Comparative Study of MVC (Model View Controller) Architecture with respect to Struts Framework and PHP

Dr. Sindhu Singh
Assistant Professor, Department of Information Technology, K.J Somaiya Institute of Management Studies & Research, Mumbai-22, Maharashtra(India), sindhusingh@ somaiya.edu
Dr. Jaya Iyer
Assistant Professor, Department of Information Technology, K.J Somaiya Institute of Management Studies & Research, Mumbai-22, Maharashtra(India),jayaiyer_rk@ somaiya.edu

Abstract— Model View Controller (MVC) architecture is implemented over the middleware. MVC is mainly used for developing web applications, because it helps the developer to have a clear understanding of all the modules. MVC architecture separates the business layer (Model logic), the display layer (View logic), and the control layer (Controller logic). This study attempted to understand the implementation difference of MVC architecture using the Struts framework and PHP. MVC architecture is basically used to implement complex real time applications. It is easier to implement MVC architecture in Struts since it supports various IDE (Integrated Development Environment) which has complete support of frameworks. In PHP, the developer needs to create manually frameworks for model, view, and controller.

Keywords-MVC, Struts,PHP,web applications

I. INTRODUCTION

Model View Controller is popularly known as MVC architecture which is made up of three parts: Model, View and Controller. MVC was designed by Trygve Reenskaug in 1979, to provide better solution for large and complex problem. For the First time it was used in the Smalltalk-80 framework – Used in making Apple interfaces (Lisa and Macintosh) (Reenskaug, 1979). MVC is mainly used for developing web applications, because it helps the developer to have a clear understanding of all the modules. MVC architecture separates the business layer (Model logic), the display layer (View logic), and the control layer (Controller logic). MVC helps to organize the data and separate the code as per requirement, for reducing complexity and working on one file as business logic. Code reusability and flexibility can be achieved by implementing the MVC architecture. Using MVC, the developer has a clear understanding of all the modules.

II. MVC ARCHITECTURE

A) Model

The Model is a class having variables with different data types along with getter and setter properties. There is no database connectivity with it, only data management is carried out. The main activity of model class is to give a proper response from the view's class request and instruction from the controller class. The example 1.1 shows that how to define a model class with getter and setter properties for data member in the class. In this example *photo* is the model class name followed by the *description* which is a method name which provide the get and set for the data variables in photo class.

```
Example 1.1:  public class photo    //model class
 {
public string description {get; set ;}
}
```

B) View

The main responsibility of the view class to provide the graphical user interface output for model component that is specifically designed for client requirements. The View contains the data (the database records). The response generated by a model needs to generate an output representation. It presents data in a particular format like JSP, ASP, and PHP.

C) Controller

When a request is released, the controller responds to the user. Controllers can read data from a view, control user input, send the input data to the model, invokes changes in the model and view. The controller class has all the controls lined with view and model classes.

The following figure 1 depicts the relationship between model, view and controller class. The control flow for the architecture is as follows [1][2].

The user provides a request through the user interface in View class.

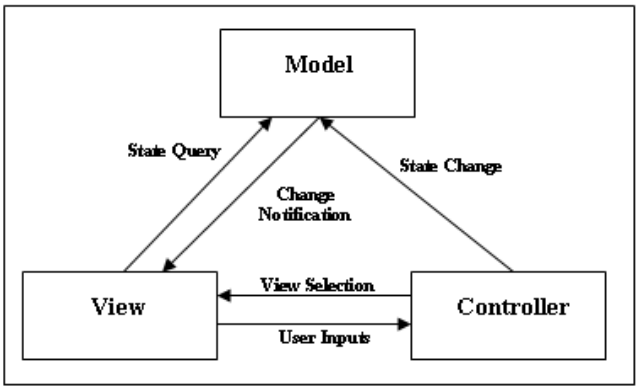
The user inputs are passed to Controller class.

The controller class transfers the information to the model class to take appropriate action for the input provided by the user.

The controller class has the responsibility to fire the proper event based on the request.

Finally the state change updated with the model class by controller class will be notified in the view class.

