**Spring**

1. Spring framework is famous due to dependency injection.
2. Injecting object into your application called dependency injection
3. **Maven:**

**Maven** is a build automation tool used primarily for Java projects.**Maven** addresses two aspects of building software: first, it describes how software is built, and second, it describes its dependencies.

1. Dependency means library
2. When you creating maven project following thins you have to know about
   1. **Pom.xml** : where we mention all the jar file dependency
   2. **Group id :** define your package **com.tcs**
   3. **Artefact id:** define your project name **web**
   4. **Package :** combination of group id and atifact id : **com.tcs.web**
3. Mvn repository : for all the jar file dependency
4. Repository **local and remote** first it search in local than find in remote repository
5. For android we use **gradle(**we can use for java aswell**)** inplace of **maven**
6. **Bean:**

The objects that form the backbone of your application and that are managed by the **Spring** IoC container are called **beans**. A **bean** is an object that is instantiated, assembled, and otherwise managed by a **Spring** IoC container.

1. **Inversion of Control (IOC)** :

Spring container uses dependency injection (DI) to manage the objects

1 Bean Factory : for small application use

2 Application Context : for enterprise or big application use this

1. We can achieve dependency injection by three way

Xml based

Annotation based :@Component

Java based

1. Bean property : property tag in sping.xml

<bean id=*"tyre"* class=*"com.tcs.SpringByNavin.Tyre"*>

<property name=*"brand"* value=*"MRF"*></property>

</bean>

1. Contructor injection
2. Annotation based configuration

Tags @Bean @configuration @autowired

**MVC:**

Bcakbone of spring mvc

In **Spring MVC** all incoming requests go through a single **servlet**. This **servlet** - **DispatcherServlet** - is the front controller. Front controller is a typical design pattern in the web applications development. ... The task of the **DispatcherServlet** is to send request to the specific **Spring MVC** controller

Pom.xml (**project object model**):

1. For mvc project first create maven with archtype as webapp
2. Configure spring-web.xml and web.xml(mapping of dispatcher servlet)
3. Annotation for mvn @Controller and @RequestMapping

**IOC(Inversion of control):**

Approach of outsourcing the construction and management of object

**Spring container:**

Create and manage object : **inversion of control**

Inject object dependency: (**dependency injection**)

Both is diff and DI is the part of IOC

**Spring container configuration:**

* XML based
* Annotation based
* Java based

Spring container is generally known as **application context**

* classPathXmlApplicationContext
* annotatioConfigApplicationContext

spring steps

create bean

create container

retrieve bean from container

**Spring Bean :** its simply java code ,when object are created by container spring refer as spring bean.

**Dependency injection:**

Dependency means helper objects

**Tyes of in injection:**

1. **constructor injection**
2. **setter injection  
   auto wiring in annotation section**

setter injection: inject dependency by setter method called setter injection

form-tag: used for data binding used to automatically setting /retrieving data from java/object or bean

@Controller

@RequestMapping:

is one of the most common annotation used in Spring Web applications. This annotation maps HTTP requests to handler methods of MVC

@RequestParam : used to bind the data in variable

@ModelAttribute : used to bind all the form attribute to the object

 @RequestMapping(value="/save",method = RequestMethod.POST)

Spring Core::

1) Thus, IOC makes the code loosely coupled. In such case, there is no need to modify the code if our logic is moved to new environment.

In Spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.

2 )

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

* to instantiate the application class
* to configure the object
* to assemble the dependencies between the objects

There are two types of IoC containers. They are:

1. **BeanFactory**
2. **ApplicationContext**

Spring conta

iner :

Primary function

1 create and manage object (inversion of control)

2 inject object dependencies (Dependency injection)

Spring configuration

XML based

Annotation based

Java based

Bean scopes:

* 1. Singleton :: Default bean scope is singleton shared same memory for all the object scope=”singleton” it is cached in memory Create on instance of the bean.
  2. Prototype :: create new bean request for each container request.

Bean life cycle::

Container started 🡪 Bean instantiated 🡪 Dependencies injected -> internal spring processing -> custom inti method -> bean Is ready for use ->destroy method

We can use some business logic setting up db, socket during bean initialization.

