save(contact);
	}
	@Override
	public List<Contact> listContacts() {
		List<Contact> contacts = contactRepo.findAll();
		return contacts;
	}
	@Override
	public Contact getOneContactById(long id) {
		Optional<Contact> findById = contactRepo.findById(id);
		Contact contact = findById.get();
		return contact;
	}
	@Override
	public void deleteOneContactById(long id) {
	contactRepo.deleteById(id);
	}
}
```

Billing Service (Interface): "BillingService.java"

```
package com.crmapp.services;
import java.util.List;
import com.crmapp.entities.Billing;
public interface BillingService {
	public List<Billing> listAllBills();
	public void saveBill(Billing bill);
	public Billing getOneBill(long invoiceNo);
	public void deleteOneBill(long invoiceNo);
}
```

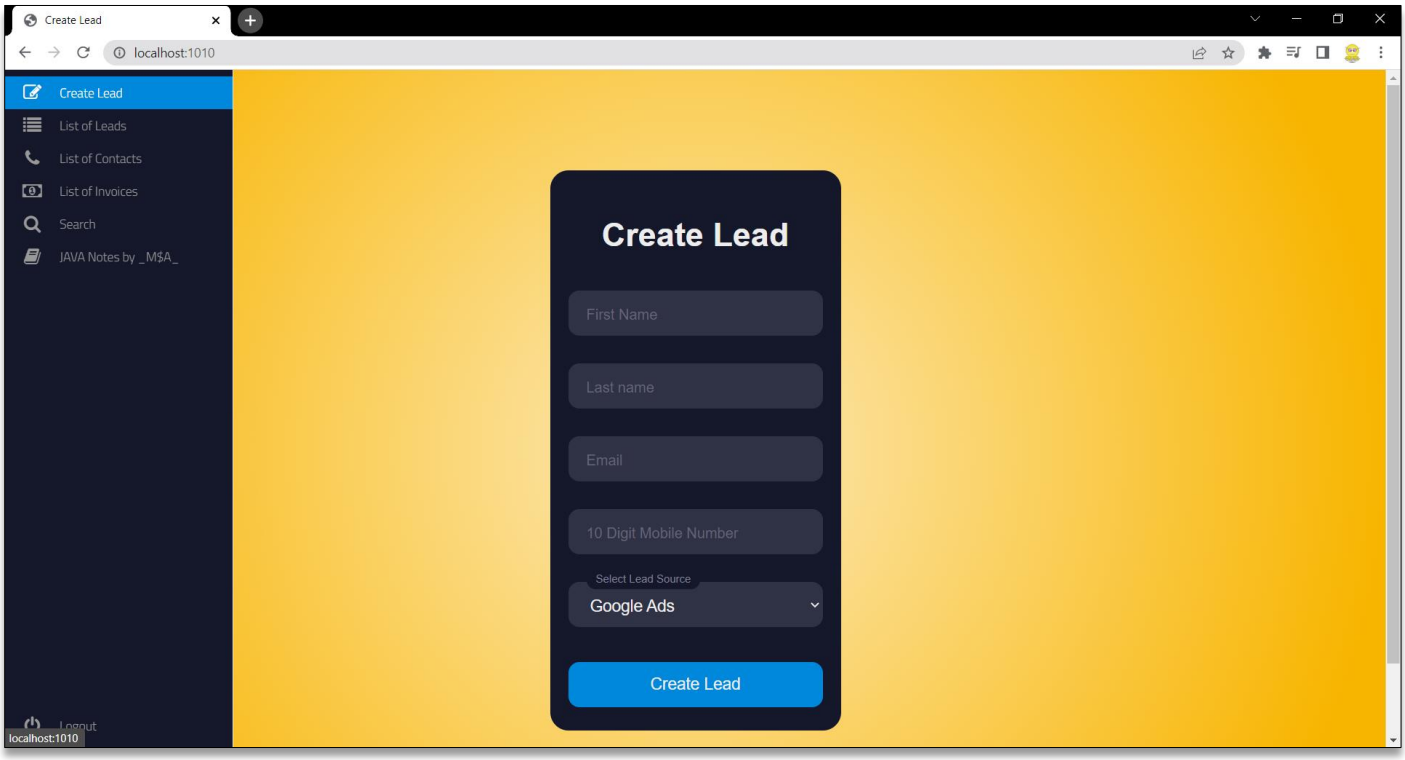
Billing Service Implementation (Class): "BillingServiceImpl.java"

```
package com.crmapp.services;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.crmapp.entities.Billing;
import com.crmapp.repositories.BillingRepository;

@Service
public class BillingServiceImpl implements BillingService {
	@Autowired
	private BillingRepository billingRepo;

	@Override
	public void saveBill(Billing bill) {
		billingRepo.save(bill);
	}
	@Override
	public List<Billing> listAllBills() {
	List<Billing> bills = billingRepo.findAll();
		return bills;
	}
	@Override
	public Billing getOneBill(long invoiceNo) {
		Optional<Billing> findById = billingRepo.findById(invoiceNo);
		Billing bill = findById.get();
		return bill;
	}
	@Override
	public void deleteOneBill(long invoiceNo) {
		billingRepo.deleteById(invoiceNo);
	}
}
```

View Layers

Create Lead (View Layer): "create_lead.jsp"

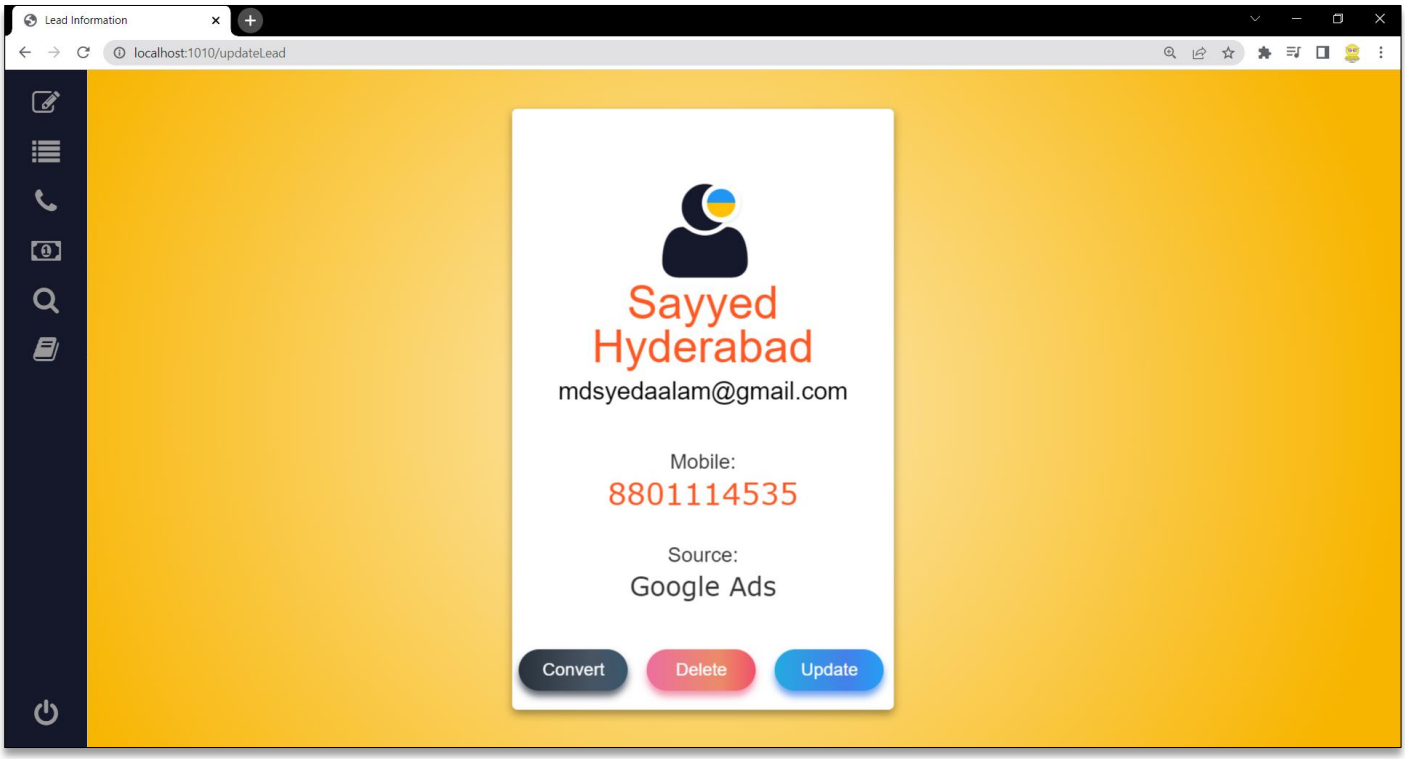
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<!DOCTYPE html>
<html>
<head><link rel="stylesheet" type="text/css" href="/css/style.css">
<meta charset="ISO-8859-1">
</head>
<title> Create Lead </title>
<body>
<form action="saveLead" method="post">
         <div class="form">
      <div class="title">Create Lead</div>
      <div class="input-container ic1">
        <input name="firstName" id="firstname" class="input" type="text" placeholder=" "
required/>
        <div class="cut"></div>
        <label for="firstname" class="placeholder">First Name</label>
      </div>
      <div class="input-container ic2">
        <input name="lastName" id="lastname" class="input" type="text" placeholder=" "
required/>
        <div class="cut"></div>
        <label for="lastname" class="placeholder">Last name</label>
      </div>
      <div class="input-container ic2">
        <input name="email" id="email" class="input" type="email" placeholder=" " re-
quired/>
        <div class="cut cut-short1"></div>
        <label for="email" class="placeholder">Email</>
      </div>
      <div class="input-container ic2">
        <input name="mobile" id="mobile" class="input" type="text" minlength="10"
maxlength="10" placeholder=" " required/>
        <div class="cut cut-short"></div>
        <label for="mobile" class="placeholder">10 Digit Mobile Number</>
      </div>
      <div class="input-container ic2">
            <label for="mobile" class="placeholder"></label>
                <select name="source" id="email" class="input" required>
                <option value="Google Ads">Google Ads</option>
                  <option value="Facebook">Facebook</option>
                  <option value="Twitter">Twitter</option>
                  <option value="News Paper">News Paper</option>
                  <option value="TV Commercial">TV Commercial</option>
                  <option value="Google Ads">Google Ads</option>
                </select>
                <div class="cut cut-short2"></div>
                <label for="mobile" class="placeholder">Select Lead Source</>
      </div>
      <button type="submit" class="submit">Create Lead</button>
    </div>
</form>
</body>
</html>
```



