Spring MVC uses two popular design patterns

-----------------------------------------------------

- MVC design pattern

- Front controller design pattern

Spring MVC design pattern

------------------------------------

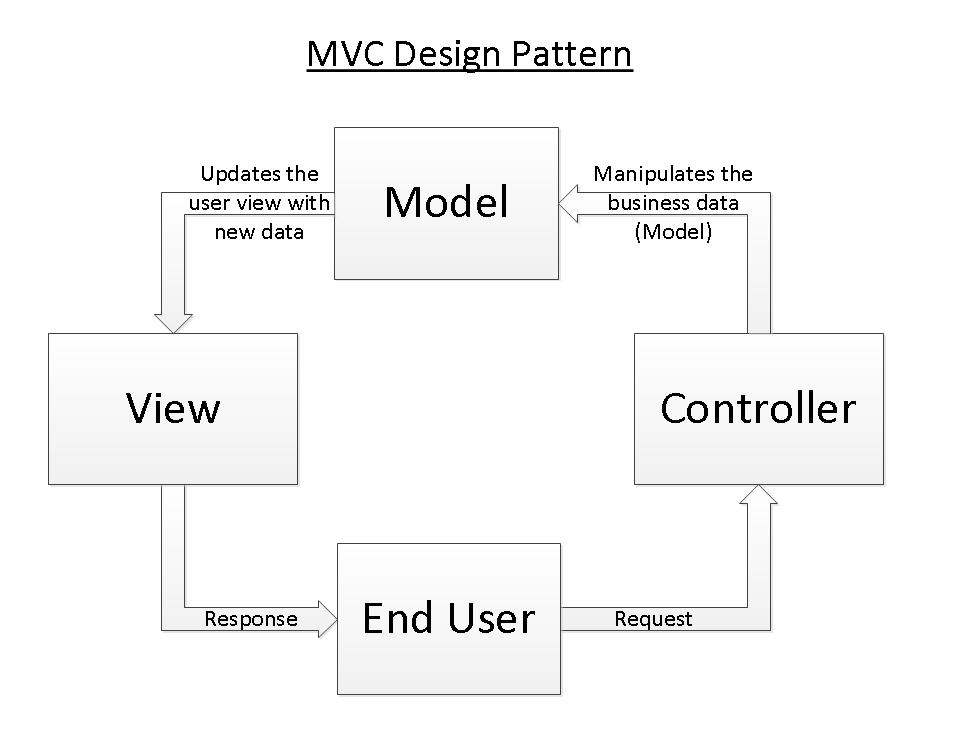
- MVC is a design pattern which provides a solution to layer an application by seperating Business(Model), Presentation(View) and Control flow

M- Model - Business logic

V- View – Presentation logic

C- Controller – Controller logic (flow of the application/ redirection and interaction between view and model)

This design pattern helps us to develop loosely coupled application by segregating various concerns into different layers.

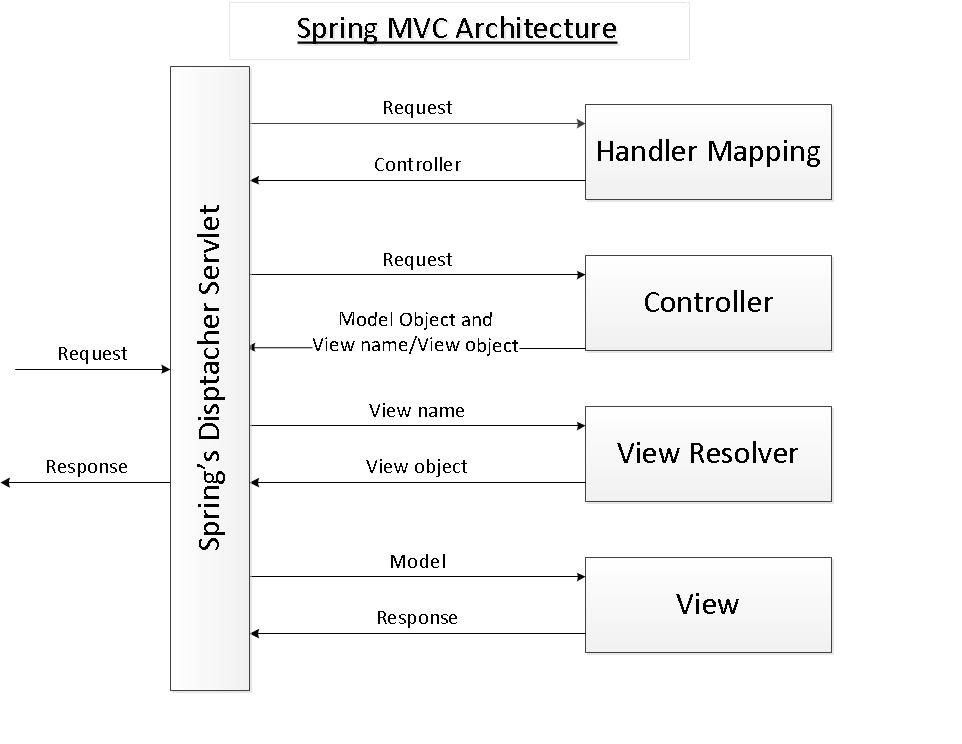


Front controller design pattern

--------------------------------------

- This design pattern enforces a single point of entry for all the incoming request.

