**API End Points and Communication**

**Day 2: 25 Feb. 25**

**IOC: Inversion of control**

IOC is a design pattern. According to IOC, in place of creating or maintaining any resource like object creation, database connection, security, connection pooling, thread management etc explicitly allow to create by container. If container create it maintain properly. Base upon our requirement you need to pull it from container user it and leave it. The life of the resources taken care by container. Spring framework container is a part of light weighted jar file which is responsible to create the object of normal java bean class.

**IOC is a concept like object**.

**Web container**: part of tomcat server : web container take care the life of servlet and jsp not normal classes.

**EJB container**: part of application server : EJB container create the object of those classes if the class is type of EJB.

**Docker container**: it maintains docker images.

**Spring container**

**DI**: Dependency injection: DI is an implementation of IOC. DI only help us to pull the resource from container.

Types of DI.

1. Constructor base DI
2. Setter base DI

To achieve both the type of DI we need to do configuration

1. Using XML file
2. Using annotation

**IOC and DI using XML Configuration**

We need to add two dependencies ie core and context.

In Spring Framework normal JavaBean class is known as POJO class.

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

**BeanFactory** is one of the interface provided by spring framework which contains set of method which help to pull the resources from container.

Singleton design pattern : if need only one object of that particular class we need to use singleton design pattern.

If we want to achieve DI using setter base in JavaBean class or POJO class we need setter methods mandatory.

**Auto wired :** Auto wired is one the great features of spring framework.

Spring framework do the DI for primitive property implicitly. If property is complex property or user defined object we need to do DI using property ref or constructor – arg ref attribute explicitly. With help of auto wired features we can achieve DI for complex property or user defined object implicitly rather than explicitly.

byType : spring container search bean tag definition of that type. If present it will inject automatically. In byType we need only one bean definition of that type. If more than one bean definition we will get the error.

byName : if more than bean definition of that type present then we need to use byName. In typeName option reference name or object name and id name must be mach.

**Annotation base DI**

1. @Component : This annotation we write on class level. The class is a type of POJO or JavaBean class. it is equal to

<bean class=”com.Address”></bean>

By default id is class name using camel naming rules. If class contains one word then it must be in lower case. If class contains more than one word then 2nd word onward first letter upper case.

If class name is Address : then id is address.

If class name is AddressDetails then id is addressDetails.

1. @Autowired : this annotation we need to use on object level or complex property level.
2. @Scope : this annotation we need to use on class level. By default scope is singleton