Spring MVC Introduction

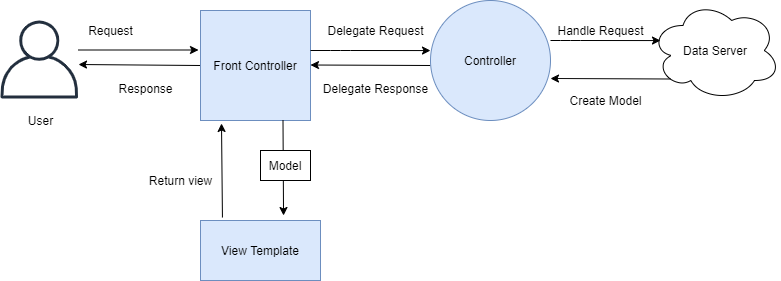
Spring MVC is a spring module that is used to create web applications using the Spring framework. Spring provides a source module spring-webmvc to build MVC based web applications.

What is MVC?

The MVC is a design pattern used to create applications. The MVC stands for Model, View, and Controller. It consists of three components Model, View, and Controller and in the below architecture, we can see how MVC components interact and work together. We will see each component of MVC in detail later in our tutorial but for now, we can understand it as the Controller is a request handler that handles the user request and responds requested view page back. The View is a page that can be an HTML or JSP page and rendered as a result while the Model represents the database table and used for data handling between application and database.

Spring MVC uses Servlet Dispatcher as a front controller that primarily works as a request handler and handles all user-defined controllers as well.

Spring MVC Architecture



**User:** a user is a client in a client-server architecture that requests a resource.

[**View:**](https://www.studytonight.com/tutorials/preview?subject=spring-framework&url=spring-mvc-view) a view is a component in MVC architecture that represents an HTML/JSP file and generated randomly by the controller. For example,

**// hello.jsp**

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Hello JSP</title>

</head>

<body>

<h1>Hello, Welcome to JSP</h1>

</body>

</html>

**Front Controller:** it is a built-in controller of the Spring framework that is used to handle all requests for the web application. Spring's DispatcherServlet works as a front controller.

**Model:** It is a component and part of MVC architecture that represents a class. The model is used to write business logic and getter, setters methods. For example,

**// User.java**

package com.studytonight.models;

public class User {

int id;

String name;

String lastName;

String email;

public User(int id, String name, String lastName, String email) {

//super();

this.id = id;

this.name = name;

this.lastName = lastName;

this.email = email;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

**Controller:** It is used to handle client requests in web applications. It receives a request from the user and sends the requested view after coordinating with models and the database. In the Spring application, a class that is marked with @Controller or @RestController annotation is treated as controller and con be used to handle requests and responses. For example,

package com.studytonight.controllers;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ResponseBody;

import com.studytonight.models.User;

import com.studytonight.service.UserService;

@Controller

public class UserController {

@Autowired

private UserService userService;

@GetMapping("/users")

@ResponseBody

public void users() {

List<User> users = userService.getUser();

for(User user: users) {

System.out.println(user.getFirstName()+" "+user.getLastName()+" "+user.getEmail());

}

}

}

Well, til here, hope you have got the idea of MVC and how MVC works in web application. In our next, topic, we will cover MVC components and working examples.