Model-View-Controller (MVC) : almost the web applications now a days follow this MVS architecture / Design pattern

A basic MVC architecture has a :

1. View (UI) – JSP
2. Controller / Servlet (Mechanism)
3. Model (Storage) Db, file ….

Controller is basically a Servlet that work as an intermediate between View and Model. It takes the data from model and send to View.

We can also write our application business logic inside the controller.

We don’t put any UI/html/css… data/code inside our Controller.

We kinda separate every thing the View in separate file .jsp (only UI Stuffs), the Controller in separate class/servlet ( can write business login inside it as well), and the Model in a separate class to get data from DB (this makes our project more manageable instead of putting all the codes in a single file)

MVC is just a design pattern just to organize things in a simpler way.

Spring boot does a lot of thing for us and make our work a lot easier. First learn spring MVC then we can easily jump to Spring boot.

TO Create a Spring MVC project:

1. Create a Dynamic Web Project
2. Past Spring .jar files in lib folder
3. Configure tomcat server
4. Configure the Front Controller / Dispatcher Servlet in web.xml
5. Create the spring configuration file inside the WEB-INF, the file name must be same as FC/DispatcherServlet name with -servlet.xml at end

Because when the FC object is made it will look for this file in WEB\_INF

{

Once the DispatcherServlet is initialized / the object is created then will look for a file inside the WEB\_INF directory with same name as servlet name that is given in <servlet> tag (it will also append “-servlet.xml” with the servlet/dispacherservlet name)

Create the following file in WEB\_INF

dad-frontController-Dispatcher-servlet.xml

it is the most important file, it is like Spring configuration file we learnt in spring core (beans.xml)

to create this Spring Configuration file: we use the STS (Spring Tools Suit ) feature embedded with Eclips IDE to create this configuration file for us (We can get some Spring stuffs / shortcuts embedded with Eclips IDE)

wc on WEB\_INF > other > Search for : Spring Bean Configuration File > next > give the file name: dad-frontController-Dispatcher-servlet.xml > next > select the name spaces you want in your configuration file:

1-beans

2-context

3-mvc

Next > finish > we will get a spring config file created by STS.

To install JST plugin in our eclips help>eclips marketplace> sts

}

1. We can then configure things for our application inside this configuration file :

Such as:

-Component Scanning

-Bean creation

-Autowiring …etct

1. Create you app Controllers (in Java resources > src) , Controllers are basically java classes
   1. @controller
   2. Add the packages scan in config file (<context:component-scan base-package="com.hom.controllers"></context:component-scan>)

Spring will scan the package and initialize / make object for all classes with @controller and put them inside the container.

1. -to Create view > create a floder (view) in WEB-INF > and put all the jsp (web pages ) inside it

-As Spring developer it is recommended to download the Spring source code so that we can see the Spring internal classes, interfaces, annotation , classes implementations ……we can see all the that by attaching the Spring source code to our workspace. (in order to see how the classes in the jar file are written)

<https://github.com/spring-projects/spring-framework>

**Front Controller (Dispatcher Servlet):** accepts any request comes to our website and rout / dispatch the request to the respective Controller (with same url patter in the request and the Controller)

We don’t need to create the front controller, The Spring framework developers have already created a front controller the name of the front controller is : DispatcherServlet. We can simply use/reuse this front controller in our project

To see the Class from Jar files : ctl+shft+t : dispatcherservlet –Attach source

We need to configure our Front Controller inside the web.xml

<servlet>

<servlet-name>dad-frontController-Dispatcher</servlet-name>

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

</servlet>

<servlet-mapping>

<servlet-name>dad-frontController-Dispatcher</servlet-name>

<url-pattern>/home.com-MVC-Project/\*</url-pattern> <!-- telling the Dispatcher/FC to handll all the request coming to /home.com-MVC-Project/\* -->

</servlet-mapping>

Simple way to create / configure the front controller in our web.xml , instead of writing these all manually

Wc on project > new > Servlet > Use an existing Servlet or JSP > brows > Search for DispatcherServlet > ok > next (the IDE will write all the code for us)

We just configured the DispatcherServlet in our web.xml when it will be initialized / it object will be created ?