Figure 1. MVC Architecture Components[1]



III. MVC WITH JAVA (STRUTS 2)

MVC architecture will be clearly explained by different server side programming concepts. A strut is an application framework for building websites in Java using the Java Enterprise Edition platform. It was initially created by Craig McClanahan and donated to the Apache Foundation in May, 2000 and Struts 1.0 was released in June 2001.The current stable release of Struts is Struts 2.3.16.1 in March 2, 2014. A strut is an open source license which is freely available [3].

A strut 1 was a popular version of Java, which provides the request and response handler and tag library. When comparing to Struts 1, Strut 2 is a popular web application framework in Java which uses an MVC architecture pattern. In order to make web development easier Java started using a Struts application with MVC architecture. The major core components for strut 2 are action handler, result handler and custom tags. The action handler interacts with other layers, whereas result handler helps to display the response to the user, custom tags helps supply the dynamic content to the application based on the request. The Struts 2 also supports POJO based actions, validation, AJAX, integration with various frameworks such as Hibernate, Spring, and Tiles, etc. It also supports various result types such as Freemarker, Velocity, and JSP etc.

The following is the steps to create a web application using Struts 2 can be created using NetBeans, Eclipse, etc.

Example program for implementing MVC architecture using Struts.

Steps to create

1. Create Web Application
2. JSP is used to design dynamic web pages in struts. The Servlet code helps to direct the request to the appropriate server page based on the web browsers request. It helps to design the web application and easy to maintain.
3. Create index.jsp followed by input page login.jsp.
4. Create a Web.xml file as an entry point (deployment descriptor)
5. Create an action class, i.e. bean class with .java extension (LoginActionform.java, LoginAction.java).
6. A mapping file as Struts-config.xml which gets data and result invocation from action class
7. Create a view file which displays processed data as sucess.jsp or failure.jsp.
8. Start deploying the project

A web application will display a login page with username and password fields. A login button will be available. On click on the login button, username and password will be validated. Upon success, a success page

will be displayed. If the login fails, an error message will be displayed on the login page. Create a login Web Application using Struts.

IV. PROGRAM CODE

1) login.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html lang="true"> <head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title><bean:message key="welcome.title"/></title>
<html:base/> </head>
<body style="background-color: white">
<html:form action="/login">
<html:errors property="username" />
<html:errors property="password" />
<table border="1" align="center">
<thead><tr>
<th colspan="2">Login to your site</th>
</tr> </thead>
<tbody>
<tr>
<td>Username:</td>
<td><html:text property="username" /></td>
</tr>
<tr>
<td>Password:</td>
<td><html:password property="password" /></td>
</tr>
<tr><td align="center" colspan="2"><html:submit value="Login" />&amp;nbsp;&amp;nbsp;&amp;nbsp; <html:reset value="Reset" /></td>
</tr>
</tbody>
</table>
</html:form>
</body>
</html:html>
```

2) index.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:forward page="login.do"/>
```

3) LoginActionForm.java

```
package com.myapp.struts;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
```

```
public class LofinActionForm extends org.apache.struts.action.ActionForm
{
  private String username;
  private String password;
//Getter Method
  public String getUsername()
 {          return username;     }
//Setter Method
public void setUsername(String username)
{    this.username = username;    }
public String getPassword()
{        return password;       }
public void setPassword(String password)
{         this.password = password;      }
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request)
{
    ActionErrors errors = new ActionErrors();
    if (getUsername()== null || getUsername().length() < 6) {
        errors.add("username", new ActionMessage("errors.username.required"));
   }
  if (getPassword()== null || getPassword().length() < 5) {
        errors.add("password", new ActionMessage("errors.password.required"));
     }
   return errors;
   }
}
```

4) LoginAction.java

```
package com.myapp.struts;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
public class LoginAction extends org.apache.struts.action.Action
{
   /* forward name="success" path="" */
private static final String SUCCESS = "success";
private static final String FAILURE = "failure";
@Override
public ActionForward execute(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response) throws Exception
    {
     LoginActionForm data = (LoginActionForm)form;
    String email = data.getUsername();
    String pass = data.getPassword();
    if ((email.equals("simsr")) && (pass.equals("admin")))
    {
       return mapping.findForward(SUCCESS);
```

```
    }
    else
    {
       return mapping.findForward(FAILURE);
    }    } }
```

