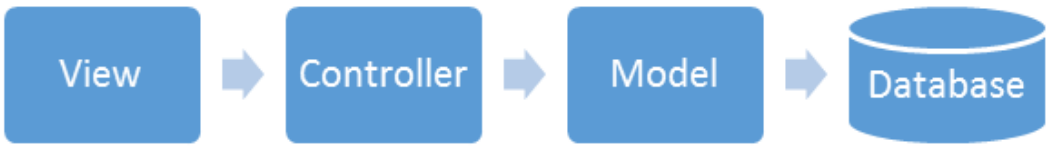
MVC is an architecture that separates business logic, presentation and data. In MVC,

- M stands for Model
- V stands for View
- C stands for controller.

In the below image, the **Controller** connects with **Model** and fetches the data from the **Database** and sends to the View layer.



MVC Architecture Diagram

Model Layer: Model layer connects with the database as well and stores the data into a database which is connected to it.

View Layer: This is for displaying data into various UI outputs like HTML, JSP, etc. into it. The view layer fetches data from the controller and displays it. This view layer shows the data on the UI of the application.

Controller Layer: It acts as an interface between View and Model. It receives the requests from the view layer and processes the requests and does the necessary validation for the request.

Advantages of MVC Architecture

The advantages of MVC are:

- Easy to maintain

- Easy to extend
- Easy to test
- Navigation control is centralized

Example of JSP Application Design with MVC Architecture

In this example, we are going to show how to use MVC architecture in JSP.

- We are taking the example of a form with two variables “email” and “password” which is our view layer.
- Once the user enters email, and password and clicks on submit then the action is passed in mvc_servlet where email and password are passed.
- This mvc_servlet is a controller layer. Here in mvc_servlet the request is sent to the bean object which acts as a model layer.
- The email and password values are set into the bean and stored for further purpose.
- From the bean, the value is fetched and shown in the view layer.

Mvc_example.jsp

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

<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>MVC Guru Example</title>
</head>
<body>
<form action="Mvc_servlet" method="POST">
Email: <input type="text" name="email">
<br />
```

```
Password: <input type="text" name="password" />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

Explanation of the code:

```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2      pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4
5  <html>
6  <head>
7  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
8  <title>MVC Guru Example</title>
9  </head>
10 <body>
11 <form action="Mvc_servlet" method="POST">
12 Email: <input type="text" name="email">
13 <br />
14 Password: <input type="text" name="password" />
15 <input type="submit" value="Submit" />
16 </form>
17 </body>
18 </html>
19
```

View Layer:

Code Line 10-15: Here we are taking a form which has two fields as parameter “email” and “password” and this request need to be forwarded to a controller Mvc_servlet.java, which is passed in action.The method through which it is passed is POST method.

Mvc_servlet.java

```
package demotest;

import java.io.IOException;
```

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Mvc_servlet
 */
public class Mvc_servlet extends HttpServlet {
     private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Mvc_servlet() {
        super();
        // TODO Auto-generated constructor stub
    }


     protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
           // TODO Auto-generated method stub
           String email=request.getParameter("email");
        String password=request.getParameter("password");

        TestBean testobj = new TestBean();
        testobj.setEmail(email);
        testobj.setPassword(password);
        request.setAttribute("gurubean",testobj);
        RequestDispatcher
rd=request.getRequestDispatcher("mvc_success.jsp");
        rd.forward(request, response);
     }

}
```

Explanation of the code:

```java
import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Mvc_servlet
 */
public class Mvc_servlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Mvc_servlet() {
        super();
        // TODO Auto-generated constructor stub
    }


    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletExcept
        IOException {
        // TODO Auto-generated method stub
        String email=request.getParameter("email");
        String password=request.getParameter("password");

        TestBean testobj = new TestBean();
        testobj.setEmail(email);
        testobj.setPassword(password);
        request.setAttribute("gurubean",testobj);
        RequestDispatcher rd=request.getRequestDispatcher("mvc_success.jsp");
        rd.forward(request, response);
    }

