**Day 20:**

**Framework**

Framework provided lot classes and interface which internally connected to each other to perform a specific task. If we develop any application using any framework 70 to 80% task taken care framework. But framework is not a final product it is template or protocol. Which help develop the project simplest way. Framework internally provide the implementation of design pattern.

Angular framework

Python with Django framework

.net framework

Struts framework

JSF Framework

Spring framework

Hibernate framework

Express js framework

EJB improve model layer.

Struts : Struts is an open source web framework for java application develop using java technologies and part of Apache company. Struts internally follow MVC architecture. It provided lot of api to improve view layer, controller layer and model layer. It internally follows front controller design patter.

Struts is known as **controller centric framework**.

JSF : Java Server Faces : JSF is part oracle. JSF also follow MVC design pattern. It follow front controller design pattern. JSF is known as View Centric framework.

JSF Vs Angular / react js / Vue JS

Hibernate framework : Hibernate is base upon ORM (Object relation Mapping).

JPA / Hibernate : it is mainly use to improve DAO layer.

Spring framework: Spring is light weighted open source layer architecture framework.

Spring framework provided lot of modules to improve all type of application or layer.

Spring modules

1. Spring core
2. Spring context
3. Spring DAO (spring jdbc)
4. Spring ORM : (JPA, Hibernate or iBaties etc)
5. Spring MVC : Spring MVC internally follow MVC architecture and it is also known as model centric framework.
6. Spring rest (using this module we can create spring rest full web service)
7. Spring security
8. Spring JPA Data
9. Spring boot
10. Spring cloud
11. Spring micro service
12. Spring AOP (Aspect oriented programming)
13. Spring with Apache Kafka
14. Spring Reactive MVC

IOC and DI

Using XML configuration

Using annotation base

**Spring core and Context**

IOC: Inversion of control, IOC is one the programming design pattern. According to IOC in place of creating or maintaining any resources like object creation, database connection, file handling, security explicitly allow to create and maintain by container. If container create or maintain it maintain properly. If we create may be we maintain or not. As a programmer or developer we need to pull the resource from a container base upon requirement and use it and leave it. IOC is a concept.

Container: run time environment.

DI: Dependency Injection,

DI is an implementation of IOC.

We can achieve DI using

1. Constructor base
2. Setter base

In Spring framework we can do DI and IOC using XML Configuration as well as annotation base.

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

Using constructor base DI we need to do full dependencies it may be empty or parameter. if we use parameterized DI using constructor it must order matter.

Setter base DI we can achieve partial DI and order not mandatory to match.

Auto wired : spring framework by default do the DI for primitive property like int, float, char, double,String etc. if property contains complex property means user defined object we need to use **ref** attribute inside property or constructor-arg and do the di explicitly. Using auto wired features we can do the DI for complex property implicitly rather than explicitly using ref attribute.

1. byType : if we use byType autowired features it automatically search that type bean definition in xml file. If present it will do DI for that type. byType we need only one that type bean definition. If more than one we get the error.
2. byName : in byName auto wired id name and reference name must be match.