SpringBoot-MVC

PreRequisite :: CoreJava + SpringCore + SpringDataJPA(JDBC,Hibernate) + Servlet,JSP,JSTL(basics knowledge)

Different ways of Developing Java Based WebApplications

Model1Architecture

=> In this architecture either servlet or jsp components is used as main components of webapplication
=> If servlet component is used, then jsp won't be used or vice-versa.
=> Here each main component contains multiple logics, ie no clear cut seperation b/w logics.
=> It is not suitable for large scale applications.

Model2Architecture

=> MVC1 Architecture, MVC2 Architecture
=> Here we take support of multiple technologies in multiple layers to develop the logics.
=> Suitable for Medium and Large scale applications.
=> M(model) V(view) C(Controller)
=> MVC2 Architecture is more industry standard.
=> All Webframeworks like Struts,JSF,Spring-MVC,Spring-Boot-MVC etc are given based on MVC2 Architecture.
=> MVC means MVC2-Architecture only.
=> MVC3,MVC4,MVC5,MVC6 are no way related to java architecture, therse are version of asp.net mvc

What is the difference b/w MVC1 and MVC2 Architecture?

=> In MVC1 we take single component(Servlet/jsp) acting as both view and controller,but we take seperate components for other layers.
=> In MVC2 we take seperate component for View(jsp), seperate Component for Controller(servlet) and seperate Componet for Model layer.
refer: Spring_mvc_01.png

W.r.t diagram

1. Browser gives the request
2. Controller servlet traps and takes the request[Depending on the url pattern]
3. Using Navigation logic the Controller-Servlet passes the request to appropriate components(Service,DAO)
4. The service,DAO component process the request.
5. DAO component interacts with Backend system(DB S/W) through model
6. The result generated will be transferred to Service and then it will be forwared to controller.
7. The controller component passes the result to view component.
8. View component formats the result using Presentation logic.
9,10. The formatted results goes to browser as the response.

MVC2 Architecture pros and cons

Pros

a. Since there are mulitple layers in application layers we can say there is a clear cut seperation b/w logics.
b. The modification done in one layer does not affect the other layers.
c. Maintainence and Enchancement of the project becomes easy.
d. Parallel development is possible so productivity is good.
e. It is industry standard architecture to develop java based medium scale and large scale websites.

Cons
  a. Since there is a parallel development we need more programmers.
  b. Knowledge of multiple technologies is required.

MVC2 architecture rules or principles
==========
 => It is not just working with multiple technologies in mulitple layers, there are set of rules we need to follow
      a. Each layer is given to place bunch of logics, just place only such logics, don't add anyother logics.
      b. There can be multiple view comps, multiple model comps,But it is recomended to take Single controller component.
      c. All the operations must take place under the control of Controller component.
      d. View components must not interact with model component directly or viceversa,they must interact with servlet component.

When we have MVC2 architecture to develop webapplications as layered applications, what is the need of webapplication frameworks in java?
 => If we develop MVC2 architecture manually by using servlet,jsp technolgies then
       a.Programmer should develop all the logics manually
       b.Programmer should take care of navigation management
       c.Programmer should take care of view management
       d.Programmer should take care of data management.
       e.Should remember and implement MVC2 rules
       f.Chances of having boiler plate code.

To overcome the above limitations and to provide abstration over jee technologies in webapplication development we can take the support of
MVC2 framework or java webapplication frameworks they are
            a. struts -> from apache
            b. JSF   -> from SUNMS
            c. Webwork -> from opensympohny
            d. SpringMVC/SpringBoot MVC -> from interface21/pivotal team
            e. ADF   -> from Oracle corportation

Advantages of developing MVC2 architecture based webapplication development using webapplication framework
       a. Gives ready made Controller-Servlet,no need to write controller logics manually.
       b. Automatically implements maximum of MVC2 Rules
       c. The ready made controller servlet can trap all or multiple request to apply common system services like auditing,logging,security etc
       note: ready made servlet will be given based on "FrontController" design pattern, so it is called as "FrontControllerServlet".
       d. FrontControllerServlet takes care of navigation management
       e. FrontControllerServlet takes care of view management
       f. FrontControllerServlet takes care of data management.
       g. Lots of boiler plate code will be avoided.

Note: In Struts, the FrontControllerServlet name is "ActionServlet".
       In JSF, the FrontControllerServlet name is "FacesServlet".
       In SpringMVC, the FrontControllerServlet name is "DispatcherServlet".

