Spring Framework:-

- Spring is a framework of frameworks.
- It provides a model of configuration and comprehensive programming for the development of any java application.
- The major goal of spring is to handle infrastructure or configuration so that the developer can focus on business logic.
- Spring Framework encourages a modular approach, making it easier to develop and maintain code.
- Dependency injection promotes writing testable code, and the framework integrates well with testing tools.
- In Spring the object which are handled by spring container are known as beans.
- The most prominent feature of Spring are:-

1. IOC - Inversion of control
2. DI - Dependency Injection

1. IOC :-

- It stands for Inversion of Control.
- The process of inverting the control of object creation and lifecycle management by the developer from the application code to the Spring container is known as IoC.
- Spring IoC is a core principle of the Spring Framework that handles the creation and management of objects and their dependencies.

2. DI :-

- The process of injecting One Object inside another Object as a dependency in spring is called as Dependency Injection.
- In other words we can say injecting one class object inside another class without actually creating the object inside it using the new keyword, but taking the object from the spring bean container.
- In order to perform DI both the classes object creations must be handled by the Spring Container.

BeanFactory(I):-

- BeanFactory is a basic container that provides fundamental features for managing bean objects.
- It is present in org.springframwork.beans.factory package.
- It is considered as a Lazy loader because it creates object for the desired class only when it is requested by invoking the getBean() method from the BeanFactory.
- It is memory efficient as compared to ApplicationContext as it doesn't create object for all the bean definitions declared in the spring.xml file once it is loaded.

- We can give implementation to BeanFactory interface by creating an object of the XMLBeanFactory which is in composition with FileSystemResource and pass the spring configuration File name inside the constructor of FileSystemResource.
- FileSystemResource can read the spring configuration file from the file path itself that mean we need to create the xml file directly inside the project in order to access it.

ApplicationContext(I):-

- It is a child interface of BeanFactory also it is an advanced container with some advanced features
- It is present in org.springframework.context package.
- It is considered to an Eager loader because it creates object for all the bean definition declared in the spring.xml file
- once it is loaded into the container. So, when the user invokes the getBean() method for the 1st time it provides the object which is already created.
- It is not memory efficient as compared to BeanFactory as it create objects for all the bean definition declared in the spring.xml file once it is loaded and won't wait for the user to ask for it.
- We can give implementation to ApplicationContext by creating object of either of the two subclasses of it i.e. ClassPathXmlApplicationContext and AnnotationConfigApplicationContext.
- ClassPathXmlApplicationContext can read the spring configuration file form the class path that means we need to place the xml file inside the src/main/java folder.

There are two ways to configure the bean:-

1. XML based Configuration

2. Annotation based Configuration

Annotation Based Configuration:-

- In annotation based configuration we don't use xml file for the bean configuration in spring, instead we use some specific annotations over the classes for which we want spring to create the object and maintain the lifecycle of the object.
- Here we declare a AppConfig class with @Configurartion and @ComponentScan annotation where we provide the details of the packages that are supposed to be scanned by spring.
- In this case we use another sub-class to provide implementation to the ApplicationContext interface i.e, AnnotationConfigApplicationContext and pass the configuration class name in the constructor.

- So, once the Configuration class is found and the basepackages details is read, then spring scans all the classes present in that package as well as its sub-packages, and create beans for all the classes annotated with stereotyped annotations.

Annotations in Spring Framework:-

1.@Configuration:-

- The @Configuration annotation is used to indicate that a class declares one or more bean definitions and is a source of bean definitions for the Spring application context.
- It is a fundamental building block for creating Spring-based applications, particularly when using Java-based configuration instead of XML.

2.@ComponentScan:-

- We use @ComponentScan annotation always with @Configuration annotation in order to scan all the stereotyped annotated class present in the package and subpackages for bean creation.

3.@Bean :-

- The @Bean annotation is used to declare a method as a bean definition method within a Spring configuration class.
- When Spring's container is initialized, the methods annotated with @Bean are Invoked, and the objects returned by these methods are managed as Spring beans within the container.

Spring MVC:-

- MVC Stands for Model View Controller.
- Spring MVC is a web framework provided by the Spring for building web applications in Java.
- Spring MVC architecture is majorly based on client-server architecture.
- It follows the MVC architectural pattern, which divides an application into three interconnected components: Model, View, and Controller.