5) Struts-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts-config PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
        "http://jakarta.apache.org/struts/dtds/struts-config_1_3.dtd">
<struts-config>
<form-beans>
<form-bean name="LoginActionForm" type="com.myapp.struts.LoginActionForm"/>
</form-beans>
<global-exceptions>
</global-exceptions>
<global-forwards>
<forward name="login"  path="/login.do"/>
</global-forwards>
<action-mappings>
<action input="/login.jsp" name="LoginActionForm" validate="true" path="/login" scope="request"
type="com.myapp.struts.LoginAction">
<forward name="success" path="/Success.jsp"/>
<forward name="failure" path="/Failure.jsp"/>
</action>
</action-mappings>
<controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
<message-resources parameter="com/myapp/struts/ApplicationResource"/>
<plug-in className="org.apache.struts.tiles.TilesPlugin" >
<set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />
<set-property property="moduleAware" value="true" />
</plug-in>
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
<set-property  property="pathnames"
          value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
</struts-config>
```

6) success.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login Successfully</title>
</head>
<body>
<center><h1>Login Successfully Done</h1></center>
</body>
```

```
</html>
```

7) failure.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login Failure</title>
</head>
<body>
<center><h1>Invalid Credentials</h1></center>
</body>
</html>
```

Struts 2 framework is simple to use and easy to understand framework. Developers can add plug-ins easily, like Hibernate, Spring, Tiles, SiteMesh etc. It includes theme based tags and Ajax tags. It introduced annotations which reduces the length and complexity of code. In this framework, templates can be added and can be easily modified. Multiple View options are supported that is, a view can be JSP, Freemarker and velocity templates. For POJO i.e. ActionForms, implementation of any interface or extend from any class is not needed. Compatibility is one major limitation in the Struct 2 framework. Struts 2 and struts1 is completely different, so migration is very difficult. Another limitation is that, Limited documentation is available for Struts 2; new users may face difficulties to understand.

V. MVC WITH PHP

PHP is server side scripting language compatible in all operating systems. Developers are moving towards PHP with MVC for better performance, extensibility and easier coding. PHP started MVC framework in 2008 and current version is 3.1 which are used Radip Application Development(RAD) model [4].PHP MVC Framework is designed to be lightweight and modular, allowing developers to build better and easier code which helps ease in maintenance. The base framework comes with a few helper classes, this is to keep code bloat down to a minimum. Classes can be easily added at any stage of development.

Web development in PHP has a powerful architecture for PHP frameworks as there are 17 frameworks like Zend, CodeIgniter, CakePHP, and Smarty –MVC which supports all databases like SQL, My SQL, Oracle and ODBC. These frameworks provide a great combination of database application (model), HTML coding (view) and input/ output instructions (controller) [5].

A web application will display a login page with username and password fields. A submit button will be available. On click on the submit button, username and password will be validated. Upon success, a success page will be displayed. If the login fails, an error message will be displayed on the login page. Create a login Web Application using Struts.

VI. STEP TO CREATE AN MVC PROGRAM IN PHP

1. Create three folders with the name as Model, Controller and View. Create an index.php landing page to start executing the MVC program.

2. View folder consists of the normal view page like login.php, success.php and failure.php.

3. The Controller will transfer the control based on the user input given in the login page. If value is true the controller will transfer the request to success page otherwise the failure page.

4. In Model components consists of the actual validation code will for validate the user input.

5. Based on the PHP server appropriately create the project and store all the three folders in the project for execution.

1) Index.php

```
<?php
 include_once ("controller/Controller.php");
 $controller = new Controller ();
 $controller->invoke ();
?>
```

A) View Components

2) Login.php

```
<html>
<head><title>Login Page</title></head>
<body>
<table align="center" border = 2>
<th align="center" colspan=2> Login Page </th>
 <tr>
 <form action="" method="POST">
 <p>
  <td><label>Username</label></td>
  <td><input id="username" value="" name="username" type="text" required="required"/>
  <br></td>
  </p></tr>
  <p><tr align="center"> <td><label>Password</label></td>
  <td><input id="password" name="password" type="password" required="required" /></td>
  </p><br/>
  <p></tr>
  <tr><th align="center" colspan=2>
  <input type="submit" name="Login">
  <input type="reset" name="Cancel">
  </th></form></tr></table></body>
</html>
```

3) Success.php