Lead Info (View Layer): "lead_info.jsp"

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<!DOCTYPE html>
<html>
<link rel="stylesheet" type="text/css" href="/css/info_style.css">
<head>
<meta charset="ISO-8859-1">
<title> Lead Information </title>
</head>
<body>
      <aside class="profile-card">
  <header>
    <a class="fa fa-user" style="font-size:80px;color:#15172b"></a>
    <h1>
            ${lead.firstName} ${lead.lastName}
     </h1>
    <h2>
     ${lead.email}
    </h2>
  </header>
  <div class="profile-bio">
    <p style="font-size:15px"> Mobile:  </p>  <p style="font-size:22px; font-family:ver-
dana; color:#FF5722"> ${lead.mobile} </p> <br>
    <p style="font-size:15px"> Source: </p>  <p style="font-size:19px; font-family:ver-
dana"> ${lead.source} </p>
  </div>
```

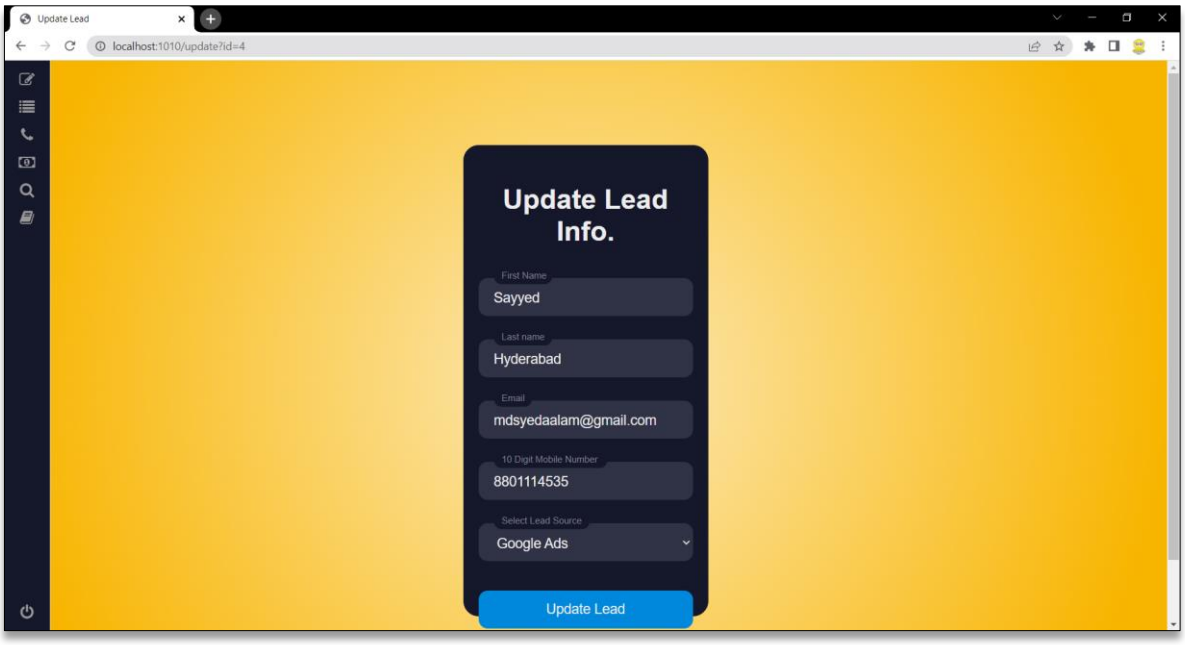
```
  <div class="social">
      <form action="convertLead" method="post" style="display: inline;">
                        <input type="hidden" name="id" value="${lead.id}">
                    <input class="btn-hover color-8" type="submit" value="Convert">
      </form>
      <a href="delete?id=${lead.id}"><input class="btn-hover color-10" type="submit"
value="Delete"></a>
      <a href="update?id=${lead.id}"><input class="btn-hover color-9" type="submit"
value="Update"></a>
  </div>
</aside>
</body>
</html>
```



Lead Update (View Layer): "lead_update.jsp"

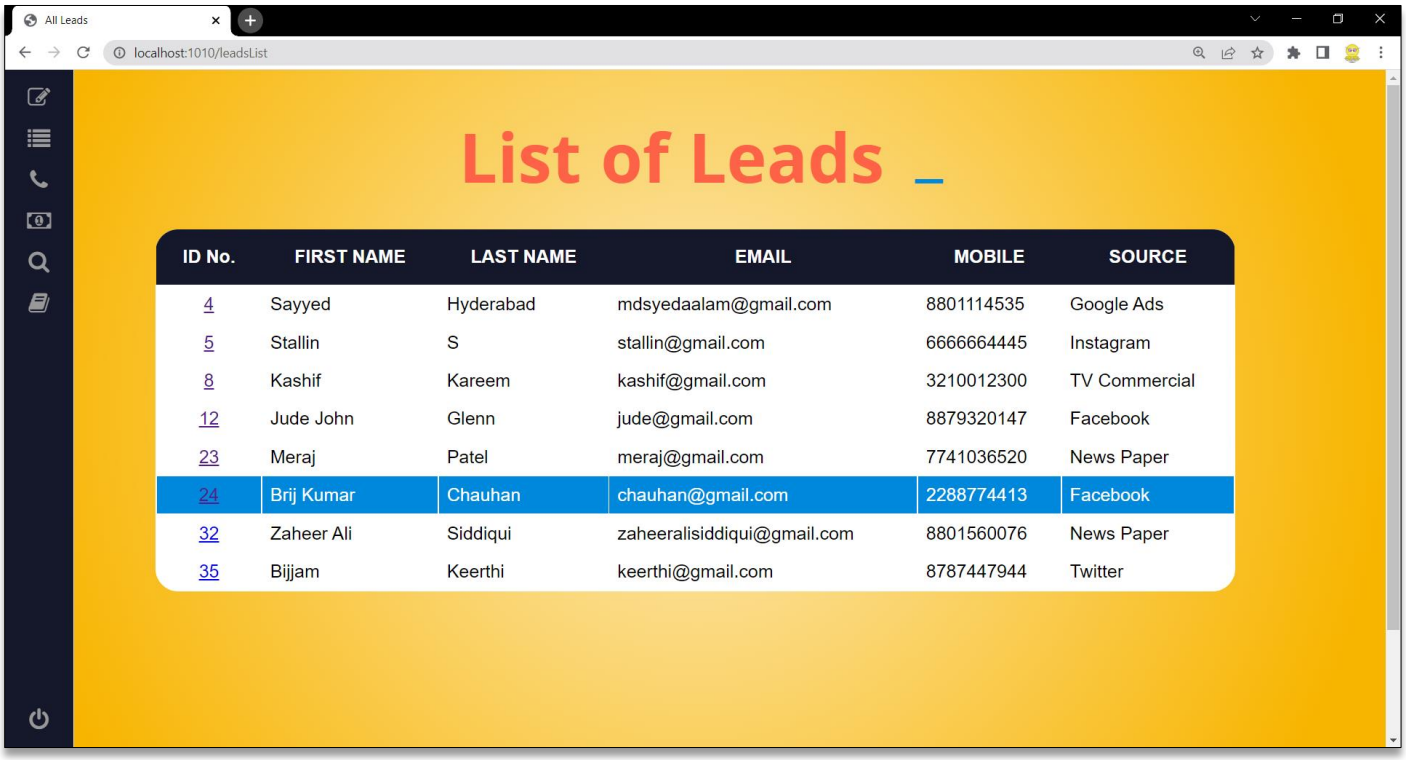
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<!DOCTYPE html>
<html>
<head><link rel="stylesheet" type="text/css" href="/css/style.css">
<meta charset="ISO-8859-1">
</head>
<title> Update Lead </title>
<body>
<form  action="updateLead" method="post">
           <input type="hidden" name="id" value="${leadUpd.id}"/>
        <div class="form">
      <div class="title">Update Lead Info.</div>
      <div class="input-container ic1">
       <input name="firstName" id="firstname" class="input" type="text"
value="${leadUpd.firstName}"/>
        <div class="cut"></div>
        <label for="firstname" class="placeholder">First Name</label>
      </div>
```

```
      <div class="input-container ic2">
        <input name="lastName" id="lastname" class="input" type="text"
value="${leadUpd.lastName}"/>
        <div class="cut"></div>
        <label for="lastname" class="placeholder">Last name</label>
      </div>
      <div class="input-container ic2">
        <input name="email" id="email" class="input" type="email"
value="${leadUpd.email}"/>
        <div class="cut cut-short1"></div>
        <label for="email" class="placeholder">Email</>
      </div>
      <div class="input-container ic2">
        <input name="mobile" id="mobile" class="input" type="text" maxlength="10"
value="${leadUpd.mobile}"/>
        <div class="cut cut-short"></div>
        <label for="mobile" class="placeholder">10 Digit Mobile Number</>
      </div>
      <div class="input-container ic2">
            <label for="mobile" class="placeholder"></label>
                  <select name="source" id="email" class="input" required>
                  <option value="Google Ads">Google Ads</option>
                    <option value="Facebook">Facebook</option>
                    <option value="Twitter">Twitter</option>
                    <option value="News Paper">News Paper</option>
                    <option value="TV Commercial">TV Commercial</option>
                    <option value="Google Ads">Google Ads</option>
                  </select>
                  <div class="cut cut-short2"></div>
                  <label for="mobile" class="placeholder">Select Lead Source</>
      </div>

      <button type="submit" class="submit">Update Lead</button>
    </div>
</form>
</body>
</html>
```



