Spring Udemy Notes

why spring:

popular framework

simper and lightweight alternative to J2EE

large number of helper classes. make things easier

get rid of EJB, former bad reputation.

New for Spring 5:

java 8 or higher

Upgrade Spring MVC to use new versions of Servlet API 4.0

Add new reactive programming framework: spring webFlux

Core Container: Beans:Core:SpEL:Context:

Infrastructure: AOP: Aspect Oriented Programming

Data Access Layer: JDBC ORM Transactions OXM JMS:Java message service

Web layer: Servlet, websocket, web ,porlet

Test layer: unit, integration, mock

Inversion of Control(IoC)

The approach of outsourcing the construction and management of objects.

Software Engineering Best Practice: code to an interface

Spring Container:

Primary functions:

1. Create and manager objects(Inversion of Control)

2. Inject object's dependencies(Dependency Injection)

Configuring Spring Container:

1. XML configuration file(legacy)

2. Java Annotations(modern)

3. Java Source Code(modern)

Spring Development Process

1. Configure ur spring Beans: treat the beans as java objects

2. Create a Spring Container: generically known as ApplicationContext

3. Retrieve Beans from Spring Container.

Dependency Injection:

Injection types two common: 1. Constructor Injection. 2. Setter Injection !!// treat dependency as a helper

Setter Injection:

1. Create setter method(s) in class for injections. 2. Configure the dependency injection in Srping config file

Injecting Literal Values:

1. Create setter methods in class for injections.: create private fields, create setter methods.

2. Configure the injection in Spring config file.

Singleton:

1.Spring Container creates only one instance of the bean, by default.

2.It's cached in memory.

3.All requests for bean, will return a SHARED reference to the SAME bean.

Bean Prototype Scope: new bean object for each request: different address different object

IF not mentioned,would be considered by default: singleton scope: only one instance, different requests only share the bean.

Bean lifecycle:

container started--> Bean Instantiated--> Dependencies Injected --> Internal Spring Processing --> your custom Init Method --> Bean is ready for use --

-- container is shutdown --> your custom destroy method --> stop

Bean lifecycle methods/ hooks:

1. you can add custom code during bean initializaion, calling custom business logic methods, setting up handles to resources(db, sockets, file etc).

2. you can add custom code during bean destruction, calling custom business logic method, clean up handles to resources(db, sockets, files ect).

Init: method configuration:set up bean initialization(init-method="any method name").

Sestroy:method configuration: set up bean destroy method(destroy-method="any method name").

Development process:

1. Define your methods for init and destroy.

2. Configure the method names on Spring config file