Spring MVC (Model-View-Controller) :

It is a popular framework for building web applications in Java. It follows the MVC architectural pattern, which separates the application into three main components: the Model, the View, and the Controller. Here's a detailed explanation of Spring MVC with two code examples.

1. Setting Up a Spring MVC Project

Before we dive into code examples, you'll need to set up a Spring MVC project. You can use a build tool like Maven or Gradle to manage dependencies and create your project's structure. Ensure you have the necessary Spring dependencies in your project's configuration. For this example, we'll use Maven.

1. **Create a Maven project**: You can create a Maven project using an IDE or the command line.

2. **Add Spring dependencies to your `pom.xml`**:

```xml
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>5.3.10</version> <!-- Use the appropriate version -->
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.3.10</version> <!-- Use the appropriate version -->
</dependency>
```

3. **Create a `web.xml` file** in the `WEB-INF` directory of your project:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!-- Configure the Spring DispatcherServlet -->
    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
```

```
    </servlet>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

4. **Create a Spring configuration file (e.g., `dispatcher-servlet.xml`) in the `WEB-INF` directory to configure the Spring MVC application.

2. Creating a Simple Spring MVC Controller

Let's create a simple Spring MVC controller that handles HTTP requests. The controller will respond to a URL and return a "Hello, World!" message.

Controller Class: `HelloController.java`

```java
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class HelloController {

    @RequestMapping(value = "/hello", method = RequestMethod.GET)
    @ResponseBody
    public String sayHello() {
        return "Hello, World!";
    }
}
```

In this example, we define a controller class using the `@Controller` annotation. The `@RequestMapping` annotation maps a URL (`/hello`) to the `sayHello` method. The `@ResponseBody` annotation tells Spring MVC to return the method's result as the response.

3. Creating a JSP View

Let's create a simple JSP view to display the "Hello, World!" message returned by the controller.

JSP View: `hello.jsp`

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Hello Spring MVC</title>
</head>
<body>
    <h1>${message}</h1>
</body>
</html>
```

4. Spring MVC Configuration

Configure Spring MVC to use the `HelloController` and render the `hello.jsp` view.

`dispatcher-servlet.xml` Configuration:

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
           http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <context:component-scan base-package="your.package.name" />
    <mvc:annotation-driven />

    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

Make sure to replace `your.package.name` with the actual package where your `HelloController` is located.

5. Run the Application

Now you can run your Spring MVC application. When you access `http://localhost:8080/your-app-context/hello`, you should see the "Hello, World!" message displayed in your browser.

This is a simple example to get you started with Spring MVC. You can expand your application by adding more controllers, views, and models as needed for your project.