

Web Component Development Using Java

Session: 10

Model-View-Controller Architecture



Objectives

- ❖ Describe the use of JSP models in Web applications
- ❖ Explain JSP Model 1
- ❖ Explain JSP Model 2
- ❖ Explain the Model-View-Controller architecture
- ❖ Explain the relationship between the components of MVC
- ❖ Explain Controller and its purpose in MVC
- ❖ Explain View and its purpose in MVC
- ❖ Explain Model and its purpose in MVC
- ❖ Develop a Web application based on MVC architecture

Introduction

- ❖ Sun Microsystems has provided the JSP specification to address the problem of tightly-coupled presentation and business logic in Servlets.
- ❖ Based on the popularity and benefits of JSP in Web development, the commonly used approaches using JSP were identified.
- ❖ These approaches are also referred to as JSP models.



- ❖ There are two types of programming models for developing Web application.

JSP Model 1

- The Model 1 architecture is very simple.
- The HTML or JSP page sends request along with the data to Web container.
- The Web container invokes the mapped Servlet which handles all responsibilities for the request.
- The responsibilities include processing the request, validating data, handling the business logic, and generating a response back to browser.

JSP Model 2 (Model-View-Controller)

- It provides a clear separation of application responsibilities.
- A central servlet, known as the Controller, receives all requests for the application from JSP.
- The Controller works with the Model to prepare any data needed by the View and forwards the data back to the JSP.
- The business and presentation logic are separated from each other, which help to reuse the logic.

JSP Model 1 Architecture 1-4

- ❖ Model 1 architecture enables the Web designers to develop Web applications such that it separates business logic from presentation logic.



The diagram illustrates the JSP Model 1 Architecture. It consists of two main components: Business Logic and Presentation Logic. The Business Logic component is represented by a red downward-pointing arrow on the left, with a corresponding rounded rectangular box to its right. The Presentation Logic component is represented by a green downward-pointing arrow on the left, with a corresponding rounded rectangular box to its right. The boxes contain bulleted descriptions of each logic type. A large, faint watermark 'For Approval Centre Use Only' is oriented diagonally across the center of the slide.

Business Logic

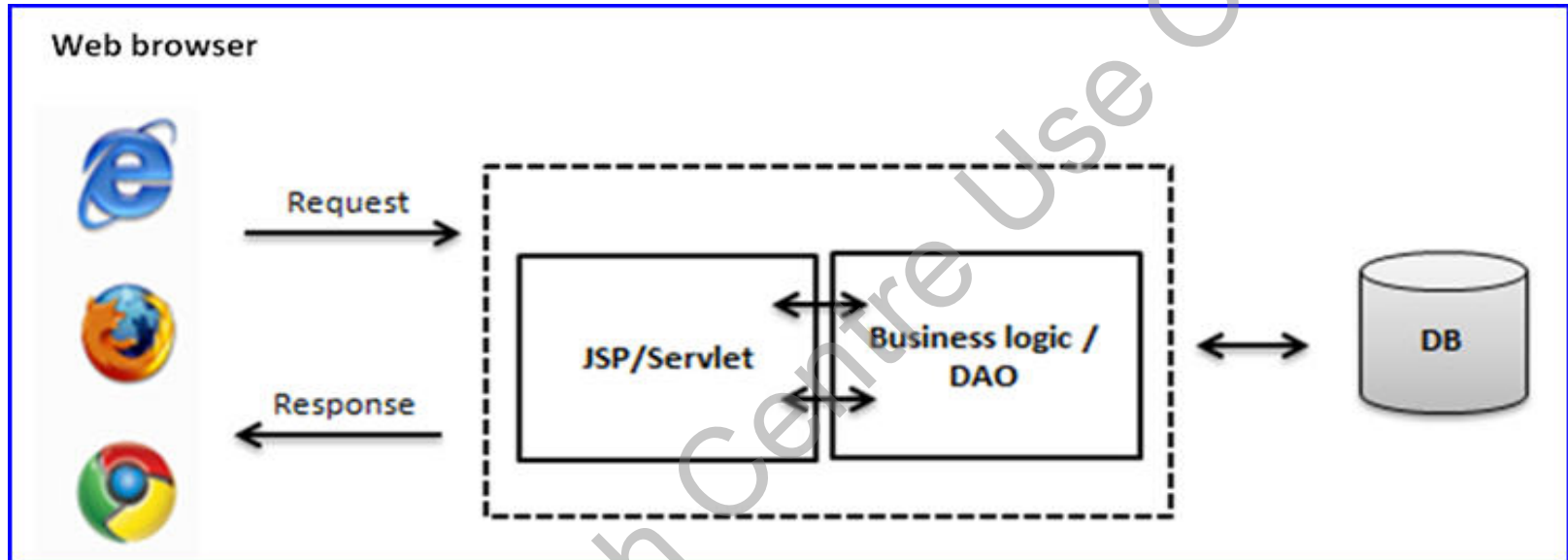
- It deals with the method of modeling real world business objects such as accounts, loans, travel, and so on in the application.
- It also deals with the storage mechanism for these objects, object interactions, access, and update rights for them.

