

A Study Focused on Web Application Development using MVC Design Pattern

Ram Naresh Thakur¹, Dr. U.S. Pandey²

¹Research Scholar, Mewar University, Rajasthan, India

²Professor, University of Delhi

Abstract - The Model-View-Controller (MVC) is very useful for developing Interactive and Dynamic Web Applications. It has become the most powerful and dominant Programming Paradigm for developing large scale and Dynamic Web Applications. With MVC, developers can trust on design patterns that are widely accepted as solutions for recurring problems and used to develop flexible, reusable and modular Software. Applying the MVC design pattern to Web Applications is therefore complicated by the fact that current technologies encourage developers to partition the application as early as in the design phase. In this work we have developed a web-based application using MVC framework using Java Web Architecture.

Key Words: MVC, Dynamic, Web Application, Programming Paradigm, Developers, Design Phase, Java Web Architecture.

1. INTRODUCTION

1.1 Web Application

A Web Application is a distributed application that runs on more than one Computer and communicates through a Network or a Server. Specifically, a Web Application is accessed with a Web Browser as a client and provides the facility to update and manage a program without deploying and installing Application on client Computers. Web Applications are used for Email, Online Retail Purchase/Sales, Online Banking, among others. The advantage of Web Application is that can be access and used by millions of people at the same time [1].

1.2 MVC Architecture

MVC is a software design pattern built around the interconnection of three main component types: Model, View, and Controller, often with a strong focus on object-oriented programming (OOP) software paradigms. MVC is a framework for building web applications using an MVC Design. It is the most important architecture of development the software now a day. This architecture automatically managed the code and help the programmer to develop the well-managed Web Applications.

2. LITERATURE REVIEW

The MVC software application architecture has been widely embraced as an approach for developing Web-based systems that contain a server-side programming component, particularly for those requiring database access. MVC isolates the business logic from the user interface, with the goal of creating applications that are easier to manage and maintain because designers can modify the visual appearance of the application and programmers can modify the underlying business rules with fewer harmful side effects. The bulk of the published literature on MVC Web applications describes the architecture and underlying specification of these systems, but does not specifically address the effectiveness of the architecture. The purpose of this research is to compare MVC to other widely-used Web development methods in terms of development time, maintainability, and ability to support and enhance interaction among designers and programmers [2].

MVC Design Patter is the Architecture that presents several views for the same data. It separates the Data Layer and the Expression Layer, and divides the Application objects into three classes: Model class, View class and Control class. Model class implements the Business and Data Logic; View class implements the Display Logic, and Control class implements the Control Processing. MVC holds the three logic types independency to build the Business and Data Logic oriented business and design the control and display logic-oriented application. The changing of business processing does not modify the business and data logic. The changing of business principles and algorithms only modifies Model class. So MVC separates the data accessing and display and ensure the modules independency [3].

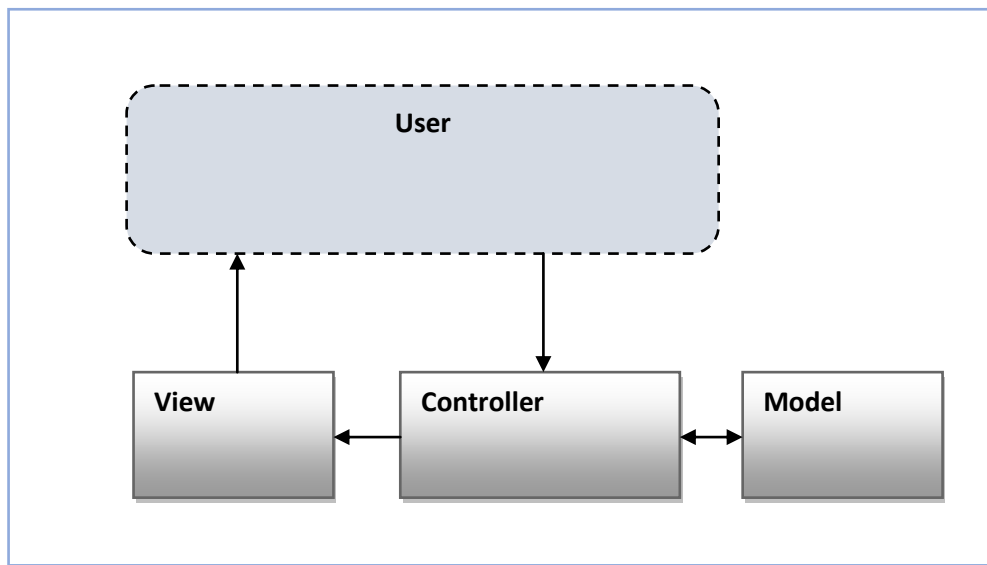
The MVC architecture pattern divides an application layer into three components. The model contains the functions and core data. View display information to the user. The controller handles user input and information. View and Controller both composed the user interface. MVC follows the most common approach to Layering. Layering is a service that separates our code into functions into different classes. This approach is easily recognized and the most widely accepted. The main advantage of this approach is the reusability of the code [4].

