**Spring MVC :::::::::**

A Spring MVC is a Java framework which is used to build web applications.

It follows the Model-View-Controller design pattern.

It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.

**Spring Web Model-View-Controller**



* **Model** - A model contains the data of the application. A data can be a single object or a collection of objects.
* **Controller** - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.
* **View** - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.
* **Front Controller** - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.



* **As displayed in the figure, all the incoming request is intercepted by the DispatcherServlet that works as the front controller.**
* **The DispatcherServlet gets an entry of handler mapping from the XML file and forwards the request to the controller.**
* **The controller returns an object of ModelAndView.**
* **The DispatcherServlet checks the entry of view resolver in the XML file and invokes the specified view component.**

Advantages of Spring MVC Framework

Let's see some of the advantages of Spring MVC Framework:-

* **Separate roles** - The Spring MVC separates each role, where the model object, controller, command object, view resolver, DispatcherServlet, validator, etc. can be fulfilled by a specialized object.
* **Light-weight** - It uses light-weight servlet container to develop and deploy your application.
* **Powerful Configuration** - It provides a robust configuration for both framework and application classes that includes easy referencing across contexts, such as from web controllers to business objects and validators.
* **Rapid development** - The Spring MVC facilitates fast and parallel development.
* **Reusable business code** - Instead of creating new objects, it allows us to use the existing business objects.
* **Easy to test** - In Spring, generally we create JavaBeans classes that enable you to inject test data using the setter methods.
* **Flexible Mapping** - It provides the specific annotations that easily redirect the page.

# **Spring MVC RequestParam Annotation**

In Spring MVC, the **@RequestParam** annotation is used to read the form data and bind it automatically to the parameter present in the provided method. So, it ignores the requirement of **HttpServletRequest** object to read the provided data.

Including form data, it also maps the request parameter to query parameter and parts in multipart requests. If the method parameter type is Map and a request parameter name is specified, then the request parameter value is converted to a Map else the map parameter is populated with all request parameter names and values.

# **Spring MVC Form Tag Library**

The Spring MVC form tags are the configurable and reusable building blocks for a web page. These tags provide JSP, an easy way to develop, read and maintain.

The Spring MVC form tags can be seen as data binding-aware tags that can automatically set data to Java object/bean and also retrieve from it. Here, each tag provides support for the set of attributes of its corresponding HTML tag counterpart, making the tags familiar and easy to use.

Like

1)drop down list

2)Radio button

3)checkbox

4)field

# **Spring MVC Pagination Example**

Pagination is used to display a large number of records in different parts. In such case, we display 10, 20 or 50 records in one page. For remaining records, we provide links.

**BASIC CRUD FLOW**

as we known we have to make a folder as view folder in src.

so in that make two files .xml file one will web.xml and another one is spring-servlet.xml file

also will provide the html files there for pages views

**1)web.xml**

:::

talking about this xml file it will work as a dispatcher servlet means whatever

request is coming will be handeled by this

{

now

question is like can we have multiple dispatcher servlet in mvc???

------->

Yes, a Spring MVC web application can have more than one DispatcherServlets.

Each DispatcherServlet has to operate in its own namespace.

It has to load its own ApplicationContext with mappings, handlers, etc.

Only the root application context will be shared among these Servlets.

when we will use multiple dispatcher servlet ??

-------->

when u have to handled multiple request and show their response beyond the limit

then we have to provide so it will handled all this request in a synchronized manner

}

**2)spring-servlet.xml**

1)In this class we will scan our base packages by using the tx:annotation-driven tag

2)Will gave the data source to provide the connectivity to the databases

3)will gave the hibernate properties

4)annotated class is name given to the property in that we have to define the model class

so it will accesss all the fields required in the pojo class

5)Lastly we have to provide hibernate template to manupulate the databases.

**Now in src/main/java**

we have to provide::

1)one pojo class

2)one dao class

3)one controller::

In controller we have to use @controller annoatation

becaue we have to return something to cntroller method.