List All Leads (View Layer): "leadslist.jsp"

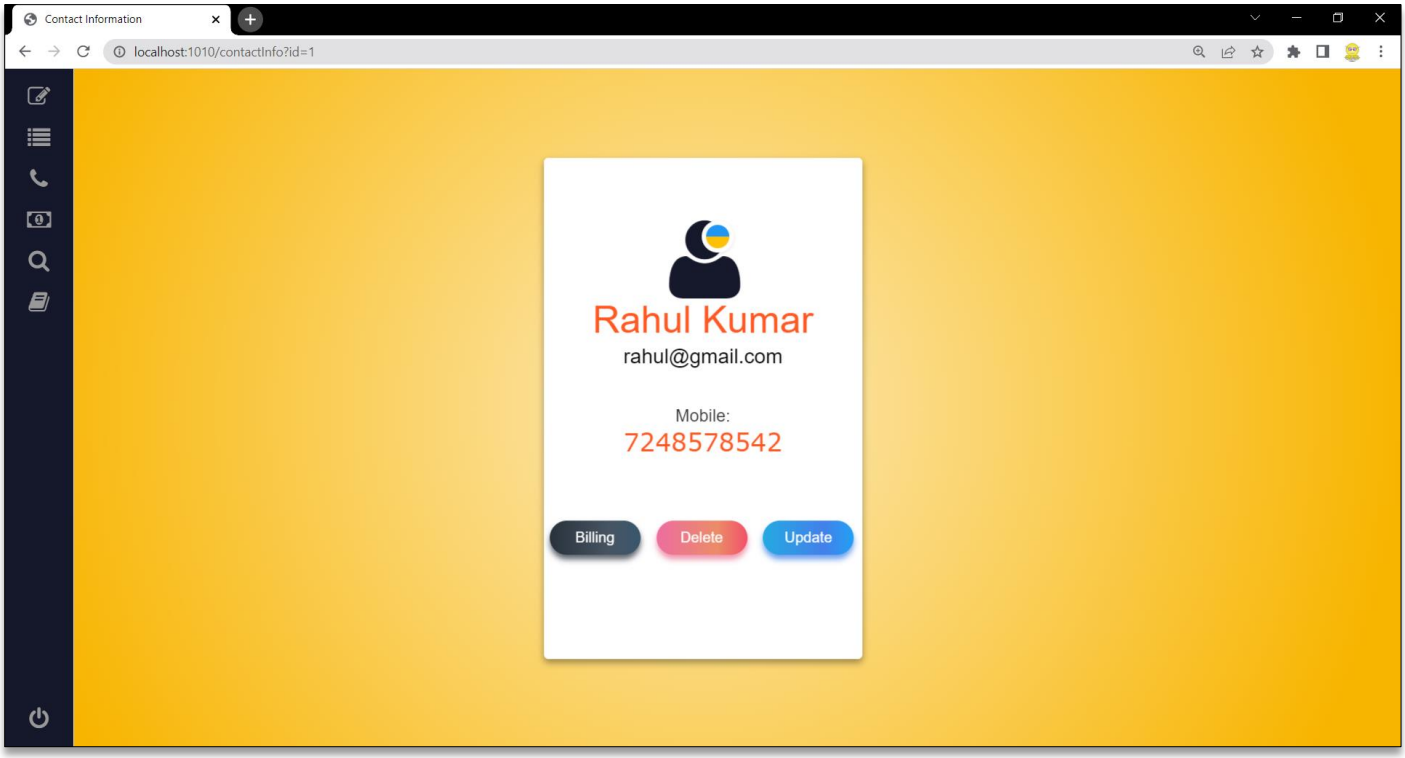
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core"%> <!-- JSTL Directive
Tag -->
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="/css/list_style.css">
<meta charset="ISO-8859-1">
<title> All Leads </title>
</head>
<body class="heading-body">
            <div class='console-container'><span id='text'></span>
            <script type="text/javascript" src="js/leadList.js"></script>
            <div class='console-underscore' id='console'>&#95;</div></div>
</body>
      <body>
            <div class="container">
                  <table class="hoverTable">
                        <thead>
                              <tr>
                                    <th>ID No.</th>
                                    <th>FIRST NAME</th>
                                    <th>LAST NAME</th>
                                    <th>EMAIL</th>
                                    <th>MOBILE</th>
                                    <th>SOURCE</th>
                              </tr>
                        </thead>
                        <tbody>
                              <c:forEach var="lead" items="${leads}">
                              <tr>
                                    <td style="text-align:center"><a href="lead-
Info?id=${lead.id}"> ${lead.id}</a></td>
                                    <td>${lead.firstName}</td>
                                    <td>${lead.lastName}</td>
                                    <td>${lead.email}</td>
                                    <td>${lead.mobile}</td>
                                    <td>${lead.source}</td>
                              </tr>
                        </c:forEach>
                        </tbody>
                  </table>
            </div>
      </body>
</html>
```



Contact Info (View Layer): "contact_info.jsp"

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<!DOCTYPE html>
<html>
<link rel="stylesheet" type="text/css" href="/css/info_style.css">
<head>
<meta charset="ISO-8859-1">
<title> Contact Information </title>
</head>
<body>
      <aside class="profile-card">
  <header>
    <a class="fa fa-user" style="font-size:80px;color:#15172b"></a>
    <h1>
            ${contact.firstName} ${contact.lastName}
     </h1>
    <h2>
     ${contact.email}
    </h2>
  </header>
  <div class="profile-bio">
    <p style="font-size:15px"> Mobile:  </p>  <p style="font-size:22px; font-family:ver-
dana; color:#FF5722"> ${contact.mobile} </p> <br>
  </div>
```

