3 approaches of spring app development

a) using Xml driven cfgs

b) Using annotation driven cfgs

c) Using 100% Code driven cfgs

c) Using 100% Code driven cfgs

=> In this approach, either we should avoid xml driven cfgs or we should minimize xml driven cfgs

=> This approach of spring programming is the base for spring boot framework based

applications development.

=>This approach helps us to develop spring boot apps every easily

=> Spring boot= spring ++

=> Spring boot App = spring App - xml cfgs (avoid or mimimize) + Autoconfiguration

(100% code driven cfgs based spring app development)

Thumb rules to remember while developing 100% code driven cfgs based Spring App development

==:

=> based on the jar files added to the

=========

spring boot app following operations takes place

-> makes pre-defined classes as the spring beans

-> Injects one spring bean with another spring bean -> adds the additional jar files

-> Gives Embedded DB s/w

-> Gives Embedded servers and etc..

a) Give inputs and instructions to IOC container using @Configuration class as alternate to spring bean cfg file (xml file)

AppConfig.java

```
@Configuration
public class AppConfig{
}
```

b) Configure user-defined classes as the spring beans using @Component annotation make these classes

as scannable classes for IOC container by specifying their package names in @ComponentScan annotation (alternate to <context:component-scan > tag of spring bean cfg file) in the @Configuration class

```
package com.nt.sbeans;
@Component("wf")
public class WeekDayFinder{
@Configuration
@ComponetScan (basePackages="com.nt.sbeans' public class AppConfig{
}
}
```

c) Configure pre-defined classes as spring beans using @Bean methods of @Configuration class

AppConfig.java

@Configuration

@ComponentScan (basePackages="com.nt.sbeans")

public class AppConfig{

@Bean(name="ldate")

}

public LocalDate createLDate(){

return LocalDate.now();

This method will be called by IOC container

on @Configuration class obj automatically and makes the method returned object as the spring bean having bean id (ldate)

d) use @Autowired annotation in target spring bean class at various of ur choice to inject the the dependent spring bean class obj to target spring bean class obj's HAS-A property

package com.nt.sbeans;

@Component("wf") public class WeekDayFinder{

@Autowired //Field Injection private LocalDate date;

=>@Autowired on top of Filed (HAS-Property) performs Filed Injection

=>@Autowired on top of Setter method (HAS-Property) performs Setter Injection

=>@Autowired on top of parameterized constructor

performs constructor Injection

=>@Autowired on top of arbitrary method performs arbitrary method Injection

}

f) create IOC container using AnnotationConfigApplicationContext giving @Configuration class as input class to provide inputs and instructions to IOC container

AnnotationConfigApplicationContext ctx=

new AnnotationConfigApplicationContext(AppConfig.class);

Example app on 100% Code driven cfgs approach using filed Injection to inject LocalDate class obj to

WeekFinder class obj and to display proper message in the b.method for week days and for week end days?

IOCProj05-Week DayFinder-100pCodeDrivenCfgs

//WeekDayFinder.java (Target spring bean class)

>

JRE System Library [JavaSE-21]

#src

#com.nt.config

> AppConfig.java

#com.nt.main

> DepedencyInjectionTest.java

com.nt.sbeans

> WeekDayFinder.java

Referenced Libraries

package com.nt.sbeans;

import java.time.LocalDate;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Component;

@Component("wdf")

public class WeekDayFinder {

@Autowired //Field Injection private LocalDate date;

public WeekDayFinder() {

System.out.println("WeekDayFinder:: 0-param constructor");

>

spring-beans-6.2.3.jar - C:\Users\Nataraz\Downloads

>

spring-context-6.2.3.jar - C:\Users\Nataraz\Downloads

>

spring-expression-6.2.3.jar - C:\Users\Nataraz\Downloads

>

<

spring-core-6.2.3.jar - C:\Users\Nataraz\Downloads spring-aop-6.2.3.jar - C:\Users\Nataraz\Downloads spring-jcl-6.2.3.jar - C:\Users\Nataraz\Downloads spring-context-support-6.2.3.jar - C:\Users\Nataraz\Downloads

}

//b.method

//AppConfig.java (Configuration class)

package com.nt.config;

import java.time.LocalDate;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan; import org.springframework.context.annotation.Configuration;

@Configuration // @Component++

@ComponentScan (base Packages = "com.nt.sbeans")

public class AppConfig {

public AppConfig() {

}

System.out.println("AppConfig:: O-param constructor");

//pre-defined class as the spring bean

@Bean(name="ldate")

public LocalDate createLDate() {

```
System.out.println("AppConfig.createLDate()");
return LocalDate.now();
}
}
}
public String showMessage(String user) {
// get current week day number
int number=date.getDayOfWeek().getValue();
// generate the message
if(number>=1 && number<=5)
return " Work Hard to build Stroing IT Career:" + user;
else
return "Take a Break and Enjoy ur week end:"+user;
}
//DependencyInjection Test.java (mainclass)
package com.nt.main;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import com.nt.config.AppConfig;
import com.nt.sbeans.WeekDayFinder;
public class DepedencyInjection Test {
public static void main(String[] args) { Create IOC container
AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext(AppConfig.class);
//get target Spring bean class obj ref Object obj=ctx.getBean("wdf");
//type casting
WeekDayFinder finder=(WeekDayFinder)obj; //invoke the b.methods
String msg=finder.showMessage("raja");
System.out.println("Result is::"+msg);
//close the IOC container
ctx.close();
```

Problems Servers Terminal Data Source Explorer Properties Console X <terminated> Depedency Injection Test [Java Application] D:\Software\eclipse\plugins\org.eclipse.justj.openjdk.hotspot

```
AppConfig:: 0-param constructor
WeekDayFinder:: 0-param constructor AppConfig.createLDate()
Result is:: Take a Break and Enjoy ur week end:raja
}
```

How to get all the bean ids that are created and managed by IOC container?

Ans)

use ctx.getBeanDefinitation Names() method on the IOC container object

```
System.out.println("All the Bean ids are ::" + Arrays.toString(ctx.getBeanDefinitionNames()));
```

```
All the Bean ids are ::

[org.springframework.context.annotation.internalConfigurationAnnotation Processor,
org.springframework.context.annotation.internalAutowiredAnnotation Processor,

org.springframework.context.event.internalEventListener Processor,
org.springframework.context.event.internalEventListenerFactory,

appConfig, wdf, ldate]
```

our cfgs based spring beans

Pre-defined classes spring beans