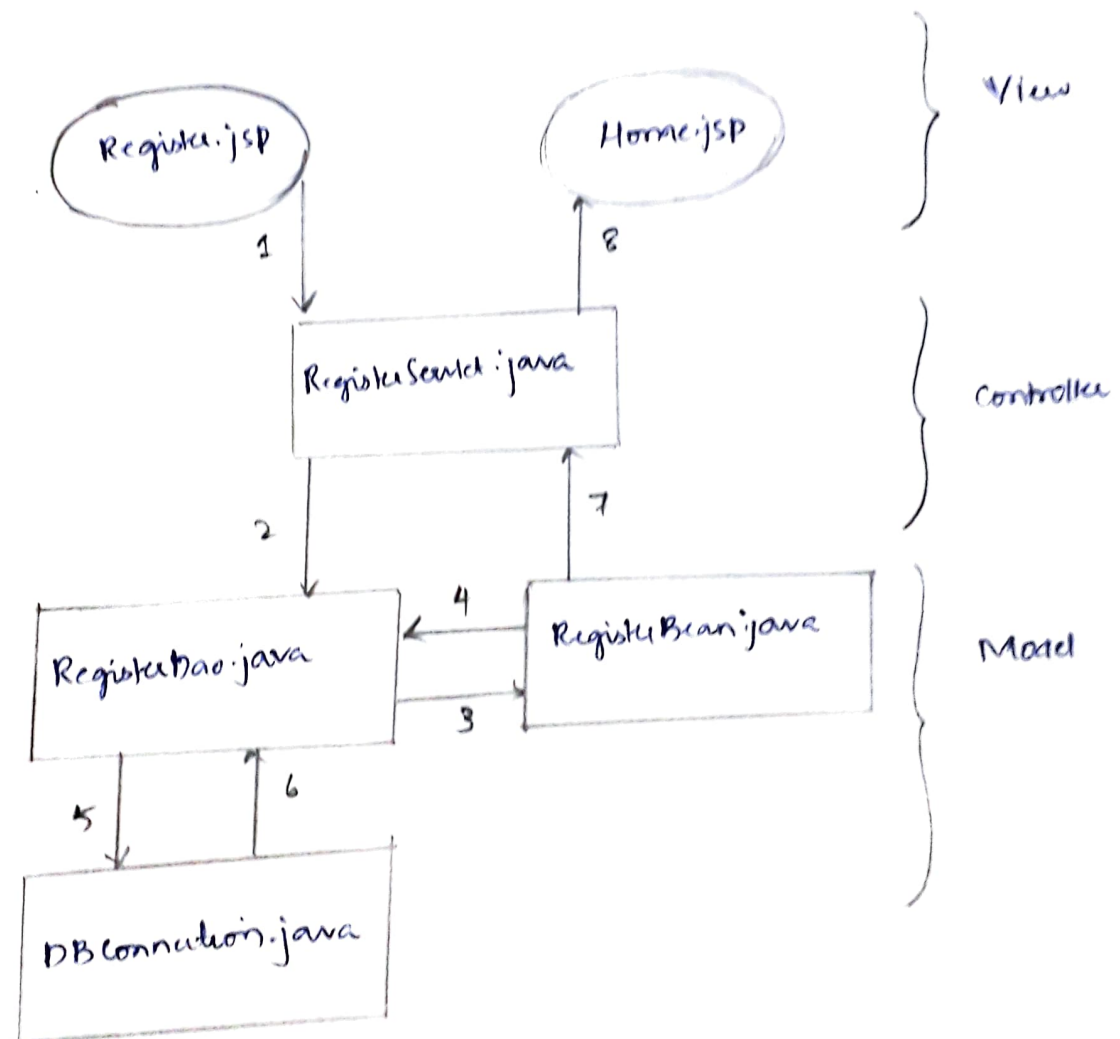


1. Write and execute Student Registration web app using Servlet & MySQL?

Sol:- This application is developed using java, servlet, jsp and mysql database server. This java registration form follows Model View Controller (MVC) architecture.



Model View controller Architecture.