Presentation Logic

- It deals with methods of displaying these objects.
- For example, decisions related to displaying user accounts in a list form.

JSP Model 1 Architecture 2-4

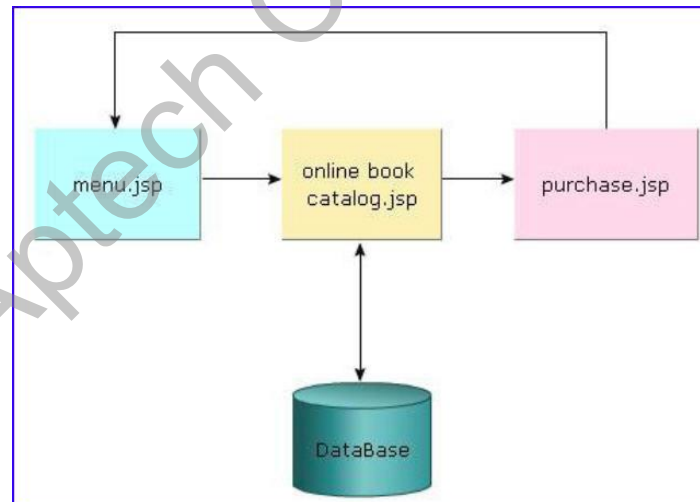
- ❖ Figure depicts the Model 1 architecture.



- ❖ In this model, there is no extra Servlet involved in the process.
- ❖ The client request is sent directly to a JSP page, which may communicate with JavaBeans or other services, but ultimately the JSP page selects the next page for the client to view.

JSP Model 1 Architecture 3-4

- ❖ JSP Model 1 architecture has a page-centric architecture.
- ❖ Page-centric architecture:
 - ❑ The application is composed of a series of interrelated JSP pages and these JSP pages handle all aspects of application.
 - ❑ The business process logic and control decisions are hard-coded inside JSP pages in the form of JavaBeans, scriptlets, and expressions.
- ❖ Figure shows an example of JSP Model 1 architecture for an online shopping Web application with all JSP pages.