MVC design pattern helps us to implement the separation of application among the Model, View and Controller classes within applications. Separation of application makes it easy for us to test our application as relation among different components of application. MVC help us to implement a test-driven application development approach, in which we implement automated test phase before we write the code. These unit test phases help us pre-define and verify requirements of new code before writing it [5].

MVC design pattern to develop and implement a dynamic E-business system or an Internet application. The enterprise level application is developing quickly, and the E-commerce market is growing fast, more and more enterprise level projects presents the development trend that they take the Web technology as the central technology. At the same time, the dependence to the server end technology like the middleware is also increasing. So, the Information Technology department of the Enterprise needs a feasible way to develop the application, and make the application be related with that middleware which are flexible and can be transplanted [6].

3. PROPOSED FRAWORK

MVC Architecture is a modern Design Pattern of developing Application. It is separate the application into three main Layers: Model, View and Controller. The Model contain the Business Logic Layer (BLL) that process the application data and also stores or retrieves data to or from the database. The View (Presentation Layer) displays information to the user and the Controller (Controlling Logic Layer) handles user interactions and input.



3.1 MVC Architecture using JAVA

Developing Web Application in Java, The Models where developed using Enterprise Java Beans (EJB) and the Java API(JPA), the View where developed using Java Server Page (JSP), The Controllers where developed using Java Servlets(.java). The given figure shows the MVC design pattern using Java.