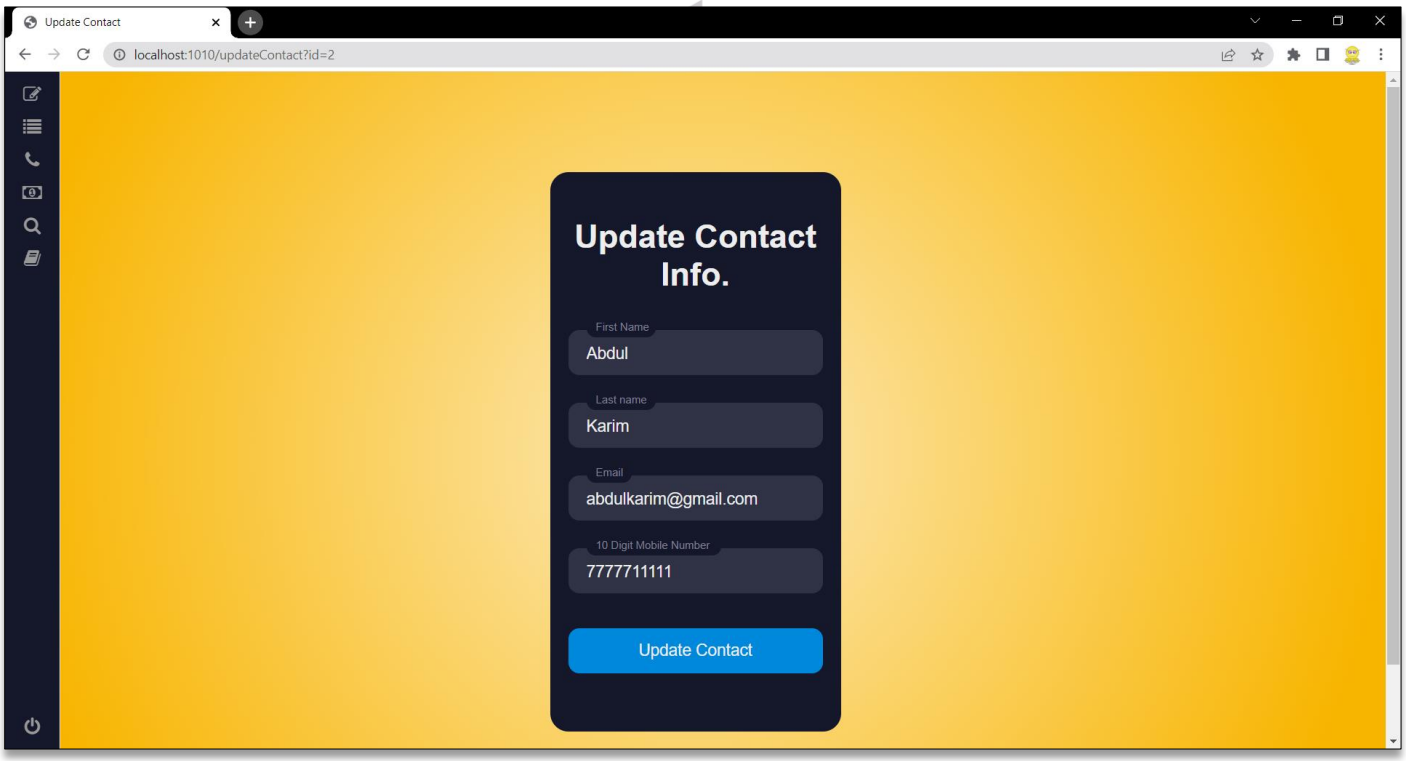
```
  <div class="social">
      <a href="updateContact?id=${contact.id}"><input class="btn-hover color-8"
type="submit" value="Billing"></a>
      <a href="delete?id=${contact.id}"><input class="btn-hover color-10" type="submit"
value="Delete"></a>
      <a href="updateContact?id=${contact.id}"><input class="btn-hover color-9"
type="submit" value="Update"></a>
  </div>
</aside>
</body>
</html>
```



Update Contacts (View Layer): "contact_update.jsp"

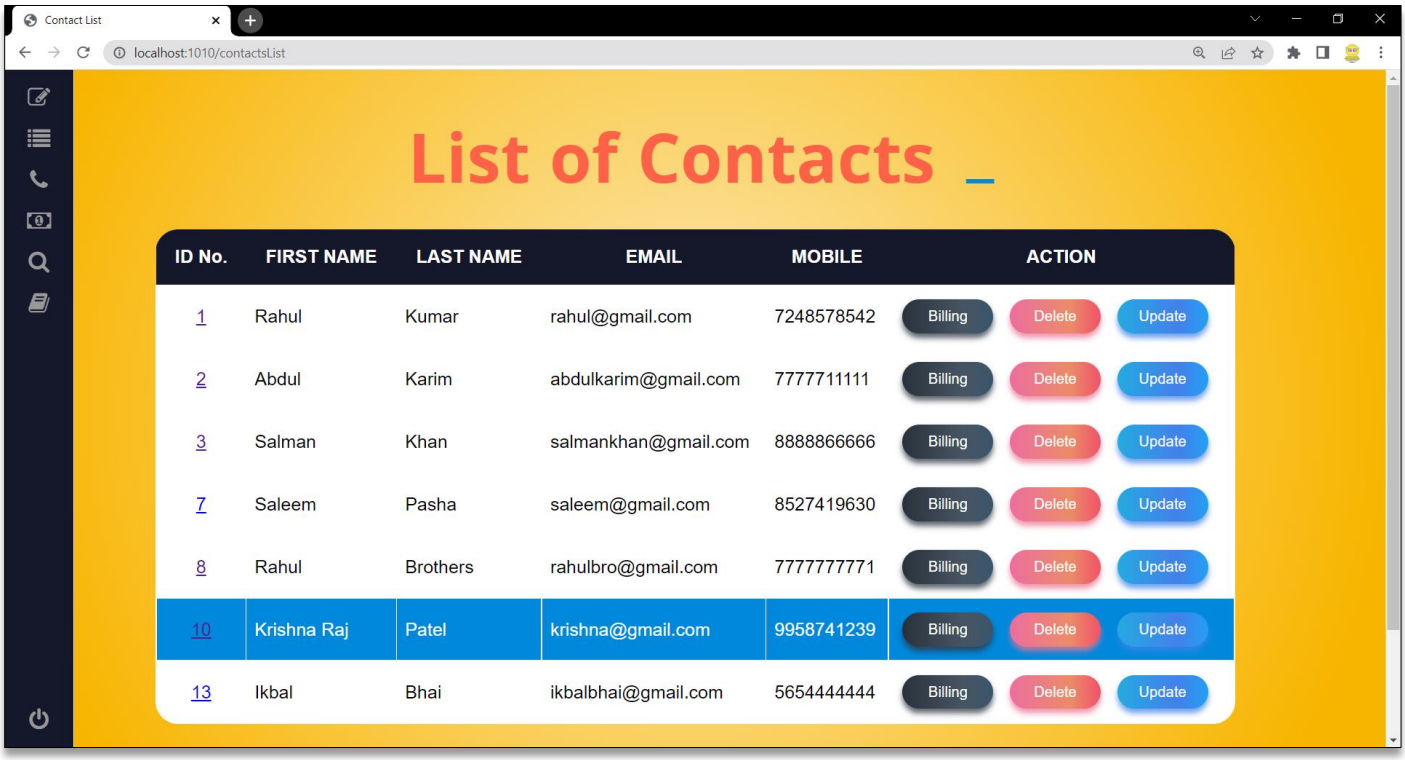
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<!DOCTYPE html>
<html>
<head><link rel="stylesheet" type="text/css" href="/css/style.css">
<meta charset="ISO-8859-1">
</head>
<title> Update Contact </title>
<body>
<form  action="updateContactInfo" method="post">
          <input type="hidden" name="id" value="${contactUpd.id}"/>
        <div class="form">
      <div class="title">Update Contact Info.</div>
```