❖ **Advantages of JSP Model 1:**

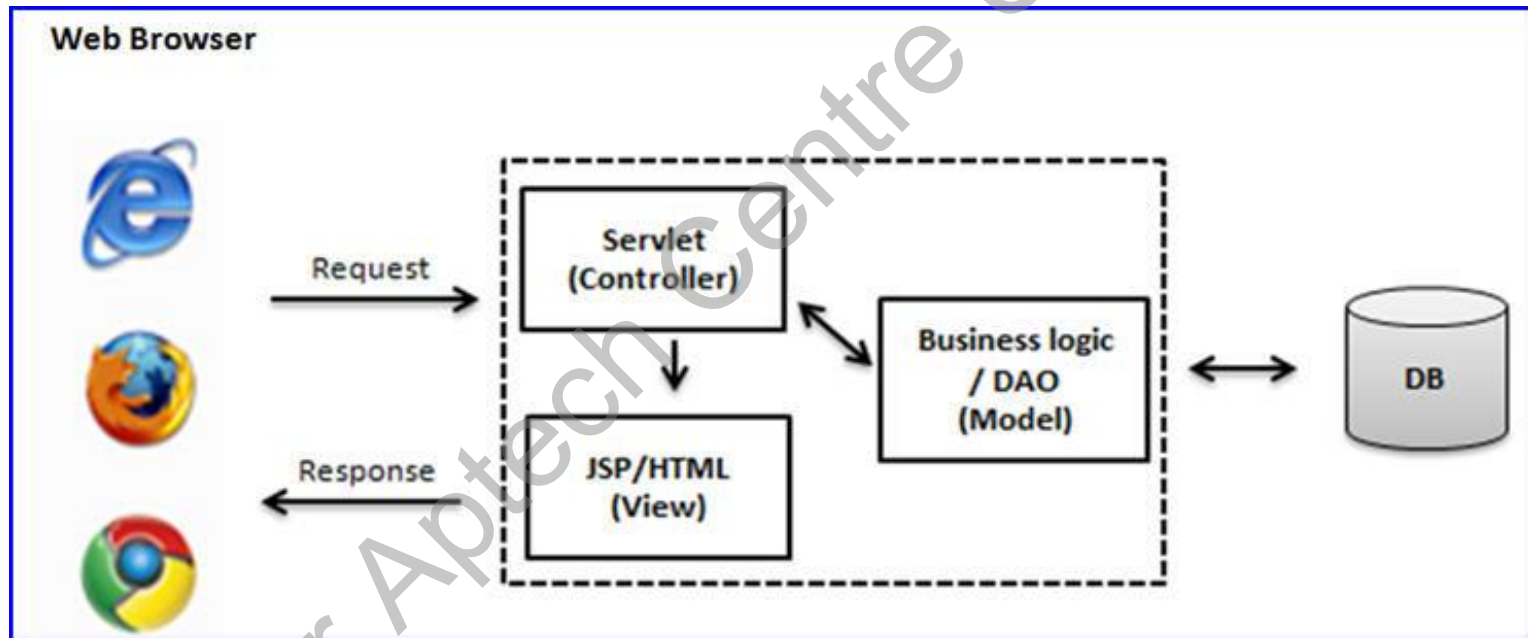
- ❑ It makes development easier as there are no Servlets involved in the application.

❖ **Disadvantages of JSP Model 1:**

- ❑ As business logic and presentation logic are tied together, it is difficult for building and maintaining a complex enterprise application.
- ❑ Each of the JSP pages is individually responsible for control logic, application logic, and also to present results to the user. This makes the JSP Model 1 more dependent and less extensible.

JSP Model 2 Architecture 1-3

- ❖ Model 2 architecture is an approach used for developing a Web application.
- ❖ It separates the Business logic from the Presentation logic.
- ❖ The Model 2 has an additional component - a Controller.
- ❖ Figure shows the Model 2 architecture.



JSP Model 2 Architecture 2-3

- ❖ In Model 2 architecture, a Servlet acts as a Controller and therefore, it has a Servlet-centric architecture.
- ❖ **Servlet - Controller:**
 - ❑ Is responsible to process the incoming request and instantiate a Model - a Java object or a bean to compute the business logic.
 - ❑ Is responsible for deciding to which JSP page the request should be forwarded.
- ❖ **JSP - View:**
 - ❑ Is responsible for handling the View component.
 - ❑ Retrieves the objects created by the Servlet.
 - ❑ Extracts dynamic content for insertion within a template for display.

JSP Model 2 Architecture 3-3

❖ Advantages of Model 2:

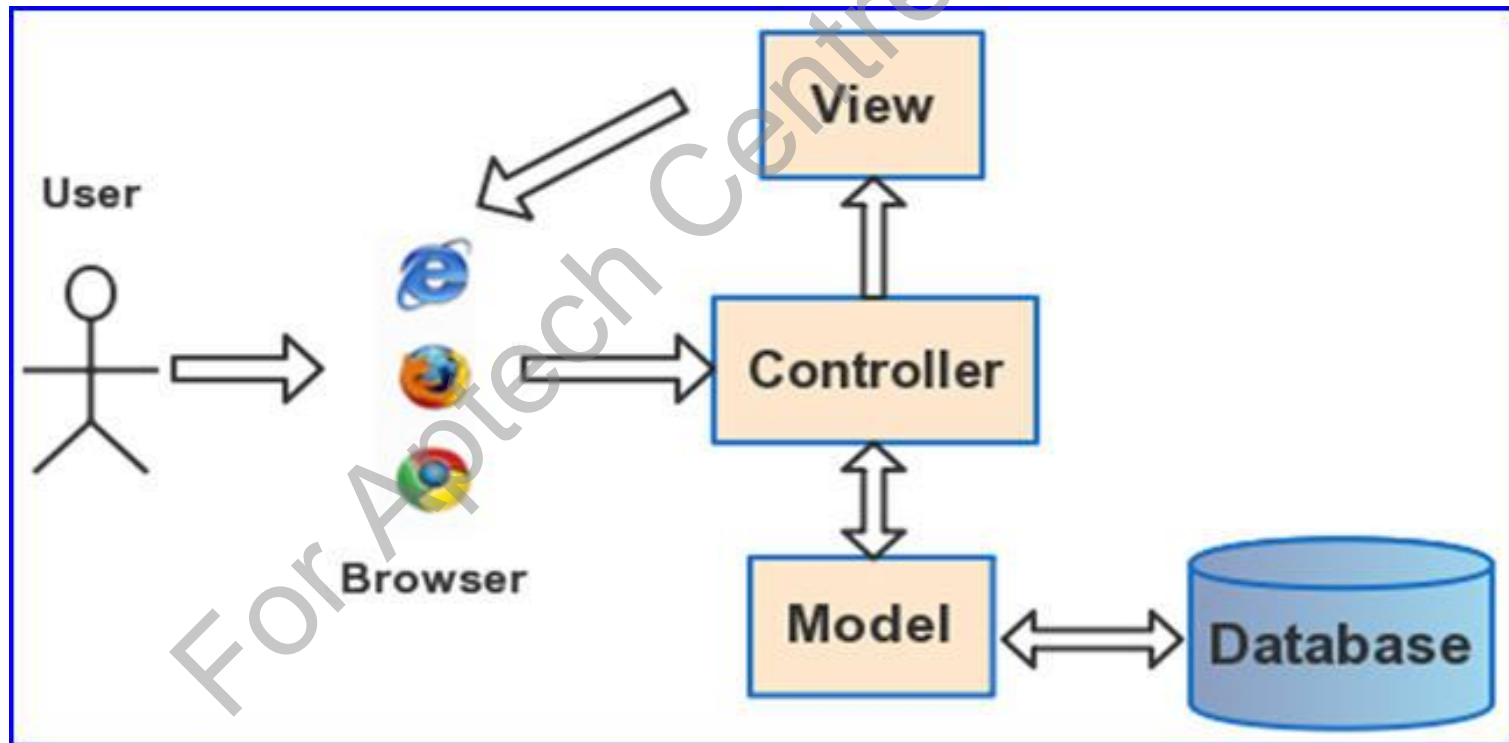
Web applications based on this model are easier to maintain and extendable.

Testing is easy in model 2.

For Aptech Centre Use Only

Model-View-Controller (MVC) 1-2

- ❖ MVC is a software architectural pattern.
- ❖ This pattern divides the application logic from User Interface.
- ❖ The division permits independent development, testing, and maintenance of each component.
- ❖ Figure shows a brief overview of the components under the MVC architecture.



Model-View-Controller (MVC) 2-2

- ❖ MVC model divides the Web based application into three layers:

Controller

- Manages the flow of data between the Model layer and the View layer.
- Calls methods in the Model to fulfil the requested action.
- After the action has been taken on the data in Model, the Controller is responsible for redirecting the appropriate view to the user.

View

- Is used to generate the response to the browser, what the user sees.
- Are simple JSP or HTML pages.

Model

- Is a layer between the Controller and the database.
- Contains the business logics and functions that manipulate the business data.
- Can access the functionalities encapsulated in the Model.

Relationships between Components

- ❖ The different relationships between MVC components are as follows:

View and Controller Relationship

- In this relationship, the Controller is responsible for creating and selecting views.

Model and View Relationship

- In this relationship, the View depends on the Model.
- If a change is made to the Model, then there might be a requirement to make parallel changes in the View.

Model and Controller Relationship

- In this relationship, the Controller is dependent on the Model.
- If a change is made to the model interface, then there might be a requirement to make parallel changes to the Controller.

MVC in Web Applications

- ❖ While developing a Web application using MVC architecture, the different components and their roles are as follows:

Model encapsulates data and business logic using JavaBean components or Plain Old Java Object (POJO), a database API, or an XML file.

View shows the current state of the Model using an HTML or a JSP page.

Controller updates the state of the Model and generates one or more views using Servlet.