Init configuration :: init-method=””;

Destroy method :: destroy-method=””;

**Dependency injection:**

Constructor

Setter

Any method

Field injection

**Qualifier:**

 @**Qualifier** annotation along with @Autowired to remove the confusion by specifying which exact bean will be wired.

@Scope is used to set the bean scope singleton and prototype.

Bean life cycle

@PostConstruct is used for bean initilazation

@PreDestrop is used after destroy of bean

Ways of configuring spring container

1. XML based
2. XML component scan
3. Java configuration class

Java configuration class

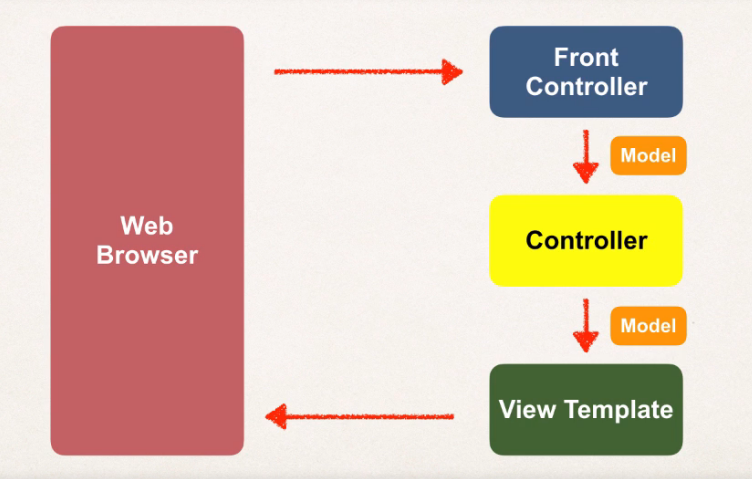
Create one class use @Configuration

@CompoentScan(“com.udemy.annotation”)

And use application context as

AnnotationConfigApplicationContext context = **new** AnnotationConfigApplicationContext(JavaConfig.**class**);

MVC : model view controller



Spring mvc strt with front controller known as dispatcherServlet.

Setup the web.xml and spring configuration file.

Mvc Annotations::

1. The **@Controller** annotation is used to mark any java class as a controller class.
2. The **@RequestMapping** annotation is used to map the web request "/userRegistration.htm" to the UserController class.
3. @RequestParam : bind the html form data

**Html form tag:**

ModelAttribute

**Hibernate::**

1 Hibernate is framework which is used to save persisting java object into database.

Code --🡪 Hibernate 🡪 database

2 Hibernate provide object relational mapping

3 Hibernate 5.2 require java 8.

4 configure the hibernate configuration file

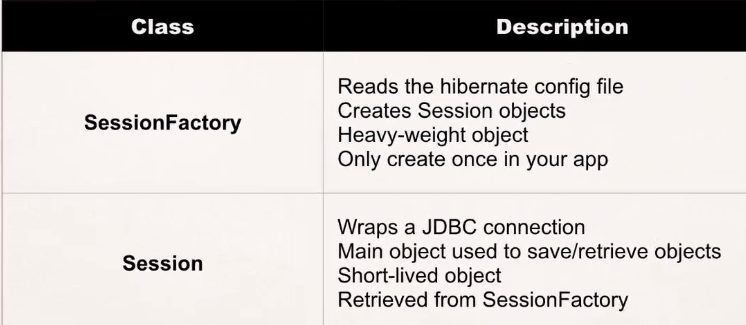
5 Entity class : java class that is mapped to the database table

6 @Entity, @Table,@Column,@Id,@

7 JPA : java persistence annotation

JPS is standard specification hibernate team recommend use JPA

8 key player of hibernate



9 Default configuration file name of hibernate:

**Hibernate.cfg.xml**

**10** SessionFactory sf = **new** Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Student.**class**)

.buildSessionFactory();

Session session = sf.getCurrentSession();

Student st = **new** Student("vipin","rathore");

session.beginTransaction();

session.save(st);

session.getTransaction().commit();

11 **HQL ::** hibernate query language

12 for auto increment key

@GeneratedValue(strategy=GenerationType.***AUTO***)

13 uni directional relationship

@OneToOne(cascade=CascadeType.***ALL***)

@JoinColumn(name="instructor\_detail\_id")

14 @tranasactional tag automatically do the hibernate stuf begin and commit and all

15 @ Repository