```
      <div class="input-container ic1">
       <input name="firstName" id="firstname" class="input" type="text" value="${contac-
tUpd.firstName}"/>
        <div class="cut"></div>
        <label for="firstname" class="placeholder">First Name</label>
      </div>
      <div class="input-container ic2">
        <input name="lastName" id="lastname" class="input" type="text" value="${contac-
tUpd.lastName}"/>
        <div class="cut"></div>
        <label for="lastname" class="placeholder">Last name</label>
      </div>
      <div class="input-container ic2">
        <input name="email" id="email" class="input" type="email" value="${contac-
tUpd.email}"/>
        <div class="cut cut-short1"></div>
        <label for="email" class="placeholder">Email</>
      </div>
      <div class="input-container ic2">
        <input name="mobile" id="mobile" class="input" type="text" maxlength="10"
value="${contactUpd.mobile}"/>
        <div class="cut cut-short"></div>
        <label for="mobile" class="placeholder">10 Digit Mobile Number</>
      </div>
      <button type="submit" class="submit">Update Contact</button>
    </div>
</form>
${msg}
</body>
</html>
```



List All Contacts (View Layer): "contactslist.jsp"

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core"%> <!-- JSTL Directive
Tag -->
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="/css/list_style.css">
<meta charset="ISO-8859-1">
<title> Contact List </title>
</head>
<body class="heading-body">
            <div class='console-container'><span id='text'></span>
            <script type="text/javascript" src="js/contactList.js"></script>
            <div class='console-underscore' id='console'>&#95;</div></div>
</body>
      <body>
            <div class="container">
                  <table class="hoverTable">
                        <thead>
                              <tr>
                                    <th>ID No.</th>
                                    <th>FIRST NAME</th>
                                    <th>LAST NAME</th>
                                    <th>EMAIL</th>
                                    <th>MOBILE</th>
                                    <th>ACTION</th>
                              </tr>
                        </thead>
                        <tbody>
                              <c:forEach var="contact" items="${contact}">
                              <tr>
                                    <td style="text-align:center"><a href="con-
tactInfo?id=${contact.id}"> ${contact.id}</a></td>
                                    <td>${contact.firstName}</td>
                                    <td>${contact.lastName}</td>
                                    <td>${contact.email}</td>
                                    <td>${contact.mobile}</td>
                                    <td>
                                          <div class="social">
                                                <a href="billingPage?id=${con-
tact.id}"><input class="btn-hover color-8" type="submit" value="Billing"></a>
                                                <a href="deleteContact?id=${con-
tact.id}"><input class="btn-hover color-10" type="submit" value="Delete"></a>
                                                <a href="updateContact?id=${con-
tact.id}"><input class="btn-hover color-9" type="submit" value="Update"></a>
                                          </div>
                                    </td>
                              </tr>
                        </c:forEach>
                        </tbody>
                  </table>
            </div>
      </body>
</html>
```

