**Day 8: 7 – Aug 2024 – Web Application**

Spring Framework : Spring is an open source light weighted layer or onion based architecture framework. Which provide lot of modules which help to improve all types of application. Spring modules

1. Spring core
2. Spring context
3. Spring MVC or web

Spring MVC internally follow MVC architecture framework. Spring MVC is known as Model centric framework. (java bean, service class, dao class, resource class etc)

1. Spring DAO (using JDBC)
2. Spring ORM -🡪 help to integrate with existing orm tool like Hibernate or JPA.
3. Spring cloud
4. Spring security
5. Spring boot
6. Spring micro service
7. Spring testing

etc

Spring core and Context

IOC : Inversion of control

IOC is a programming design pattern. IOC is a concept. According to IOC in place creating any resource(object creation, database connection, file handling, server, security etc) depending on the application requirement explicitly allow to create those resource to container or engine. If container create it will create properly and maintain properly. As developer or programmer we need to pull it from a container, use it and leave it. Container responsible to maintain those resources.

DI : Dependency Injection : DI is an implementation of an IOC. Using DI we can pull the resource from container. Container inject the resource depending upon our requirements.

Types of DI

1. Constructor base DI
2. Setter base DI

To achieve DI using constructor base or setter base we can do configuration using

1. Xml base configuration
2. Annotation base configuration

XML base DI

POJO : Plain Old Java Object . The class not to extends or implements any pre defined class.

That class configuration we can do using xml or using annotation.

ORM tools : JPA or Hibernate

Web Container : it is a part of web server. Web container only create the object of those classes if class is type of servlet or jsp file.

@WebServlet()

class MyServlet extends HttpServlet (GenericServlet or Servlet interface) {

}

In web.xml file

Build tools: build tools is responsible to compile the program, run the program, creating jar or war or ear file, downloading dependencies in the form jar file, providing common project structure, which help to create documentation.

Maven project we can create using eclipse ide or using command prompt.

Mvn clean

Mvn compile

Mvn test

Mvn package

Mvn test

pom.xml : project object model : this file hold project configuration details to deploy or run maven goal.

Maven :

Gradle : xml less

mvn --version

creating maven project using command prompt

mvn archetype:generate

hit enter key twice

Define value for property 'groupId': myproject

Define value for property 'artifactId': simplemavenproject

Define value for property 'version' 1.0-SNAPSHOT: : 1.0

Define value for property 'package' myproject: : com

Yes

Move inside a project folder then run few maven goal.

mvn validate verify pom.xml file

mvn clean it remove old build files

mvn compile it compile all classes from current project and those file keep in target folder.

mvn clean remove target file

mvn test test case execute

mvn package inside target folder it will generate jar or war file

mvn install in project become part of local repository.

mvn clean package

mvn clean install

Spring core and spring context

Spring IOC and DI

We can do DI using XML configuration as well as annotation base.

BeanFactory interface : this interface provide set of methods which help to pull the object or resource from a container.

Auto wired : Spring Framework container do the DI for primitive property implicitly. If our class contains any complex property which can hold more than one value we need to do the DI explicitly using ref attribute part property or constructor-arg tag. Using auto wired features we can do the DI for complex property implicitly rather than explicitly.

byType : that type bean declaration must be present in xml file. Then it will do the DI automatically. In this type we need only one bean declaration of that type.

byName : id name in bean tag and reference name in class must be match.

IOC and DI using annotation base

Annotation

@Component : This annotation we write on pojo or java bean class.

This is class level generic annotation.

@Autowired annotation on complex property level ie which hold more than one value.

@Component annotation by default doesn’t enable. We need to enable using

1. Xml file
2. Using java class with few more annotation.

ApplicationContext : this is a interface which internally extends BeanFactory interface. BeanFactory interface provide only OIC and DI. But ApplicationContext provide IOC and DI as well as other features.

@Value annotation : this annotation we can use on property value to set default value.

@Configuration : this annotation we use on class level. That class contains all configuration details.

@ComponentScan : this annotation we use on class level. To enable few annotation like @Component

Improve model layer :

JavaBean, Service class, Dao, Resource class etc

DataSource : DataSource provide the database connection in secure manner. Before spring boot to achieve data source we were depending upon application server their we are/were configuring database connection and that connection we are/were searching using JNDI look.

Same features Spring framework provide using spring jdbc or dao module.

Spring framework provided pre defined class ie *DriverManagerDataSource*

*Which provide data source features.*

*@Repository : This annotation we write on Dao layer. Dao layer specific annotation*