IS MVC1,MVC2 and Model1 are they design patterns or architecutre?
 => These are architecure to develop the java based applications.

What is the difference b/w Architectrue and Design pattern?
 => Architecture speaks about involving multiple components and their flow exeuction in the application development.
 => Design patterns speaks about the best solution for recurring problems in the application development.
 => In the implementation of Architecture multiple design patterns will be used

eg: MVC2 Architecture speaks about how to involve multiple components in layered web application development.
    In each layer multiple Design patterns will be used to solve the commonly recurring problems
            Contoller layer :: FrontController,ApplicationController,Intercepting filter etc ,design patterns will be applied.
            View layer   :: View Helper,CompositeView, .. design patterns will be applied.
            Model layer   :: Service Delegate/Buisness Delegate, DAO and etc design pattern will be applied.

What is FrontController Servlet?
 => The special web component/servlet component of MVC2 Architecture of java based webapplication who can trap either all request or multiple
    requests to apply common system services like auditing,logging,security etc and also takes care of navigation management, view management
    model/data management is caleld "FrontController Servlet".

Navigation Management => Decides the flow among Components.
View Management   => Decides Which model/service/handler class results should go to which view component(jsp or other component)
Data/model Management => Talks about how to store input values(form data) and how to pass the generated results to various components keeping
                         them in particular scope.

=> Once the front controller is involved in MVC2 Architecture webapplication,there will be only one servlet in the entire webapplication that is
   FrontController servlet.
=> We keep java classes as "Handler/Controller/Action" classes in webapplications either to process the request directly or to delegate the
   request to service class by taking from FrontController-Servlet.

Note:
a. FrontController is only Servlet Component.
b. Handler/Controller/DAO classes are plain java classes.
c. View components are generally JSP/html/Thymleaf etc...

refer:: Spring_mvc_01.png

Note:
 With respect to diagram
            a. To make front controller servlet trapping multiple requests take the support of extension match or directory match url pattern
                eg::Extension Match :: *.do, *.ineuron, *.all,....
                eg::Directory Match :: /x/y/* , /abc/ijk/*, /nitin/ineuron/*
            b. To make front controller servlet trapping all the request take the support of "/".

What is the difference b/w FrontController and Controller/Handler/Action classes in MVC2Architecture and FrontController Design Pattern?

FrontController
=> It is a web component(generally it is a servlet/filter web component)
=> Traps either all request/multiple requests and applies common System Services.
=> Generally managed by Servlet Container.
=> Gets request,response object that are created and given by Servlet Container.
=> The main method containing logics are servlet life cycle methods or convinence methods like doXXXX(,,)
=> Genernally it is one per entire MVC application.

Controller/Handler/Action
=> It is a java class(Spring bean in SpringMVC)
=> Either directly contains request processing logics or contains logics to delegate request to service class.
=> Managed by JVM of WEbServer(normal webapplication), In case of SpringMVC webapplication it is managed by IOC Container.
=> Gets request,response object that are passed by FrontController
=> Normally have methods that contains buisness logic or delegation logic are called "handler" methods.
=> Generally it is one per module.

Note:
=> In SpringMVC/SpringBoot-MVC the FrontController is "DispatcherServlet" which is ready made ServletComponent.
=> Every Servlet Component(Either Pre-defined/readymade/user-defined)must be configured with Servlet Container and also must be linked
with url pattern(either with "/" or with "directory/extension" match urlpattern).

3 ways of Servlet Configuration with Servlet Container
==========
1. Declarative approach
=> It is done by using web.xml file
=> if the servlet component is ready made and xml config are allowed in apps.
eg:: DispatcherServlet configuration in XML driven SpringMVC apps and in XML + Annotation apps.

```
<web-app>
	<servlet>
		<servlet-name>dispatcher</servlet-name>
		<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
		<init-param>
			<param-name>contextConfigLocation</param-name>
			<param-value>/WEB-INF/dispatcher-servlet.xml</param-value>
		</init-param>
	<load-on-startup>1</load-on-startup>
	</servlet>

	<!-- Set up URL mapping for Spring MVC Dispatcher Servlet -->
	<servlet-mapping>
		<servlet-name>dispatcher</servlet-name>
		<url-pattern>/</url-pattern>
	</servlet-mapping>
</web-app>
```