## RegistrationMVC

 Deployment Descriptor: RegistrationMVC

 Java Resources: src

 com.mvc.bean


 RegisterBean.java

 com.mvc.controller

 RegisterServlet.java

 com.mvc.dao

 RegisterDao.java

 com.mvc.util

 DBConnection.java

 Libraries

 JavaScript Resources

 build

 WebContent

 META-INF

 WEB-INF

 lib

 mysql-connector-java-5.0.2.jar

 servlet-api.jar

 web.xml

 Home.jsp

 Register.jsp



Register.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>Register</title>
<script>
function validate()
{
    var fullname = document.form.fullname.value;
    var email = document.form.email.value;
    var username = document.form.username.value;
    var password = document.form.password.value;
    var conpassword= document.form.conpassword.value;

    if (fullname==null || fullname=="")
    {
        alert("Full Name can't be blank");
        return false;
    }
    else if (email==null || email=="")
    {
        alert("Email can't be blank");
        return false;
    }
    else if (username==null || username=="")
    {
```

```

else if(password.length<6)
{
alert("Password must be at least 6 characters long.");
return false;
}
else if (password!=conpassword)
{
alert("Confirm Password should match with the Password");
return false;
}
}
</script>
</head>
<body>
<center><h2>Java Registration application using MVC and MySQL </h2></center>
<form name="form" action="RegisterServlet" method="post" onsubmit="return validate()">
<table align="center">
<tr>
<td>Full Name</td>
<td><input type="text" name="fullname" /></td>
</tr>
<tr>
<td>Email</td>
<td><input type="text" name="email" /></td>
</tr>
<tr>
<td>Username</td>
<td><input type="text" name="username" /></td>

```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import com.mvc.bean.RegisterBean;
import com.mvc.dao.RegisterDao;
```

```
public class RegisterServlet extends HttpServlet {
```

```
    public RegisterServlet() {
    }
```

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

```
        //Copying all the input parameters in to local variables
```

```
        String fullName = request.getParameter("fullname");
```

```
        String email = request.getParameter("email");
```

```
        String userName = request.getParameter("username");
```

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

```
        RegisterBean registerBean = new RegisterBean();
```

```
        //Using Java Beans - An easiest way to play with group of related data
```

```
        registerBean.setFullName(fullName);
```

```
        registerBean.setEmail(email);
```

```
        registerBean.setUserName(userName);
```

```
        registerBean.setPassword(password);
```

```
        RegisterDao registerDao = new RegisterDao();
```

```
        //The core Logic of the Registration application is present here. We are going to insert user data
in to the database.
```

```
        String userRegistered = registerDao.registerUser(registerBean);
```

RegisterDao.java

```
package com.mvc.dao;
```

```
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.SQLException;
```

```
import com.mvc.bean.RegisterBean;
```

```
import com.mvc.util.DBConnection;
```

```
public class RegisterDao {
```

```
    public String registerUser(RegisterBean registerBean)
```

```
    {
```

```
        String fullName = registerBean.getFullName();
```

```
        String email = registerBean.getEmail();
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

```
        // Insert user details into the database
```

RegisterBean.java

```
package com.mvc.bean;
```

```
public class RegisterBean {
```

```
    private String fullName;
```

```
    private String email;
```

```
    private String userName;
```

```
    private String password;
```

```
    public String getUserName() {
```

```
        return userName;
```

```
    }
```

```
    public void setUserName(String userName) {
```

```
        this.userName = userName;
```

```
    }
```

```
    public String getPassword() {
```

```
        return password;
```

```
    }
```

```
    public void setPassword(String password) {
```

```
        this.password = password;
```

```
    }
```

```
    public void setFullName(String fullName) {
```

```
        this.fullName = fullName;
```

```
    }
```

```
    public String getFullName() {
```

```
        return fullName;
```

```
    }
```

```
    public void setEmail(String email) {
```

DBConnection.java

```
package com.mvc.util;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBConnection {

    public static Connection createConnection()
    {
        Connection con = null;