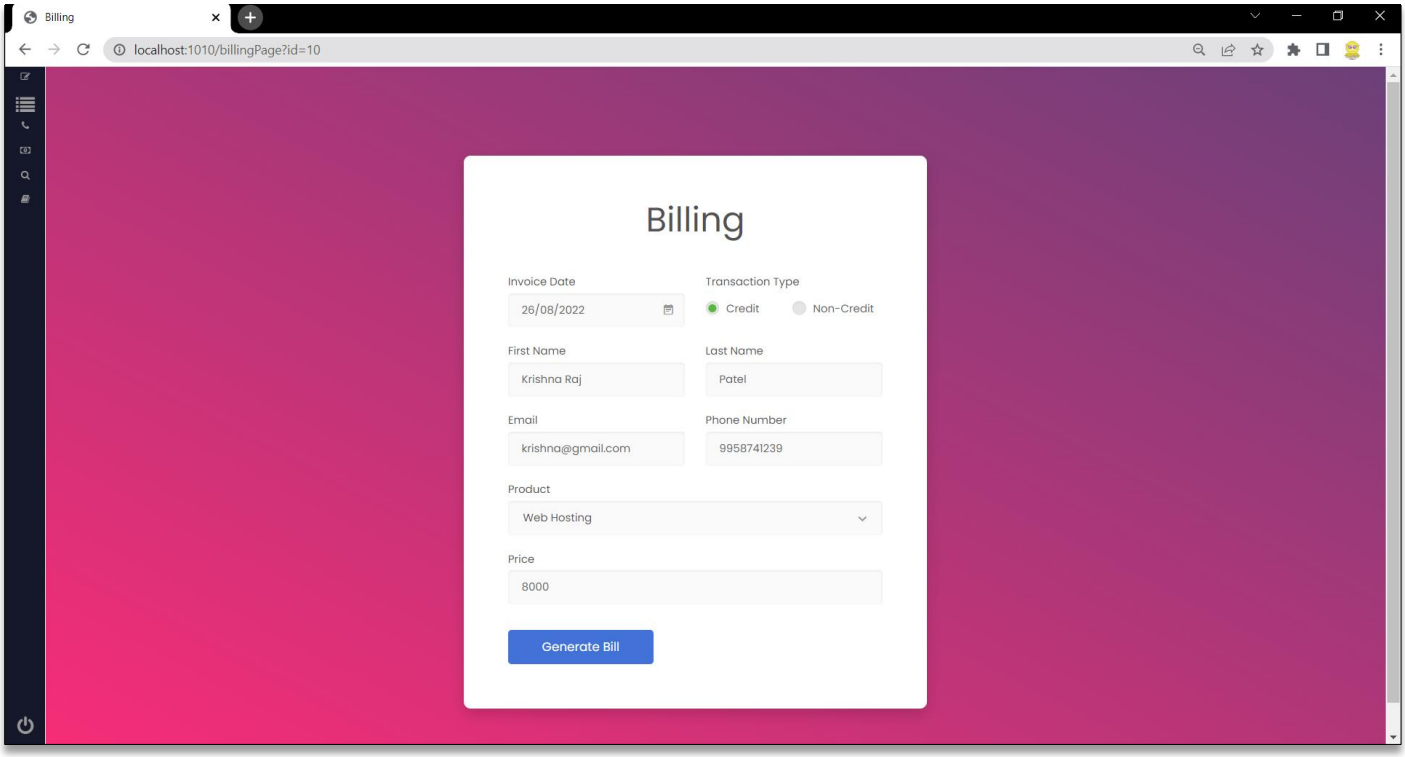


Billing Page (View Layer): "bill_generate.jsp"

```
<%@ include file="Index.jsp" %>
<!DOCTYPE html>
<html lang="en">
 <title> Billing </title>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
    <meta name="description" content="Colorlib Templates">
    <meta name="author" content="Colorlib">
    <meta name="keywords" content="Colorlib Templates">
    <link href="vendor/mdi-font/css/material-design-iconic-font.min.css" rel="stylesheet"
media="all">
    <link href="vendor/font-awesome-4.7/css/font-awesome.min.css" rel="stylesheet" me-
dia="all">
    <!-- Font special for pages-->
    <link href="https://fonts.googleapis.com/css?family=Pop-
pins:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i"
rel="stylesheet">
    <!-- Vendor CSS-->
    <link href="vendor/select2/select2.min.css" rel="stylesheet" media="all">
    <link href="vendor/datepicker/daterangepicker.css" rel="stylesheet" media="all">
    <link href="css/generateBillStyle.css" rel="stylesheet" media="all">
</head>
<body>
    <div class="page-wrapper bg-gra-02 p-t-130 p-b-100 font-poppins">
        <div class="wrapper wrapper--w680">
            <div class="card card-4">
                <div class="card-body">
                    <h2 class="title">Billing</h2>
```

```
                <form action="generateBill" method="post">
                  <div class="row row-space">
                          <div class="col-2">
                              <div class="input-group">
                                  <label class="label">Invoice Date</label>
                                  <div class="input-group-icon">
                                      <input name="date" class="input--style-4 js-date-
picker" type="text" name="birthday" required>
                                      <i class="zmdi zmdi-calendar-note input-icon js-
btn-calendar"></i>
                                  </div>
                              </div>
                          </div>
                          <div class="col-2">
                              <div class="input-group">
                                  <label class="label">Transaction Type</label>
                                  <div class="p-t-10">
                                      <label class="radio-container m-r-45">Credit
                                          <input type="radio" checked="checked"
name="gender">
                                          <span class="checkmark"></span>
                                      </label>
                                      <label class="radio-container">Non-Credit
                                          <input type="radio" name="gender">
                                          <span class="checkmark"></span>
                                      </label>
                                  </div>
                              </div>
                          </div>
                      </div>
                      <div class="row row-space">
                          <div class="col-2">
                              <div class="input-group">
                                  <label class="label">first name</label>
                                  <input class="input--style-4" type="text"
name="firstName" value="${contact.firstName}" readonly>
                              </div>
                          </div>
                          <div class="col-2">
                              <div class="input-group">
                                  <label class="label">last name</label>
                                  <input class="input--style-4" type="text" name="last-
Name" value="${contact.lastName}" readonly>
                              </div>
                          </div>
                      </div>
                      <div class="row row-space">
                          <div class="col-2">
                              <div class="input-group">
                                  <label class="label">Email</label>
                                  <input class="input--style-4" type="email"
name="email" value="${contact.email}" readonly>
                              </div>
                          </div>
```

```
                <div class="col-2">
                                <div class="input-group">
                                    <label class="label">Phone Number</label>
                                    <input class="input--style-4" type="text" name="mo-
bile" value="${contact.mobile}" readonly>
                                </div>
                            </div>
                        </div>

                        <div class="input-group">
                            <label class="label">Product</label>
                            <div class="rs-select2 js-select-simple select--no-search">
                                <select name="product" required>
                                    <option disabled="disabled" selected="se-
lected">Choose Product or Service</option>
                                     <option>Custom Software Development</option>
                                                <option>Web Hosting</option>
                                          <option>Website Development</option>
                                          <option>Cloud Storage</option>
                                          <option>Web Application Development</option>
                                          <option>Mobile Application Development</option>
                                          <option>Cloud Consulting</option>
                                          <option>DevOps Automation</option>
                                          <option>Software Prototyping</option>
                                          <option>Quality Assurance</option>
                                          <option>Systems Integration</option>
                                </select>
                                <div class="select-dropdown"></div>
                            </div>
                        </div>
                        <div class="input-group">
                            <label class="label">Price</label>
                            <div class="rs-select2 js-select-simple select--no-search">
                          <input class="input--style-4" type="number" name="price" required>
                                <div class="select-dropdown"></div>
                            </div>
                        </div>
                        <div class="p-t-15">
      <button class="btn btn--radius-2 btn--blue" type="submit">Generate Bill</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>

    <!-- Jquery JS-->
    <script src="vendor/jquery/jquery.min.js"></script>
    <!-- Vendor JS-->
    <script src="vendor/select2/select2.min.js"></script>
    <script src="vendor/datepicker/moment.min.js"></script>
    <script src="vendor/datepicker/daterangepicker.js"></script>

    <script src="js/global.js"></script>

</body>
</html>
```

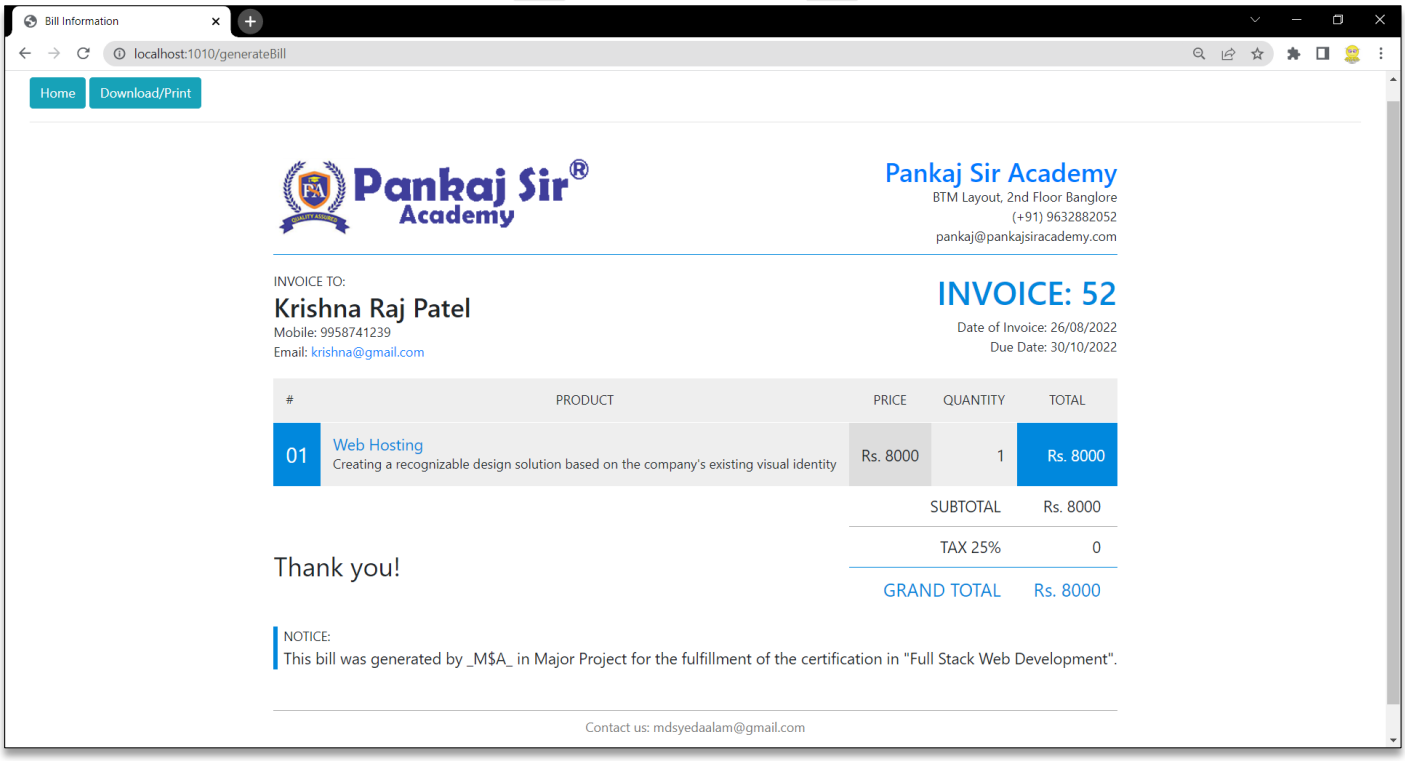


Bill Info, Invoice (View Layer): "bill_info.jsp"

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<head>
      <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
      <script src="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/boot-
strap.min.js"></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/1.3.4/jspdf.debug.js"
></script>
      <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
      <link rel="stylesheet" href="css/bill_info.css">
</head>
<title> Bill Information </title>

<div id="invoice">
         <div class="toolbar hidden-print">
           <div class="text-left">
                 <a href="/">
                 <button class="btn btn-info">
                 <i class="fa fa-file-pdf-o">
                  </i> Home</button> </a>