2. Annotation driven approach
=> It is done by using @WebServlet
=> If the servlet component is userdefined then we can configure in annotation drive approach
eg: Building webapplication using Model1,MVC1,MVC2 and having FrontController in those applications.

```
@Configuration
@ComponentScan({ "in.ineuron.web" })
public class MVCconfig extends WebMvcConfigurerAdapter {

}
```

```
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class WebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected Class<?>[] getRootConfigClasses()
    {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    protected Class<?>[] getServletConfigClasses()
    {
        // TODO Auto-generated method stub
        return new Class[] { MVCconfig.class };
    }

    @Override
    protected String[] getServletMappings()
    {
        // TODO Auto-generated method stub
        return new String[] { "/" };
    }
}
```

3.Programmatic approach
=> It is done using ServletContext.addServlet("") method
=> It is useful to configure PreDefined/Readymade servlet Component in webapplication where xml configuration are not allowed.
eg: In 100%code driven SpringMVC apps,In SpringBoot MVC apps xml configuration are not allowed, so to configure DispatcherServlet Component this programmatic approach will be used either directly or indirectly.

Different approaches to develop SpringMVC applications
==========
1. Declartive Cfgs Approach(XML driven cfgs) ======> web.xml file we configure DispatcherServlet
2. Annotation Cfgs Approach(XML + Annotation driven cfgs) ====> In useredefined class, with @Configuration set we configure "DispatcherServlet"
3. 100%code driven configuration

4. SpringBoot MVC appps

Note::
 => In 1st,2nd approach DispatcherServlet will be configured using web.xml
 => In 3rd,4th approach DispatcherServlet will be configured using Programatic/DynamicApproach
 => The source code of DispatcherServlet doesn't contains @WebServlet Annotation.
 => All SpringMVC/SpringBootMVC applications that are developed in different approaches are based on MVC2Architecture and FrontController(Dispatcher Servlet) design pattern.

While building SpringMVC/SpringBootMVC applications we need to use the following annotations while developing handler classes
	a. @Controller => To make java class as SpringBean + handler/controller class
	b. @RequestMapping => To mark java method of @Controller class as handler method having requestpath and request mode/method(POST/GET)
Note: As of now browser can send only 2 types of request[GET(default)/POST]

eg#1.
```
@Controller
public class LoginController{
	@RequestMapping(value="/login" method =RequestMethod.GET)
	public String login(parameters...){
		//logic for validation or logic to delegate the request to service class
	}
}
```

The possible parameters for Handler method arguments are
	a. javax.servlet.ServletRequest
	b. javax.servlet.ServletResponse
	c. @PathVariable
	d. @RequestParam
	e. @RequestHeader
	f. @RequestAttribute
	g. @ModelAttribute
	h. Errors,BindingResult
	i. @SessionAttribute

The possible return types of Handler methods
	a. String
	b. View
	c. Model
	d. @ModelAttribute
	e. ModelAndView
	f. void

SpringBoot MVC Flow/Spring MVC Flow
==========
 => It is designed around Dispatcher Servlet which is predefined FrontController Servlet Component.
 => All the activities in SpringMVC takes place under the control or monitoring of DispatcherServlet Component.

Note:
 => DispatcherServlet takes the support of HandlerMapping Component to link the

given request with Handler method of Controller class.
    DispatcherServlet gives Requestpath and RequestMode to HandlerMapping Component and gets the RequestMappingInfo object having
    Controller class beanid and HandlerMethod signature.

 => DispatcherServlet takes the support of ViewResolver Component to map the given request related results to one of View Component
    and gets View Object having physical view component name and location.

 => In SpringMVC View component is Abstract Entity ie we can take any things in view component like jsp/html/freemarker,velocity components.

refer:: Spring-mvc-01.png

Note::
EmbededEnvironment
==========
Application---> Started =======> JRE,TomcatContainer(virtual)
                                                    |
                  Stopped <======= JRE(removed) TomcatContainer(removed)