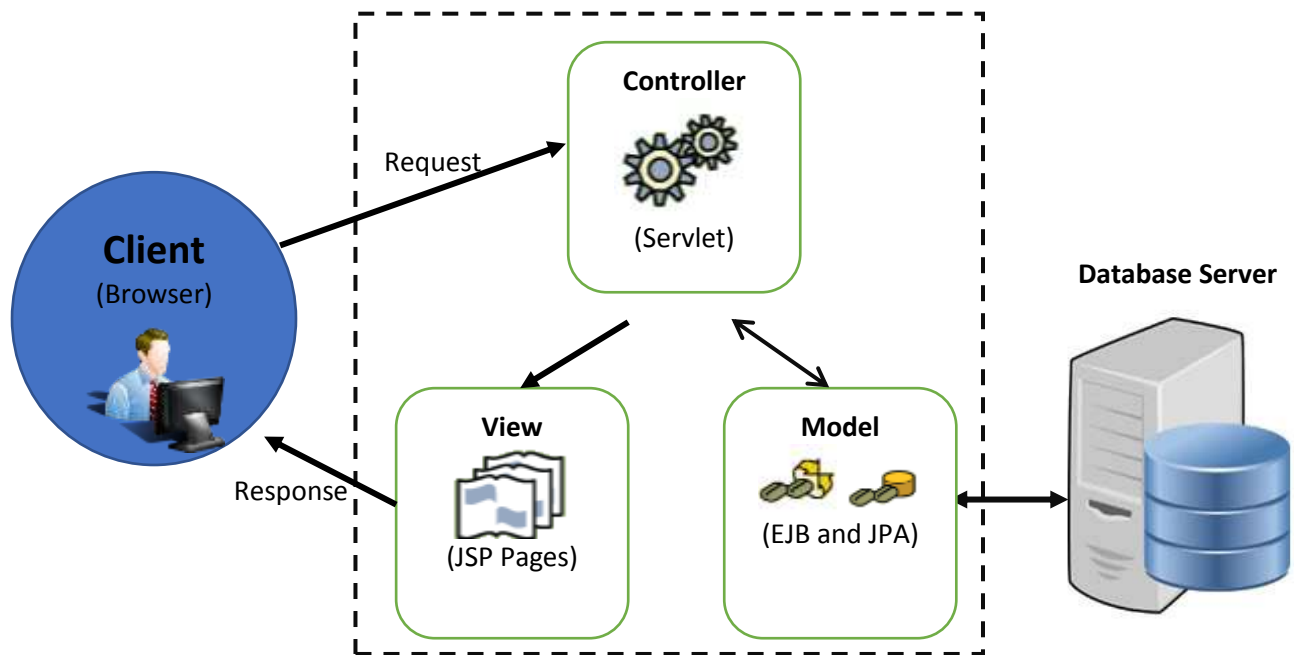


Fig -2: MVC Architecture using JAVA

4. IMPLIMENTATION

In this architecture, user HTTP requests are routed through a Controller, which is implemented as a Servlet (StudentServlet.java). The controller is responsible for processing the request, instantiating Model class (Student.java) if needed, and selecting the response to be returned to the browser. If using JSP page (StudentPage.jsp) for the view, the JSP page is able to retrieve data stored in the model's JavaBeans using the JSP Expression Language.

```

Student.java
public class Student {
    private int Sid;
    private String Name;
    private String Address;
    private String Course;

    public int getSid() {
        return Sid;
    }
    public void setSid(int Sid) {
        this.Sid = Sid;
    }
    public String getName() {
        return Name;
    }
    public void setName(String Name) {
        this.Name = Name;
    }
    public String getAddress() {
        return Address;
    }
    public void setAddress(String Address) {
        this.Address = Address;
    }
    public String getCourse() {
        return Course;
    }
    public void setCourse(String Course) {
        this.Course = Course;
    }
}

StudentController.java
import com.osp.dao.StudentDao;
import com.osp.model.Student;
import java.io.IOException;
import java.io.PrintWriter;
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(name = "StudentController", urlPatterns = {"/StudentController"})
public class StudentController extends HttpServlet {

```

Fig -3: Snapshot 1

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doPost(request, response);
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    try {
        Student s = new Student();
        StudentDao sd = new StudentDao();
        if (request.getParameter("Sid") != null) {
            s.setSid(Integer.parseInt(request.getParameter("Sid")));
        }
        s.setName(request.getParameter("Name"));
        s.setAddress(request.getParameter("Address"));
        s.setCourse(request.getParameter("Course"));
        if (request.getParameter("add") != null) {
            sd.saveStudent(s);
            request.setAttribute("sList", sd.getStudentList());
        }
        RequestDispatcher rd = request.getRequestDispatcher("ListStudent.jsp");
        rd.forward(request, response);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
StudentDao.java

package com.osp.dao;
import com.osp.model.Student;
import com.osp.utils.DbConnection;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class StudentDao {
    Connection cn;
    public StudentDao() {
        cn=DbConnection.getConnection();
    }
    public void saveStudent(Student s){
        try {
            String sql="INSERT INTO STUDENT (NAME, ADDRESS, COURSE, PHOTO) VALUES(?, ?, ?, ?)";
            PreparedStatement ps=cn.prepareStatement(sql);
            ps.setString(1, s.getName());
```

Fig -4: Snapshot 2

```

        ps.setString(2, s.getAddress());
        ps.setString(3, s.getCourse());
        ps.execute();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
}
}
AddStudent.jsp