```
<html>
<head></head>
<body>
<?php
echo 'Login success';
$result=' ';
?>
</body>
</html>
```

4) Failure.php

```
<html>
<head></head>
<body>
<?php
echo 'Invalid User try again to input proper value ' ;
$result=' ';
?>
</body>
</html>
```

B) Model Component

5) Model.php

```
<?php
class Model {
 public function getlogin()
 {
```

```
 if(isset($_REQUEST['username']) && isset($_REQUEST['password'])){
   if($_REQUEST['username']=='simsr' && $_REQUEST['password']=='admin')
  {
   return 'login';
  }
  else
  {
   return 'invalid user';
  }  }  }  }
?>
```

C) Controller Component

6) Controller.php

```
<?php
include_once("model/Model.php");
class Controller {
 public $model;
 public function __construct()
  {
      $this->model = new Model();
  }
 public function invoke()
 {
  $result =' ';
  $result = $this->model->getlogin();
  if($result == 'login')
  {
   include 'view/success.php';
  }
  elseif($result == 'invalid user')
  {
   include 'view/failure.php';
   }
  else
   {
  include 'view/login.php';
    }  }  }
?>
```

Here new file .htaccess file is used which redirects request to index.PHP, so it should be in the same folder with index.PHP.

VII. COMPARISON BETWEEN FRAMEWORK AND CORE PHP

For simple projects using PHP core is fine, but as the logic increases and so the complexity of projects it creates a big mess to manage. Inconsistency increases as the project could have been divided as per different models. Basic knowledge of PHP can be learnt from the core PHP only. It is not possible to jump directly to the top level.

The framework is used to develop PHP projects which save time and minimize the debugging effort as well as provide safe and clean projects. It also helps to increase the reliability, reusability and consistency of the projects through its lifetime. The main purpose of the framework is the separation of the logic by using three different components like model, view and controller. This MVC framework is well suited for complex application which provides better control for the maintenance. The MVC PHP can be used if the project scope is large requires platform independence provided having good knowledge of core PHP [6].

VIII. COMPARISON BETWEEN MVC IN STRUT, AND PHP

The MVC framework can be implemented in both strut as well as PHP. The table-1 depicts the detail comparison between PHP and strut.

Table 1- MVC architecture comparison between Strut and PHP

Terms	PHP	Strut
Implementation	PHP is a server side scripting language.MVC can be implemented by using the Zend framework.	MVC is easy to implement
Model Component	Using PHP class.	Getter and Setter method with Java Bean Class
View Component	HTML, PHP pages	JSP pages.
Controller Component	PHP page redirect based on the request parameter value.	Java Servlet
Security	Less secure.	More secure.
Unicode	Does not support Unicode.	Full Unicode support.
MVC Framework	Partially.	Fully Supported.

IX. CONCLUSION

The overview of MVC architecture with the two languages, i.e. Struts and PHP, is discussed in this paper. All of these languages are useful and makes a drastic difference while using an MVC framework instead of Core version of it. The MVC is the best option but choosing language is completely depends on the expertise of that language or the organization which develop the project. MVC architecture is basically used to implement complex real time applications. It is easy to maintain and update without affecting the other components. MVC implementation helps to modularize the complex projects.

REFERENCES

[1] Kumar, S. (2013). JDBC, Servlets, and JSP (Includes JSF and Design Patterns) Black BookDreamtech Press/Wiley India.
[2] Bhatt, M. (2014). J2EE and MVC Architecture. Journal of Global Research Computer Science(JGRCST), 1(2), 2349 – 5170.
[3] D'mello, N., & Carvalho, L. (2013). Struts 2- The modern web application framework. International Journal of Modern Engineering Research (IJMER), 3(3), 1854–1756.
[4] Holzner, S. (2007). PHP: the complete reference. Tata McGraw-Hill Education.
[5] Paikens, A., & Arnicans, G. (2008). Use of design patterns in PHP-based web application frameworks. Scientific Papers University of Latvia, Computer Science and Information Technologies, 733, 53-71.
[6] Supaartagorn, C. (2011). PHP framework for database management based on MVC pattern. International Journal of Computer Science and Information Technology, 3(2), 251–258.