}

```

Controller layer

Code Line 14:mvc_servlet is extending HttpServlet.

Code Line 26: As the method used is POST hence request comes into a doPost method of the servlet which process the requests and saves into the bean object as testobj.

Code Line 34: Using request object we are setting the attribute as gurubean which is assigned the value of testobj.

Code Line 35: Here we are using request dispatcher object to pass the success message to mvc_success.jsp

TestBean.java

```
package demotest;

import java.io.Serializable;
```

```
public class TestBean implements Serializable{

	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	private String email="null";
	private String password="null";


}
```

Explanation of the code:

```
1  package demotest;
2
3  import java.io.Serializable;
4
5  public class TestBean implements Serializable{
6
7      public String getEmail() {
8          return email;
9      }
10     public void setEmail(String email) {
11         this.email = email;
12     }
13     public String getPassword() {
14         return password;
15     }
16     public void setPassword(String password) {
17         this.password = password;
18     }
19     private String email="null";
20     private String password="null";
21
22
23 }
24
```

Model Layer:

Code Line 7-17: It contains the getters and setters of email and password which are members of Test Bean class

Code Line 19-20: It defines the members email and password of string type in the bean class.

Mvc_success.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@page import="demotest.TestBean"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Guru Success</title>
</head>
<body>
<%
TestBean testguru=(TestBean)request.getAttribute("gurubean");
out.print("Welcome, "+testguru.getEmail());
%>
</body>
</html>
```

Explanation of the code:

Code Line 12: we are getting the attribute using request object which has been set in the doPost method of the servlet.

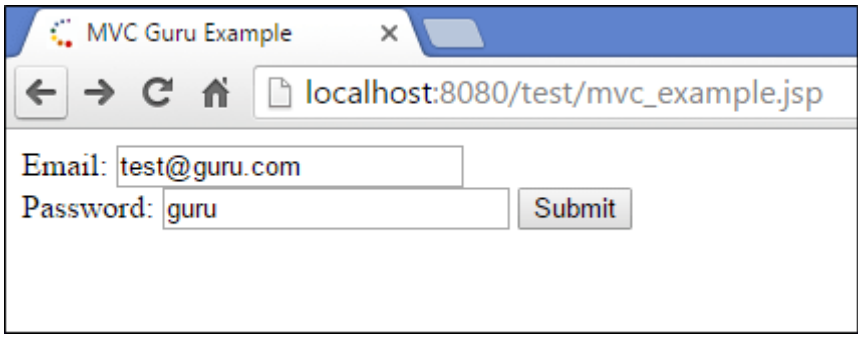
Code Line 13: We are printing the welcome message and email id of which have been saved in the bean object

Output:

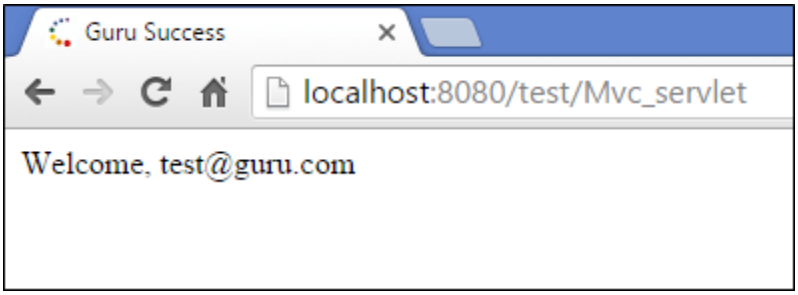
When you execute the above code, you get the following output:

When you click on mvc_example.jsp you get the form with email and password with the submit button.

Once you enter email and password to the form and then click on submit



After clicking on submit the output is shown as below



Output:

When you enter email and password in screen and click on submit then, the details are saved in TestBean and from the TestBean they are fetched on next screen to get the success message.