1. Model

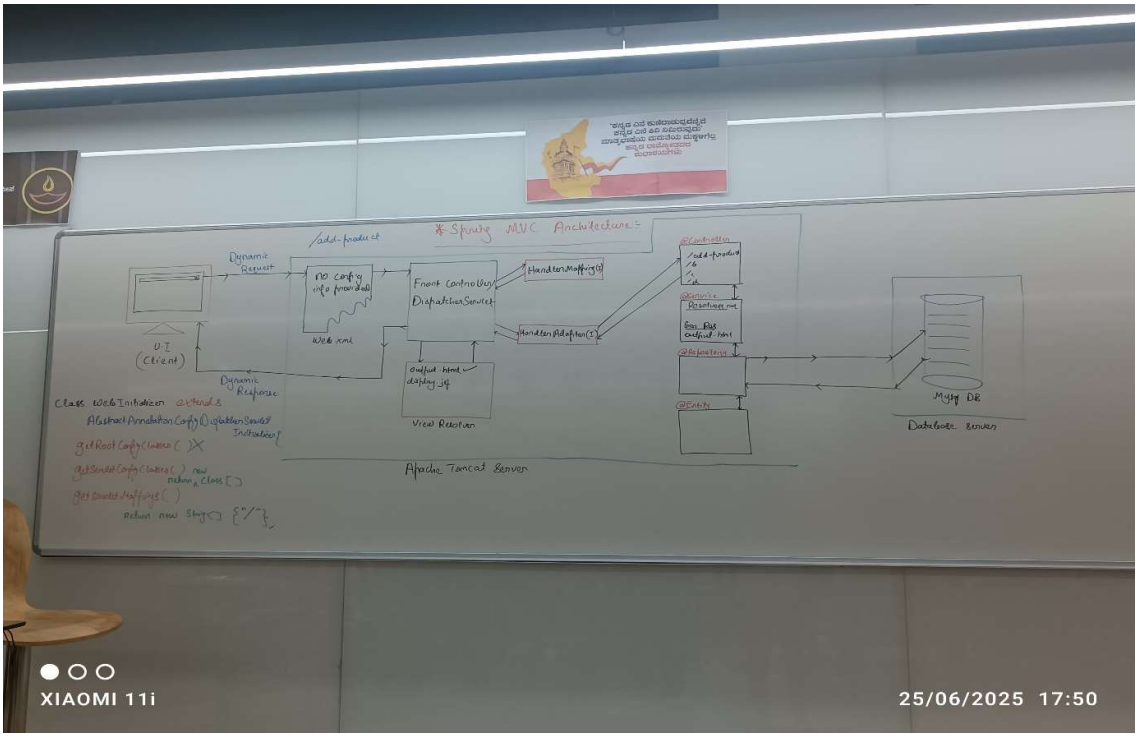
- The Model represents the application's data and business logic. It encapsulates the state of the application and provides methods to manipulate that data.
- In Spring MVC, the Model typically consists of objects that represent entities in the application, such as users, products, orders, etc.

2.View

- The View represents the presentation layer of the application, responsible for rendering the user interface based on the data provided by the Model.
- Views are responsible for generating HTML output that is sent back to the client's web browser for display.

3.Controller

- The Controller acts as an intermediary between the Model and the View. It receives incoming requests from the client, processes them, and determines how to respond.
- In Spring MVC, Controllers are implemented as Java classes annotated with @Controller or @RestController annotations.
- Controllers contain handler methods annotated with @RequestMapping that map specific URL patterns to business logic.



Flow of a request in Spring MVC Architecture: -

- Whenever a request is triggered from client it hits the configuration file known as web.xml file (Deployment Descripter).
- If there is no information present in the web.xml file it tries to search for the DispatcherServlet class.
- We use a custom initializer class which must be a child of AbstractAnnotationConfigDispatcherServletInitializer (Abstract class) as a replacement of web.xml file.

- The DispatcherServlet is responsible for 2 jobs using its specific overridden methods, first job is to invoke the spring bean container and creating the respective objects for the mentioned classes in the AppConfig class and second job is receiving the dynamic URL request coming from the U.I and send back the responses after processing the request.
- The DispatcherServlet uses HandlerMapper and HandlerAdapter to achieve the above mentioned goal.
- The HandlerMapper contain the information of all the handler methods present in the classes annotated with @Controller annotation.
- After this all the respective handler details are sent back to the DispatcherServlet, using this information the DispatcherServlet talks to the HandlerAdapter and forwards the request to the respective Controller layer of the Application.
- The HandlerAdapter acts as a connecting bridge between DispatcherServlet and Controller layers of the application.
- Once the data is received on the Controller layer, it is then forwarded to other layers of the application to process the request.
- Once the data is processed and the response is ready to be sent back to the client from the controller layer it again takes the help of HandlerAdapter to reach the DispatcherServlet.
- Then the DispatcherServlet takes the help of the view resolver to get the information of the view technology file that needs to be rendered on the browser based on the request received and processed.
- After the view technology file details is received then the DispatcherServlet will give back the response to the client.

All the stereotyped annotations serves the basic purpose of creating a bean for the spring bean object inside the container once it is scanned by spring.

It also notifies what kind of content is present inside the particular class.

1.@Component - Supposed to be added in the entity classes and helper classes of the application.

2.@Repository - It is an specialized version of @Component supposed to annotated over a class having database logic.

3.@Service - It is an specialized version of @Component supposed to be annotated over a class having business logic of the application.

4.@Controller - It is an specialized version of @Component supposed to be annotated over the controller classes having the end-points for the URL requests and serves as the entry and exit point of the application.

@Autowired :-

- The @Autowired annotation in Spring Framework is used for automatic dependency injection.
- It tells Spring to automatically wire up dependencies for a bean by matching them with other beans available in the Spring application context.
- You can use @Autowired on fields, constructors, or setter methods of a Spring-managed bean to indicate that Spring should automatically resolve and inject dependencies into those points.