<%@page import="com.osp.model.Student"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
        <link href="site.css" rel="stylesheet" type="text/css"/>
    </head>
    <body>
        <h1>Add Student</h1>
        <%
            Student s=(Student)request.getAttribute("s");
            if(s==null){
                s=new Student();
                s.setSid(0);
                s.setName("");
                s.setAddress("");
                s.setCourse("");
            }
        %>
        <form action="StudentController" method="post">
            <div class="imgcontainer">
                <h1>ONLINE STUDENT PORTAL</h1>
            </div>
            <div class="container">
                <input type="hidden" value="<%=s.getSid()%>" name="Sid">
                <label for="name"><b>Name</b></label>
                <input type="text" value="<%=s.getName()%>" placeholder="Enter Name" name="Name"
                    required>
                <label for="addr"><b>Address</b></label>
                <input type="text" value="<%=s.getAddress()%>" placeholder="Enter Address"
                    name="Address" required>
                <label for="cours"><b>Course</b></label>
                <input type="text" value="<%=s.getCourse()%>" placeholder="Enter Course"
                    name="Course" required>
            </div>
        </form>
    </body>
</html>

```

Fig -5: Snapshot 3

```

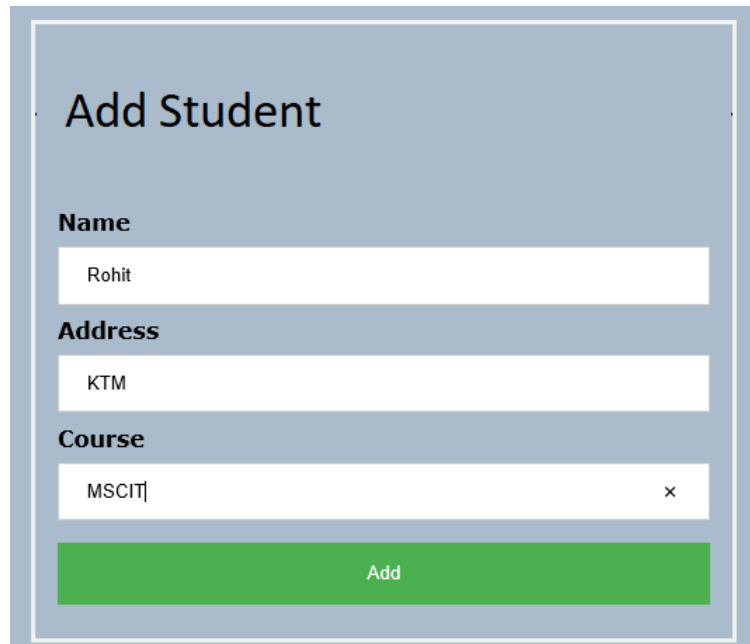
        <button type="submit"
name="<%=s.getSid()==0?"add":"update"%><%=s.getSid()==0?"Add":"Update"%></button>
    </div>
    <div>
        <% if (session.getAttribute("myMessage") != null) {%>
        <%=session.getAttribute("myMessage")%>
        <%}%>
    </div>
</form>
</body>
</html>
StudentPage.jsp
<%@page import="com.osp.dao.StudentDao"%>
<%@page import="java.util.List"%>
<%@page import="java.util.ArrayList"%>
<%@page import="com.osp.model.Student"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
    <link href="site.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <h1>Student Details</h1>
    <%
    StudentDao sd=new StudentDao();
    List<Student> sList=(ArrayList<Student>)sd.getStudentList();
    //List<Student> sList = (ArrayList<Student>) request.getAttribute("sList");
    %>
    <table>
        <tr><td><a href="AddStudent.jsp">Add Studebnt</a></td></tr>
        <tr>
            <th>Student ID</th>
            <th>Name</th>
            <th>Address</th>
            <th>Course</th>
            <th>Edit</th>
            <th>Delete</th>
        </tr>
        <% for (Student s : sList) {%>
        <tr>
            <th><%=s.getSid()%></th>
            <th><%=s.getName()%></th>
            <th><%=s.getAddress()%></th>
            <th><%=s.getCourse()%></th>
            <th><a href="StudentController?SidForEdit=<%=s.getSid()%>">Edit</a></th>