W.r.t Diagram
==========
 1. Programmer deploys SpringMVC/SpringBootMVC application in webserver or webapps.
 2. Deployment activities takes place which involves IOC container creation ,DispatcherServlet registration with ServletContainer
    Pre-Instantiation of Singleton scope spring beans like Controller class,handlermapping,viewResolvers,Service class,DAO class etc.
    In the mean time necessary dependancy injection also takes place.
 3. Browser gives request to deployed springmvc application.
 4. The frontcontroller(DispatcherServlet) traps the request and applies the common system services on the request
       like logging,auditing,tracking etc,...
 5. DispatcherServlet hands over the request to HandlerMapping component to map incoming request with handler method of handler/controller
    class and gets RequestMapping object from HandlerMapping component having Controller class bean id and HandlerMethod signature(it uses lot
    of reflection api code internally).
 6. DispatcherServlet takes controller bean class id from the recieved RequestMapping object and gets the Controller class object from
    DispatcherServlet created IOC container.DispatcherServlet also prepares the necessary objects based on the method signature of the
       handled method collected from RequestMapping Info object.
 7. DispatcherServlet calls handler method having necessary object as the arguments on the above received Controller class object.
 8. The handler method of the controller class either directly process the request or takes the support of service/DAO and keeps the result
    in a scope(prefereablly in request scope)
 9. The handler method of Controller class returns LVN(logical view name) back to Dispatcher Servlet.
 10. DispatcherSerlvet gives LVN to ViewResolver
 11. ViewResolver map/link LVN to PhysicalView component and returns View Object having PhysicalViewName and location.
 12. DispatcherServlet gets PhysicalViewName and location from the recieved View Object and send the control to Physical view component
     using rd.forward(,) to format the results gathered from particular scope(preferabley request scope) using presentation logics.
     These formatted results goes back to DispatcherServlet.
 13. DispatcherServlet sends the formatted result to browser as the response.

HandlerMapping
==============
 => This component takes the incoming request through DispatcherServlet and maps the request with appropriate handler method of appropriate
    handler/controller class by matching the incoming url path with handler request method path using reflection api and returns
    "RequestMappingInfo" object back to DispatcherServlet having the mapped controller class bean id and handlerMethod signature.

 => All HandlerMapping comps are the classes implementing org.springframework.web.servlet.HandlerMapping(I)
 => Generally we work with ready made "HandlerMapping" components
            a. BeanNamedUrlHandlerMapping[Default in XML driven configuration]
            b. RequestMappingHandlerMapping[Default in annotation driven configuration,100%code driven configuration,SpringBootMVC]
            c. ControllerClassHandlerMapping
            d. SimpleUrlHandlerMapping
                        etc...

```
<beans>
  <bean id="handlerMapping"
class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"/>
<beans>
```

ViewResolver
============
 => This componet takes the recieve logicalViewName with physical view component returns View Object having the name and location of physical
    view component back to dispatcher servlet.
 => It can collects the inputs either in the form of xml file(XMl driven configuration) or from properties file(SpringBoot App)
 => ViewResolver Components are the classes that implements an interface called "org.springframework.web.servlet.ViewResolver(I)"
 => Generally we work with ready made "ViewResolver" components
            a. UrlBasedViewResolver
            b. InternalResourceViewResolver(default in SpringBootMVC)
            c. ResourceBundleViewResolver
            d. XmlViewResolver
            e. TilesViewResolver
            f. FreeMarkerViewResolver
            g. BeanNameViewResolver
      Note: No default ViewResolver in xml,annotationdriver and 100%drivenconfiguration

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

=> InternalViewResolver is capable of taking jsp,html,servlet component placed in private area of view component through html,jsp components
   are not configured in "urlpatttern".
=> In SpringMVC,SpringBootMVC we generally take "jsp" comps as the view component

and it is even recomeneded to keep those jsp components in
   "private" area.

Note: Outside of WEB-INF area is called public area of webapplication,the request to these components can be given directly without giving
      mapping details to ServletContainer.
       Inside WEB-INF area is called private area of webapplication, the request to private are components must be given only after giving
      mapping details to ServletContainer.

Advantages of placing jsp components in private area
==========
 => Helps to hide technology of the webapplication becoz the jsp files names does not appear in the request urls,so many things does not
    appear in the browser address bars.This helps to protect apps from hackers and jackers.
 => Protects the source code of the jsp components from outsiders
 => If the jsp component is developed to display request scope data given by servlet component, then direct request to jsp component by
    by keeping jsp in public area may give null values or ugly values.[so to stop that place jsp comp in private area]
                                        refer:: spring-mvc_01.png

=> In our SpringMVC/SpringBoot-MVC keeps jsp component in private area like [WEB-INF\pages] then InternalResourceViewResolver needs 3 details
    to locate these jsp components.
                a. Location of jsp components as the prefix info
                b. suffix of jsp components
                c. jsp filename as the LVN

 application.properties
 ==========
 spring.mvc.view.prefix =/WEB-INF/pages/   #location of view
 spring.mvc.view.suffix = .jsp   #Extension of Technology

Note: In SpringBoot-MVC Apps the following components comes automatically so we need not develop them
      a. DispatcherServlet
      b. IOC-Container
      c. HandlerMapping
      d. ViewResolver
                To get all these we need to go for a starter file called "spring-boot-starter-web".