                 <button id="printInvoice" class="btn btn-info"><i class="fa fa-print">
                 <script type="text/javascript" src="js/bill3.js"> </script></i> Down-
load/Print</button>
           </div>
        <hr>
        </div>
```

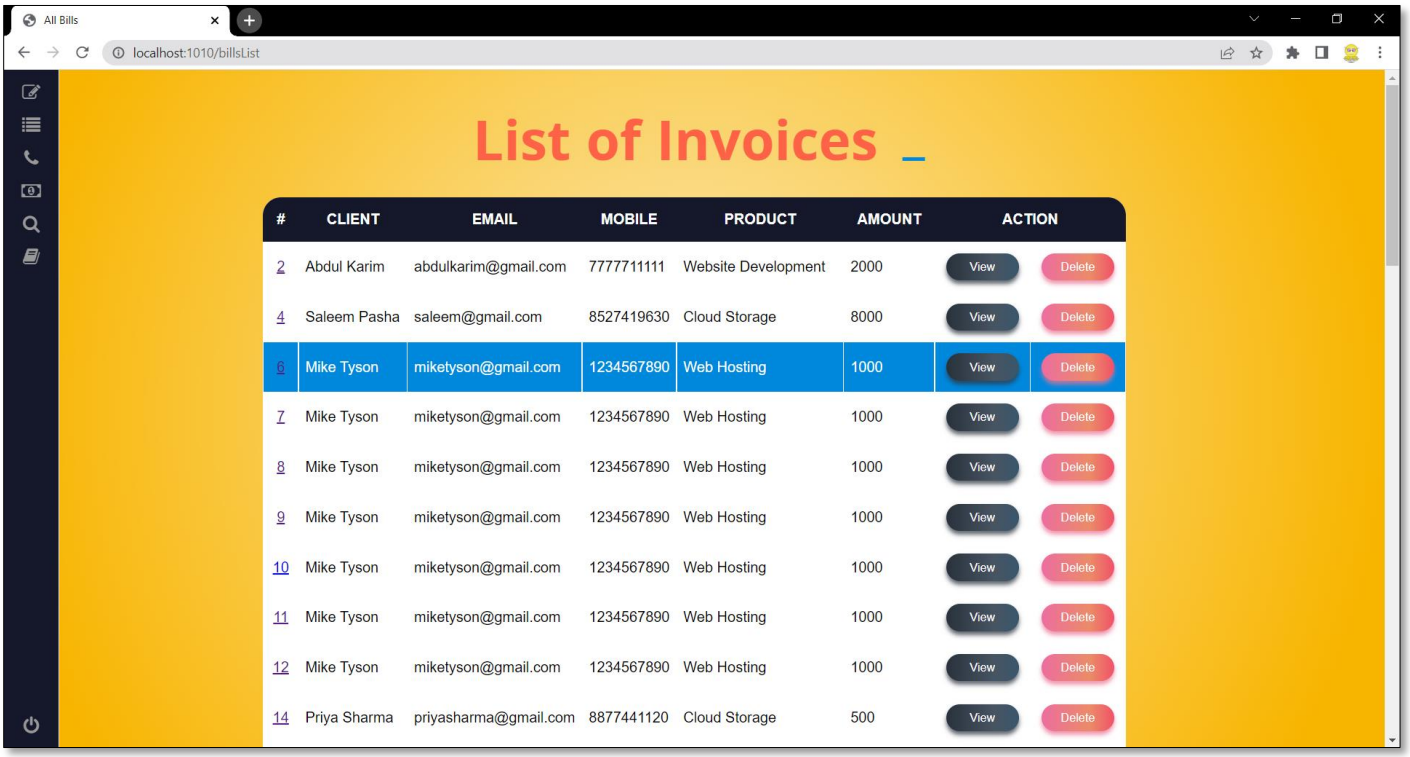
```
<div class="invoice overflow-auto">
  <div style="min-width: 600px">
      <header>
          <div class="row">
              <div class="col">
                  <a target="_blank" href="https://www.pankajsiracademy.in/">
                      <img src="images/logo.png" data-holder-rendered="true" />
                      </a>
              </div>
              <div class="col company-details">
                  <h2 class="name">
                      <a target="_blank" href="https://www.pankajsiracademy.in/">
                      Pankaj Sir Academy
                      </a>
                  </h2>
                  <div>BTM Layout, 2nd Floor Banglore</div>
                  <div>(+91) 9632882052</div>
                  <div>pankaj@pankajsiracademy.com</div>
              </div>
          </div>
      </header>
      <main>
          <div class="row contacts">
              <div class="col invoice-to">
                  <div class="text-gray-light">INVOICE TO:</div>
                  <h2 class="to"> ${bill.firstName} ${bill.lastName}</h2>
                  <div class="address">Mobile: ${bill.mobile}</div>
                  <div class="email">Email: <a
href="mailto:${bill.email}">${bill.email}</a></div>
              </div>
              <div class="col invoice-details">
                  <h1 class="invoice-id">INVOICE: ${bill.invoiceNo}</h1>
                  <div class="date">Date of Invoice: ${bill.date}</div>
                  <div class="date">Due Date: 30/10/2022</div>
              </div>
          </div>
          <table border="0" cellspacing="0" cellpadding="0">
              <thead>
                  <tr>
                      <th>#</th>
                      <th class="text-center">PRODUCT</th>
                      <th class="text-center">PRICE</th>
                      <th class="text-center">QUANTITY</th>
                      <th class="text-center">TOTAL</th>
                  </tr>
              </thead>
            <tbody>
                  <tr>
                      <td class="no">01</td>
                      <td class="text-left"><h3>${bill.product}</h3>Creating a rec-
ognizable design solution based on the company's existing visual identity</td>
                      <td class="unit">Rs. ${bill.price}</td>
                      <td class="qty">1</td>
                      <td class="total">Rs. ${bill.price}</td>
                  </tr>
              </tbody>
```

```
                <tfoot>
                    <tr>
                        <td colspan="2"></td>
                        <td colspan="2">SUBTOTAL</td>
                        <td>Rs. ${bill.price}</td>
                    </tr>
                    <tr>
                        <td colspan="2"></td>
                        <td colspan="2">TAX 25%</td>
                        <td>0</td>
                    </tr>
                    <tr>
                        <td colspan="2"></td>
                        <td colspan="2">GRAND TOTAL</td>
                        <td>Rs. ${bill.price}</td>
                    </tr>
                </tfoot>
            </table>
            <div class="thanks">Thank you!</div>
            <div class="notices">
                <div>NOTICE:</div>
                <div class="notice">This bill was generated by _M$A_ in Major Project
for the fulfillment of the certification in "Full Stack Web Development".</div>
            </div>
        </main>
        <footer>
            Contact us: mdsyedaalam@gmail.com
        </footer>
    </div>
    <div></div>
</div>
```



List of All Bills (View Layer): "billslist.jsp"

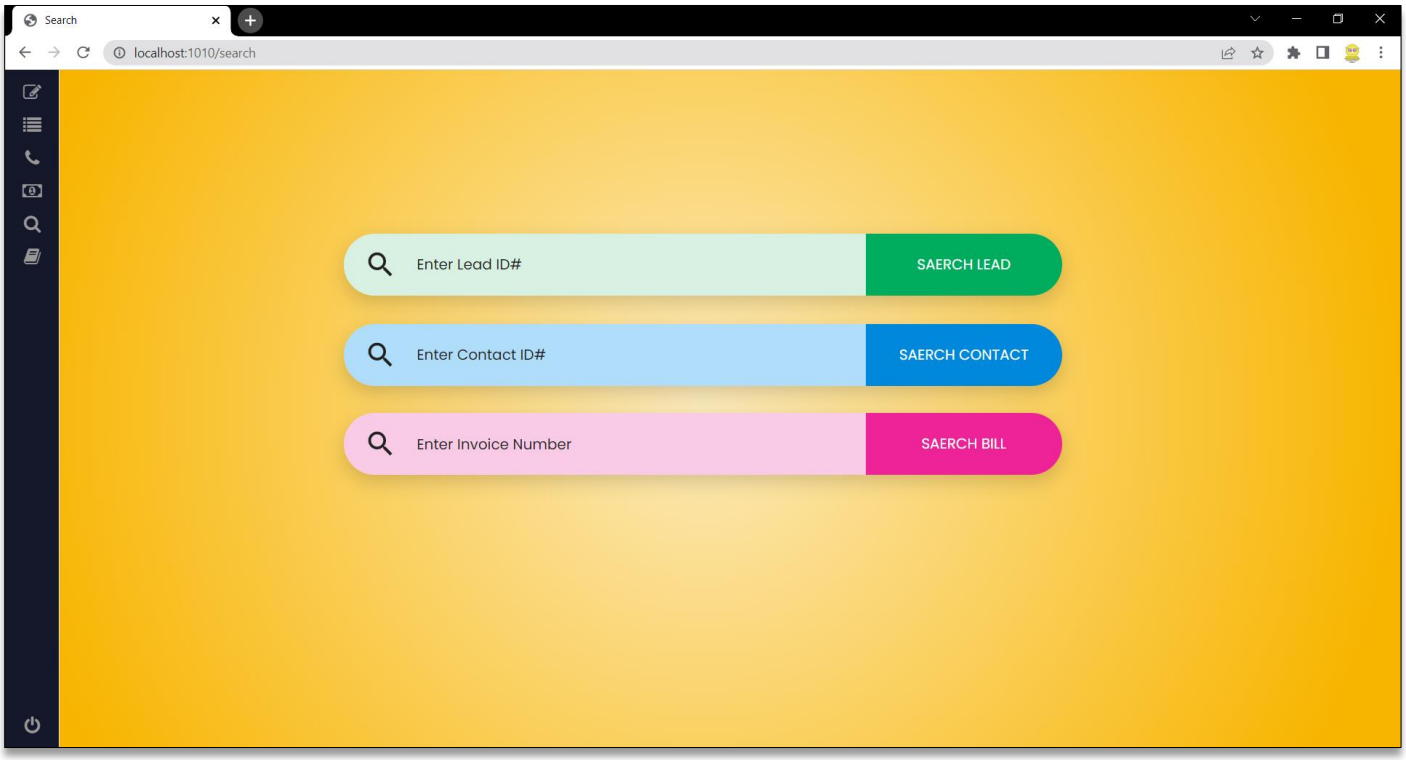
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core"%> <!-- JSTL Directive
Tag -->
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="/css/list_style.css">
<meta charset="ISO-8859-1">
<title> All Bills </title>
</head>
<body class="heading-body">
            <div class='console-container'><span id='text'></span>
            <script type="text/javascript" src="js/billList.js"></script>
            <div class='console-underscore' id='console'>&#95;</div></div>
</body>
      <body>
            <div class="container">
                  <table class="hoverTable">
                        <thead>
                              <tr>
                                    <th>#</th>
                                    <th>CLIENT</th>
                                    <th>EMAIL</th>
                                    <th>MOBILE</th>
                                    <th>PRODUCT</th>
                                    <th>AMOUNT</th>
                                    <th colspan="2">ACTION</th>
                              </tr>
                        </thead>
                        <tbody>
                              <c:forEach var="bill" items="${bills}">
                              <tr>
                                    <td style="text-align:center"><a href="billInfo?in-
voiceNo=${bill.invoiceNo}">${bill.invoiceNo}</a></td>
                                    <td>${bill.firstName} ${bill.lastName}</td>
                                    <td>${bill.email}</td>
                                    <td>${bill.mobile}</td>
                                    <td>${bill.product}</td>
                                    <td>${bill.price}</td>
                                    <td>
                                          <div class="social">
                                                <a href="billInfo?invoiceNo=${bill.in-
voiceNo}"><input class="btn-hover color-8" type="submit" value="View"></a>
                                    </td>
                                    <td>
                                          <a href="deleteBill?invoiceNo=${bill.in-
voiceNo}"><input class="btn-hover color-10" type="submit" value="Delete"></a>
                                           </div>
                                    </td>
                              </tr>
                        </c:forEach>
                        </tbody>
                  </table>
            </div>
      </body>
</html>
```