Implementation of MVC Pattern 1-8

- ❖ **Scenario:** In a Web application, you have to develop login service which will validate login details and accordingly display the appropriate JSP page.
- ❖ To design such service, we will use the MVC pattern.
- ❖ The code snippet develops the JSP pages required to handle the presentation and result pages.

```
<!-- login.jsp -->

<html>
  <head>
    <title>MVC Example</title>
  </head>

  <body>
    <form action="LoginController" method="post">
      Enter username : <input type="text"
name="username"> <BR>
      Enter password : <input type="password"
name="password"> <BR>
      <input type="submit" />
    </form>
  </body>
</html>
```


Implementation of MVC Pattern 2-8

```
<!-- success.jsp -->
<html>
  <head>
    <title>Success</title>
  </head>
  <body>
    Welcome, you have successfully login.
  </body>
</html>
```

```
<!-- error.jsp -->
<html>
  <head>
    <title>Error</title>
  </head>
  <body>
    Login failed, please try again.
  </body>
</html>
```

Implementation of MVC Pattern 3-8

- ❖ Next, we will design a Controller that takes request from the user, this is the Servlet class.
- ❖ The servlet class calls the Model which is JavaBean.
- ❖ The code snippet demonstrates the design of the Controller named `LoginController.java`.

```
package com.mvc.Controller;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.mvc.model.LoginModel;

public class LoginController extends HttpServlet {
    RequestDispatcher rd = null
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {

        // Receive the values from the request parameters
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Instantiate the LoginModel JavaBean
        LoginModel login = new LoginModel();
```

Implementation of MVC Pattern 4-8

```
// Verify the login credentials from model
String result = login.authenticate(username, password);

/**
Dispatch the control based on the value of the
result variable **/

if (result.equals("success")) {
    rd = request.getRequestDispatcher("/success.jsp");
} else {
    rd = request.getRequestDispatcher("/error.jsp");
}
// Forward the response to appropriate JSP
rd.forward(request, response);
}

}
```

Implementation of MVC Pattern 5-8

- ❖ The code snippet demonstrates the design of LoginModel class.

```
package com.mvc.model;

public class LoginModel {
    public String authenticate(String username, String
password) {

// Validate the login credentials with the values
        if (("username".equalsIgnoreCase(username))
            && ("password".equals(password))) {
            return "success";
        } else {
            return "failure";
        }
    }
}
```

Implementation of MVC Pattern 6-8

- ❖ The code snippet shows the deployment descriptor, `web.xml` used to configure the servlet in the Login application.