The DispatcherServlet object will be initialized when we deployed our project / application inside the tomcat server

<load-on-startup>1</load-on-startup> <!-- this means that when the server is started the DispatcherServlet will be initialiez, and if we dont give load-on-startup the FC will be initialized when the first request comes to the project addres /home.com-MVC-Porject/\* -->

Once the DispatcherServlet is initialized / the object is created then will look for a file inside the WEB\_INF directory with same name as servlet name that is given in <servlet> tag (it will also append “-servlet.xml” with the servlet/dispacherservlet name)

Create the following file in WEB\_INF

dad-frontController-Dispatcher-servlet.xml

it is the most important file, it is like Spring configuration file we learnt in spring core (beans.xml)

to create this Spring Configuration file: we use the STS (Spring Tools Suit ) feature embedded with Eclips IDE to create this configuration file for us (We can get some Spring stuffs / shortcuts embedded with Eclips IDE)

wc on WEB\_INF > other > Search for : Spring Bean Configuration File > next > give the file name: dad-frontController-Dispatcher-servlet.xml > next > select the name spaces you want in your configuration file:

1-beans

2-context

3-mvc

Next > finish > we will get a spring config file created by STS.

To install JST plugin in our eclips help>eclips marketplace> sts

We can then configure things for our application inside this configuration file :

Such as:

-Component Scanning

-Bean creation

-Autowiring …etct

**Web Container / WebApplicationContext / Spring web MVC project flow**

**How the container is created in Web Application (MVC)**

* The DispatcherServlet gets initialized at the time of deployment of our project on tomcat. (when tomcat starts). Because we mentioned (Because we mentioned (<load-on-startup>1</load-on-startup> ) in our web.xml

Then the object will look for a file same as name of servlet (config file). If the file not found it will through a file not found exception

Once it found the file It will created a container out of the file called WebApplicationContext

**(previous in Spring core while developing stand alone applications we made the container (context) from ApplicationContext.**

**But here in web application containers are make automatically from WebApplicationContext.**

**Since we deal with a web application we must make a WebApplicationContext container.**

**We don’t create this container and we don’t have to start and close the container it is done by Spring**

In web apps we deal with WebApplicationContext container: to see it ctrl+shft+t > webapplicationcontext …..

WebApplicationContext extends ApplicationContext : it has all the features of ApplicationContext plus extra features to handle web applications

**How the object for Controllers are made.**

Once the FC/DispacherServlet object is made it will look for config file, in config file we have the <Context:Component-Scan…../> tag. It will scan through the package and make object for any class having the @controller / @ component annotation , and their objects will be register inside the container.

How it calls a specific handler method?

All the url mappings are registered with dispatcher servlet by what dispatcher servlet will decide which handler method to call.

Ctrl+shft+o : to delete unused imports

Multi Action Controller / @RequestMapping at class level:

A controller can have as many handler method as want. With specife url mapping. (a controller can handle multiple url requests)

How to change DispatcherServlet context configuration file name ( how to customize Spring config file name)

It is not mandatory that our Spring config file should be same as name of our Front Controller / DispatcherServlet name (which is in web.xml) with -servlet.xml at end.

<servlet>

<servlet-name>Dad-FC-DispatcherServlet</servlet-name>

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

<init-param>

<param-name>contextConfigLocation</param-name> <!-- to let the DispatcherServlet know dont search for your default Spring Config file name, -->

<param-value>WEB-INF/MyCustomName.xml</param-value> <!-- our custome Config file name.. it can be any where we just need to put it url -->

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

Views in Spring MVC (views means pages)

@ResponseBody : above the method name inside controller means that whatever string the method returns goes in http response body and displayed in browser.

If we don’t mention it the method return message will go to DispatcherServlet and the DispatcherServlet looks for a view. That is why we will get 404 error.

What ever the method return the DispatcherServlet will think it is a view (page name)

To make our controller return a web page instead of text we us view

-to Create view > create a floder (view) in WEB-INF > and put all the jsp (web pages ) inside it

View Resolver:

Using view resolver technology we can make our project more flexible and loosely coupled.

(the handler method should not return a view path/url . because what if we want to change the view folder from WEB-INF to some where else, what if we change the technology used in view to something else fe. Instead of jsp , what if we want to use html……. Then we can to bring these changes to all the handler methods return statement)

To make our project flexible and loosely coupled with these we use the View Resolver in our project.

@RequestMapping("/cricketbat")

**public** String giveCricketBat()

{

//return "Ok...Here is the Cricket Bat....";

//to make the controller return a view / webpage

// return "/WEB-INF/view/CricketBat.jsp"; // this way is tightly coupled. instead better to use View Resolver

// the view location and the view extention might be changing. we can solve this issue using the View Resolver (the view name does not change)

//the way to return a view is to return only the view name with no extention and path

**return** "CricketBat"; // base name.

// this name will be returned to the DispatcherServlet, DispatcherServlet doesnt know where this file is

//located and what the extention to this file is. to resolve it we use the View Resolver

//DS passes the name to the view resolver, view resolver add the path and extention to the name and send it back to DS

}

There are plenty of ViewResolvers in Srping framework

The one we will use is InternalResourceViewResolver. It is class written by developer

Model:

Model means data, we can get the data from database/user/file …