Search Page (View Layer): "search_page.jsp"

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ include file="Index.jsp" %>
<!DOCTYPE html>
<html>
<title> Search </title>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="author" content="colorlib.com">
    <link href="https://fonts.googleapis.com/css?family=Poppins" rel="stylesheet" />
    <link href="css/search_style.css" rel="stylesheet" />
  </head>
  <body>
    <div class="s130">
      <form action="searchLead" method="post">
        <div class="inner-form">
          <div class="input-field first-wrap">
            <div class="svg-wrapper">
              <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0
24 24">
                <path d="M15.5 14h-.79l-.28-.27C15.41 12.59 16 11.11 16 9.5 16 5.91 13.09
3 9.5 3S3 5.91 3 9.5 5.91 16 9.5 16c1.61 0 3.09-.59 4.23-1.57l.27.28v.79l5 4.99L20.49
19l-4.99-5zm-6 0C7.01 14 5 11.99 5 9.5S7.01 5 9.5 5 14 7.01 14 9.5 11.99 14 9.5
14z"></path>
              </svg>
            </div>
            <input id="search" name="id" type="text" placeholder="Enter Lead ID#" />
          </div>
```

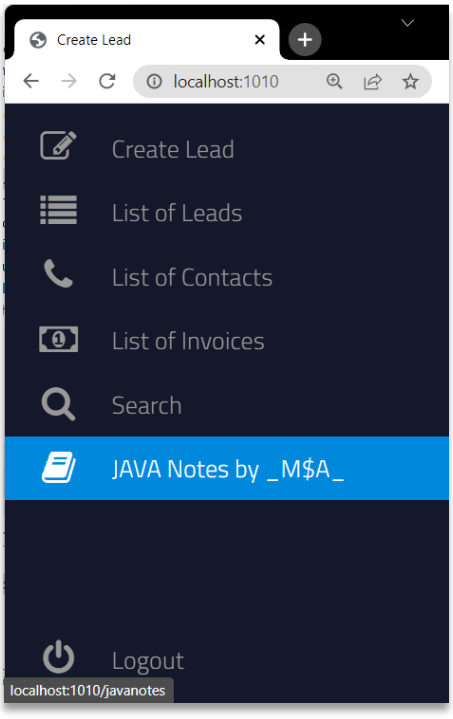
```
<div class="input-field second-wrap">
            <input class="btn-search" type="submit" value="SAERCH LEAD"></input>
          </div>
        </div>
      </form>
    </div>
     <div class="s130-1">
      <form action="searchContact" method="post">
        <div class="inner-form">
          <div class="input-field first-wrap">
            <div class="svg-wrapper">
              <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0
24 24">
                <path d="M15.5 14h-.79l-.28-.27C15.41 12.59 16 11.11 16 9.5 16 5.91 13.09
3 9.5 3S3 5.91 3 9.5 5.91 16 9.5 16c1.61 0 3.09-.59 4.23-1.57l.27.28v.79l5 4.99L20.49
19l-4.99-5zm-6 0C7.01 14 5 11.99 5 9.5S7.01 5 9.5 5 14 7.01 14 9.5 11.99 14 9.5
14z"></path>
              </svg>
            </div>
            <input name="id" type="text" placeholder="Enter Contact ID#" />
          </div>
          <div class="input-field second-wrap">
            <input class="btn-search" type="submit" value="SAERCH CONTACT"></input>
          </div>
        </div>
      </form>
    </div>
    <div class="s130-2">
      <form action="billInfo" method="post">
        <div class="inner-form">
          <div class="input-field first-wrap">
            <div class="svg-wrapper">
              <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0
24 24">
                <path d="M15.5 14h-.79l-.28-.27C15.41 12.59 16 11.11 16 9.5 16 5.91 13.09
3 9.5 3S3 5.91 3 9.5 5.91 16 9.5 16c1.61 0 3.09-.59 4.23-1.57l.27.28v.79l5 4.99L20.49
19l-4.99-5zm-6 0C7.01 14 5 11.99 5 9.5S7.01 5 9.5 5 14 7.01 14 9.5 11.99 14 9.5
14z"></path>
              </svg>
            </div>
            <input name="invoiceNo" type="text" placeholder="Enter Invoice Number" />
          </div>
          <div class="input-field second-wrap">
            <input class="btn-search" type="submit" value="SAERCH BILL"></input>
          </div>
        </div>
      </form>
    </div>
</html>
```



Index (View Layer): "Index.jsp"

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head><link rel="stylesheet" type="text/css" href="/css/nav_style.css">
  <head>

  </head>
  <body><div class="area"></div><nav class="main-menu">
            <ul>
                <li>
                    <a href="/">
                        <i class="fa fa-edit"></i>
                        <span class="nav-text">
                            Create Lead
                        </span>
                    </a>

                </li>
                <li class="has-subnav">
                    <a href="leadsList">
                        <i class="fa fa-list fa-2x"></i>
                        <span class="nav-text">
                            List of Leads
                        </span>
                    </a>
                    <li class="has-subnav">
                    <a href="contactsList">
                       <i class="fa fa-phone"></i>
                        <span class="nav-text">
                            List of Contacts
                        </span>
                    </a>
                </li>
```

```
                <li class="has-subnav">
                  <a href="billsList">
                     <i class="fa fa-money"></i>
                      <span class="nav-text">
                          List of Invoices
                      </span>
                  </a>
              </li>
              <li class="has-subnav">
                  <a href="search">
                     <i class="fa fa-search"></i>
                      <span class="nav-text">
                          Search
                      </span>
                  </a>
                <li class="has-subnav">
                  <a href="javanotes">
                     <i class="fa fa-book"></i>
                      <span class="nav-text">
                          JAVA Notes by _M$A_
                      </span>
                  </a>
              </li>
          </ul>

          <ul class="logout">
              <li>
                 <a href="#">
                       <i class="fa fa-power-off fa-2x"></i>
                      <span class="nav-text">
                          Logout
                      </span>
                  </a>
              </li>
          </ul>
      </nav>
</body>
  </html>
```



Application Properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/crmapp
spring.datasource.username=root
spring.datasource.password=Test

spring.jpa.hibernate.ddl-auto=update

spring.mvc.view.prefix=/WEB-INF/jsps/
spring.mvc.view.suffix=.jsp

server.port=1010
```

pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.7.2</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.crmapp</groupId>
	<artifactId>crmapp</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>crmapp</name>
	<description>Customer Relationship Management App (ZOHO CRM)</description>
	<properties>
		<java.version>1.8</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
```

```
		<dependency>
			<groupId>org.apache.tomcat.embed</groupId>
			<artifactId>tomcat-embed-jasper</artifactId>
			<scope>provided</scope>
		</dependency>
		<dependency>
		    <groupId>javax.servlet</groupId>
		    <artifactId>jstl</artifactId>
		    <version>1.2</version><!--$NO-MVN-MAN-VER$-->
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
		</dependency>
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>
</project>
```