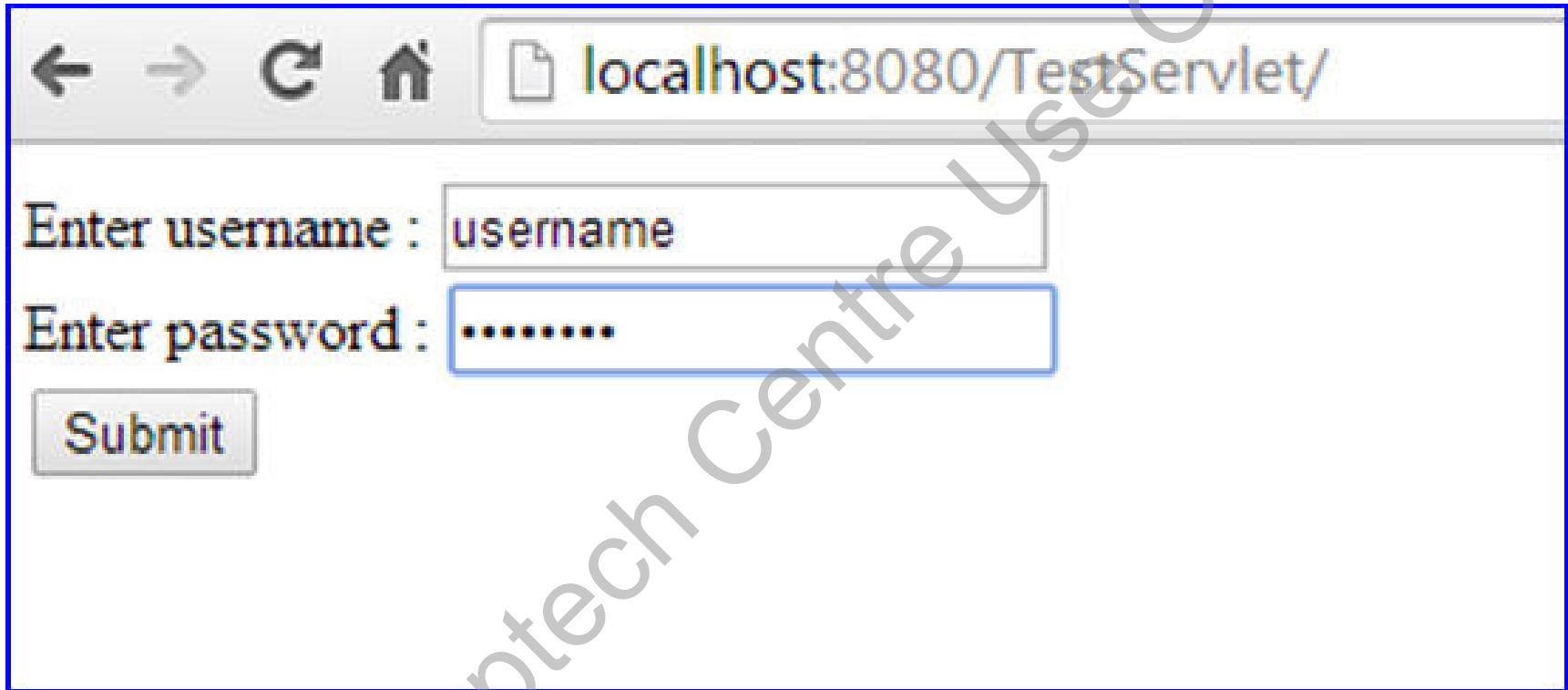
```
<!-- web.xml -->
<?xml version="1.0" encoding="UTF-8"?>
. . .
<servlet>
    <servlet-name>LoginController</servlet-name>
    <servlet-class>com.mvc.Controller.LoginController</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>LoginController</servlet-name>
    <url-pattern>/LoginController</url-pattern>
</servlet-mapping>

<welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

Implementation of MVC Pattern 7-8

- ❖ Figure shows the request to the URL `localhost:8080/TestServlet/`.



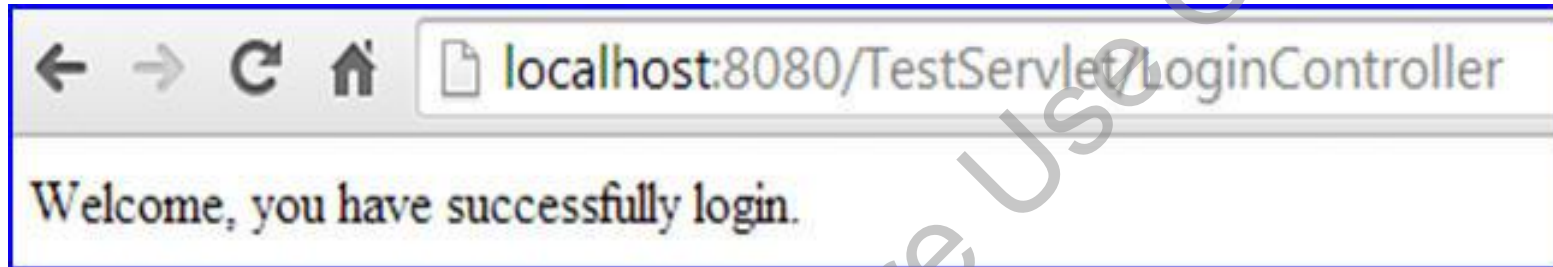
The screenshot shows a web browser window with the address bar displaying `localhost:8080/TestServlet/`. The page content includes a login form with the following elements:

- A label "Enter username :" followed by a text input field containing the text "username".
- A label "Enter password :" followed by a password input field represented by a series of dots.
- A "Submit" button located below the password field.

- ❖ This displays the `login.jsp` with the username and password fields.

Implementation of MVC Pattern 8-8

- ❖ Figure shows the `success.jsp`.



Summary

- ❖ The JSP specification presents two approaches for developing Web applications namely, JSP Model I and JSP Model II.
- ❖ JSP Model II is also known as MVC.
- ❖ MVC is a software design pattern, which can be used to design medium and large sized applications.
- ❖ MVC has three components as follows:
 - ❑ Model
 - ❑ View
 - ❑ Controller
- ❖ In MVC Web application, servlet acts as Controller, which receives the request from client.
- ❖ The View handles presentation of the content on the Web page and could be an HTML file or a JSP file.
- ❖ The Model component contains the business logics and functions that manipulate the business data.