-All the requests are handelled by single piece of code which can further delegate the responsibility of processing the request to further application objects.



1.Frontcontroller – servlet – Dispatcher servlet

As we are using the framework – we have to configure this Dispatcher servlet

1. Spring MVC pattern is based on front controller design pattern followed by MVC design pattern.

2.All the incoming request are handelled by single servlet named DispatcherServlet which acts as front controller in Spring MVC module.

3.DispatcherServlet then refers to the Handler mapping to find a controller object which can handle the request.

4. DispatcherServlet then dispatches the request to the controller object so that it can actually perform the bussiness logic to fullfill the user request(Controller may delegate the responsibility to further application object known as service objects)

5.The Controller will return the encapsulated object containing the model object and view Object(or a logical name of the view)

6.This encapsulated object is represented by Class ModelAndView

7.ModelAndView Class contains the logical name of the view, DispatcherServlet refers to ViewResolver(bean) to find the actual view object(path) based on the logical name

8.ViewResolver provides mapping between view name and actual views.

8.DispatcherServlet then passes the model object to the view Object which is then rendered to the end user.

------------------------------------------------------------------------

Download tomcat server

-------------------------------

- Install tomcat server

- Set the path for CATALINA\_HOME and system variables

- check whether the tomcat server runs by using below cmd:

To Start server: <Tomcat Root>/bin>catalina.bat start

To Stop server: <Tomcat Root>/bin>catalina.bat stop

- tomcat server will run in the port 8080: http://localhost:8080/

**For creating WebApp Application (Project)**

- org.apache.maven.archetypes:maven-archetype-webapp

- spring MVC dependancy on pom.xml(spring-webmvc)

*<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.24</version>  
</dependency>

- Configure DispatcherServlet in WEB-INF/web.xml

web.xml

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd" *>*

<web-app>

<display-name>Archetype Created Web Application</display-name>

<servlet>

<servlet-name>springdispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>springdispatcher</servlet-name>

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

</servlet-mapping>

</web-app>

- creating HandlerMapper where web.xml file is present (within WEB-INF)

xxx-servlet.xml

xxx is a servlet-name should be same as dispatcher servlet name(springdispatcher)

springdispatcher-servlet.xml - HandlerMapper

\* component-scan

\* declare bean InternalResourceViewResolver

/WEB-INF/views/yyy.jsp

yyy - view name

If controller gives the view name has success - /WEB-INF/views/success.jsp

springdispatcher-servlet.xml

*<?*xml version="1.0" encoding="UTF-8"*?>*

<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"

xsi:schemaLocation="http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

https://www.springframework.org/schema/context/spring-context.xsd">

<context:component-scan base-package="in.stackroute.controller"/>

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/views/"></property>

<property name="suffix" value=".jsp"></property>

</bean>

</beans>

-----------------------------------------------------------------

**Steps to run Tomcat server**

1.Download the tomcat

2.we have to get smart tomcat from intellij – plugin (download and restart your intellij)

3.Go to settings – tomcat server – provide the path where it is downloaded – apply -ok

4.Right click on project folder – open module settings – delete target folder – apply -ok

5.Run package life cycle

6.Right click on project folder – open module settings – **Artifacts** - **+** - **other**-

give **name** as projectname with war extension (projectName.war) – **archive** – paste the name((projectName.war)).

7. Edit configuration – smart tomcat- give a name for server(ex : tomcat) – deployment directory – target – springMVC (project folder) – build artifacts – add the war file – apply -ok.

8.Run server.

9.*http://localhost:8080:/Name of the project/path*

**JSTL tags**

**Jakarta Standard Tag Library** (JSTL; formerly JavaServer Pages Standard Tag Library) is a set of tags that can be used for **implementing some common operations such as looping, conditional formatting, and others**.

**JSP tags - scritping tags**

**Jakarta Server Pages** (JSP; formerly JavaServer Pages) is a collection of technologies that helps software developers create dynamically generated web pages based on HTML, XML or other document types.  
<%= for loop>  
<% employee.get(o)%>