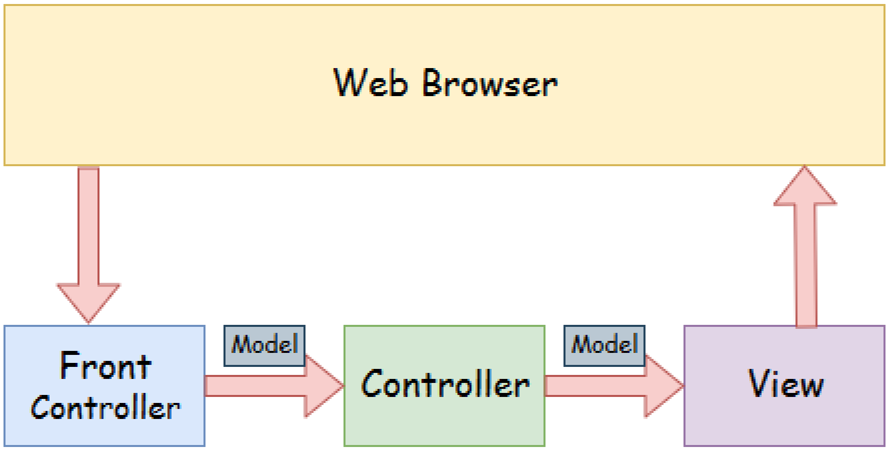
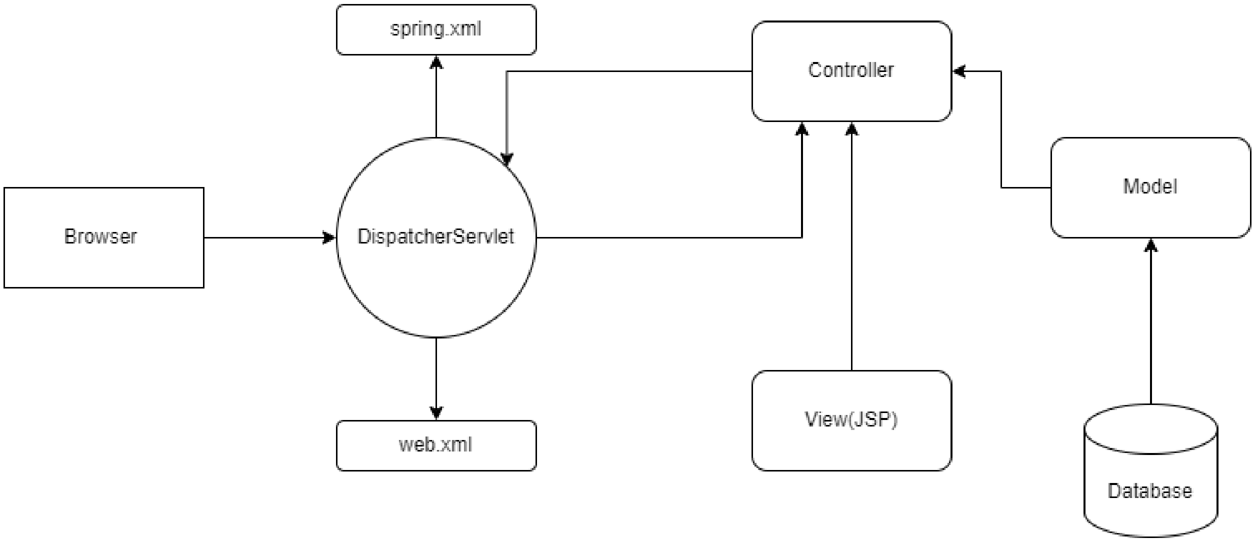
Spring MVC

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.

A Spring MVC provides an elegant solution to use MVC in spring framework by the help of **DispatcherServlet**. Here, **DispatcherServlet** is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.



- **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.



1. Web Application runs in a browser where requests are generated by clicking the button, clicking on hyperlinks, etc.,
2. The first request should be reached to DispatcherServlet
3. DispatcherServlet is defined in web.xml and we can configure the spring XML in the same file.
4. DispatcherServlet knows what controller to execute based on the URL selected and executed in web browser.
5. Controller is a class where we can implement logic for the functionality, attaching model and redirecting to the right view.

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 Validation

The Spring MVC Validation is used to restrict the input provided by the user. To validate the user's input, the Spring 4 or higher version supports and use Bean Validation API. It can validate both server-side as well as client-side applications.

Bean Validation API

The Bean Validation API is a Java specification which is used to apply constraints on object model via annotations. Here, we can validate a length, number, regular expression, etc. Apart from that, we can also provide custom validations.

As Bean Validation API is just a specification, it requires an implementation. So, for that, it uses Hibernate Validator. The Hibernate Validator is a fully compliant JSR-303/309 implementation that allows to express and validate application constraints.

Validation Annotations

Some of the most widely used annotations are:

Annotation	**Description**
@NotNull	It determines that the value can't be null.
@Min	It determines that the number must be equal or greater than the specified value.
@Max	It determines that the number must be equal or less than the specified value.
@Size	It determines that the size must be equal to the specified value.

@Pattern	It determines that the sequence follows the specified regular expression.
@Positive	It determines that the value entered should be positive
@Negative	It determines that the value entered should be negative
@Email	It determines that the value entered should be in email format