        String url = "jdbc:mysql://localhost:3306/customers"; //MySQL URL followed by the database
        name
        String username = "root"; //MySQL username
        String password = "root123"; //MySQL password
        System.out.println("In DBConnection.java class ");

        try
        {
            try
            {
```

---

```
                Class.forName("com.mysql.jdbc.Driver"); //loading MySQL drivers. This differs for database
                servers
            }
            catch (ClassNotFoundException e)
            {
                e.printStackTrace();
            }

            con = DriverManager.getConnection(url, username, password); //attempting to connect to
            MySQL database
            System.out.println("Printing connection object "+con);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        return con;
    }
}
```

Web.xml

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">

    <display-name>RegistrationMVC</display-name>

    <welcome-file-list>

    <welcome-file>Register.jsp</welcome-file>

</welcome-file-list>
```



## Web.xml

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">

<display-name>RegistrationMVC</display-name>

<welcome-file-list>

<welcome-file>Register.jsp</welcome-file>

</welcome-file-list>

<servlet>
```

```
<description></description>

<display-name>RegisterServlet</display-name>

<servlet-name>RegisterServlet</servlet-name>

<servlet-class>com.mvc.controller.RegisterServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>RegisterServlet</servlet-name>

<url-pattern>/RegisterServlet</url-pattern>

</servlet-mapping>

</web-app>
```

## Home.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>

<title>Home Page</title>

</head>

<body>

<center><h2>Home Page</h2></center>

<b>User Registration Successful</b>

<br></br>

<b>Please <a href="https://krazytech.com/programs/a-login-application-in-java-using-model-view-
controllermvc-design-pattern">log-in</a> to continue.</b>
```

</body>

</html>

Full Name

srija chitikesi

Email

srijachitikesi16@gmail.com

Username

srija

Password

••••

Confirm Password

••••|

Register

Reset

# Home Page

**User Registration Successful**

**Please [log-in](#) to continue.**

## 2. Differentiate MVC and MVT Design Architectures

### Sol: 1. Model View Controller (MVC):

- It is a software design pattern that is used to implement user interfaces and gives emphasis on separating data representation from the components which interact and process the data.
- 3 components has a specific purpose:
  - Model - is a central component of this architecture and manages the data, logic as well as other constraints of application
  - View - deals with how the data will be displayed to the user and provides various data representation components
  - Controller - manipulates the Model and renders the View by acting as a bridge between both of them

### 2. Model View Template (MVT):

This is yet another design pattern similar to MVC. It is also used for implementing web interfaces and applications but in contrast to MVC, the controller part is taken care for us by the framework itself.

- 3 components has a specific purpose
  - Model similar to MVC acts as an interface for your data and is basically the logical structure behind the

entire web application which is represented by a database such as MySQL, PostgreSQL.

- The View executes the business logic and interacts with the Model and renders the template. It accepts HTTP request and then return HTTP Responses.
- The Template is the Component which makes MVT different from MVC. Templates act as the presentation layer and, are basically the HTML code that renders the data. The content in these files can be either static or dynamic.

#### Differences:

S.NO	MVC	MVT
1.	MVC has controller that drives both Model and View	MVT has Views for receiving HTTP request and returning HTTP Response.
2.	View tells how the user data will be presented	Templates are used in MVT for that purpose
3.	In MVC, we have to write all the control specific code	controller part is managed by the framework itself
4.	Highly coupled	Loosely coupled
5.	Modifications are difficult	Modifications are easy
6.	Flow is clearly defined	Flow is sometimes hard to understand
7.	Doesn't involve mapping of URLs	URL mapping takes place

3. Explain in detail about CSRF protection in Django with Suitable Diagram.

Sol.

`{% csrf_token %}` is a special tag reserved for cases when a web form is submitted via POST and processed by Django. The CSRF initials mean Cross-Site Request Forgery, which is a default security mechanism enforced by Django.

- While it's possible to disable CSRF and not include the `{% csrf_token %}` Django tag in forms, it would advise against it and recommend you keep adding the `{% csrf_token %}` Django tag to all forms with POST, as CSRF works as a safeguard and mostly behind the scenes.



## Diagram!