            <th><a href="StudentController?SidForDelete=<%=s.getSid()%>">Delete</a></th>
        </tr>
        <%}%>
    </table>
</body>
</html>

```

Fig -6: Snapshot 4

4.1 User Interface of the above application

CRUD Operation Using Java



Add Student

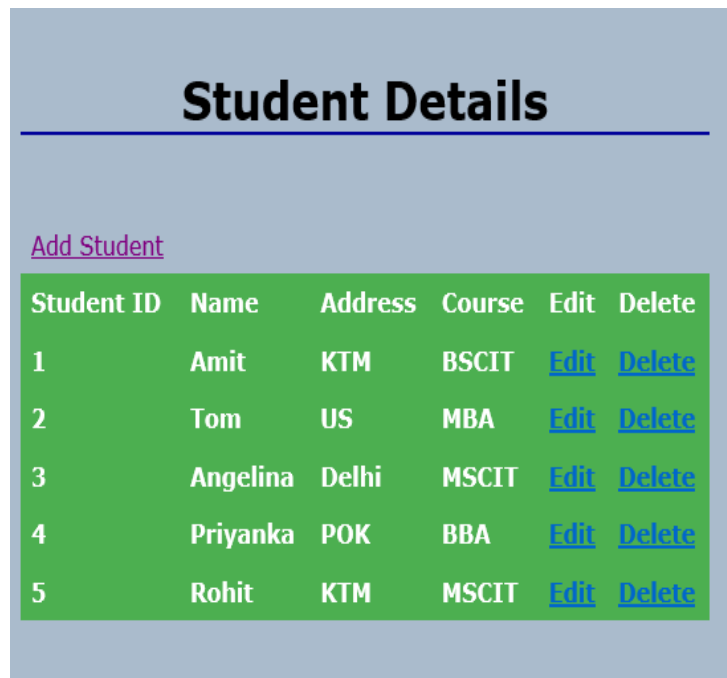
Name
Rohit

Address
KTM

Course
MSCIT

Add

Fig -7: Adding Information



Student Details

[Add Student](#)

Student ID	Name	Address	Course	Edit	Delete
1	Amit	KTM	BSCIT	Edit	Delete
2	Tom	US	MBA	Edit	Delete
3	Angelina	Delhi	MSCIT	Edit	Delete
4	Priyanka	POK	BBA	Edit	Delete
5	Rohit	KTM	MSCIT	Edit	Delete

Fig -8: View/ Edit/ Delete Information

5. CONCLUSION

MVC is a software design pattern built around the interconnection of three main component types: Model, View, and Controller, often with a strong focus on object-oriented programming (OOP) software paradigms. MVC is a framework for building web applications using an MVC Design. The MVC is very useful for developing Interactive and Dynamic Web Applications. It has become the most powerful and dominant Programming Paradigm for developing large scale and Dynamic Web Applications. Here we have developed a web-based application using MVC framework using Java Web Architecture.

REFERENCES

- [1] J. G. D. David A. Botwe, "A Comparative Study of Web Development Technologies Using Open Source and Proprietary Software," *International Journal of Computer Science and Mobile Computing*, vol. 4, no. 2, pp. 154-165, 2015.
- [2] J. M. a. M. M. Nick Heidke, "Assessing the Effectiveness of the Model View Controller," *Architecture for Creating Web Applications*.
- [3] L. Hao, "Application of MVC Platform in Bank E-CRM," *International Journal of u- and e- Service, Science and Technology*, vol. 6, no. 2, pp. 33-42, 2013.
- [4] A. W. A. S. Rinci Kembang Hapsari, "Architecture Application Model View Controller (Mvc) in Designing Information System of Msme Financial Report," *Quest Journals Journal of Software Engineering and Simulation*, vol. 3, no. 7, p. 37, 2017.
- [5] A. M. a. L. Rauf, "MVC Architecture: A Detailed Insight to the Modern Web Applications Development," *Crimson Publishers*, vol. 1, no. 1, pp. 1-7, 2018.
- [6] M. Bhatt, "J2EE and MVC Architecture," *Journal of Global Research Computer Science & Technology*, vol. 1, no. 2, pp. 25-29, 2014.
- [7] C. Pitt, *Pro PHP MVC*, New York: Apress, 2012.

BIOGRAPHIES



Mr. Ram Naresh Thakur is an IT professional & currently pursuing his PhD from Mewar University, Rajasthan, India. His research interest includes software engineering.



Dr. U. S. Pandey is currently working as a Professor in University of Delhi, India. He has authored several books and has good number of paper publications in national & international journals.