First Application development in spring boot for Dependency Injection

==========

Dependency Injection

=> Making the underlying container/server/framework/Runtime env.. dynamically assingning/injecting dependent class object to taget class object is called dependency Injection. we need to use @Autowired annotation for this.

at filer level (Field Injection) (best)

at setter method level (setter Injection)

======

from spring prospective

=> It is all about making the IOC container /Spring Container

assigning the dependent spring bean class obj to target spring bean class object

at parameterized constructor (constructor Injection) (fastest)

at arbitary method (arbitary method Injection)

=> The class that uses the services of other class is called Target class (In spring/spring boot it is called target spring bean) => The helper class to target class is callled dependent class.

WishMessageGenerator (target class)

SeasonFinder

(Target class)

Flipkart

Flipkart

(target class) (target class)

(In spring/spring boot it is called dependent spring bean)

LocalDateTime (dependent class) LocalDate (dependent class)

(WishMessageGenerator uses LocalDateTime to generate the wish messages based on the current hour of day) (SeasonFinder uses LocalDate class to display the current month) and season name

DTDC (dependent calss) GooglePe (dependent class)

(Flipkart uses DTDC as courier service to deliver the products)

(Flipkart uses Google Pay for Online UPI Payment)

example

target spring bean class- user-defined class

@Component("sf") //sf is bean id

com.nt.sbeans.Season Finder

@Autowired

|--->private LocalDate date; //field (HAS-A property)

b.method

--> public String showSeason(){

Here we can logic by using

}

the injected LocalDate class obj to display current season name based on the current month

Dependent Spring bean (pre-defined class)

package com.nt; @SpringBootApplication public class

FirstApplication{

@Bean("dt") //dt is bean id public LocalDate createLD(){ return LocalDate.now();

ps

v main(String args[]){

When the IOC container creates /manages the spring bean it takes the bean id as the object

name

if we give the above code to IOC container/spring container the following operations takes places

(a) IOC container loads both target and dependent classes to create objs as the spring beans (based on @Component, @Bean methods)

(b) IOC container perorms Depndency Injection (field Injection) to assign dependent class obj to target class object based on @Autowired

(c) IOC container keeps the Spring bean class objs in the internal cache of the IOC container for the resusability

Season

Finden obj(sf) @Autoivred date

LocalDate class abji dt)

(b)

sf

dt

Internal cache

(a)

(a)

SeasonFinder class obj ref LocalDate class obj ref

Thumb rule to remember while working with spring boot Application (standalone app)

==========

====

========

==> configure user-defined classes as spring beans using stereo type annotations like @Component,@Service and etc... ==> configure pre-defined classes as spring beans using @Bean methods of @SpringBootApplication class

if those classes are not coming as spring beans through AutoConfiguration. note:: u can give inputs/instructions to AutoConfiguration process using application.properties/yml file

static method

==> In main(-) method of @SpringBootApplication class, call SpringApplication.run(-,-) to get Access to SpringContainer /IOC container and use that container to get

ur choice spring bean class obj by passing its bean id and to invoke b.methods on it. Inside main(-) method

//get Access to Spring container

ApplicationContext ctx-SpringApplication.run(-,-);

// get access to spring bean calss obj

Message

WishMessageGenrator generator=ctx.getbean("wmg", WishGenerator.class);

keys(bean ids)

....

values (spring bean class obj refs)

=> application.proeprties /yml comes automatically in every Spring boot Project creation in the src/main/resources folder

and we need not to configure this file separately by using @PropertySource becoz it will be configured automatically as

part of the Spring Boot App's boot strap process

geneator.method1(); geneator.method2();

Invoking

required spring bean type

bean id

the b.methods

★ 4158

Spring Tools 4 (aka Spring Tool Suite 4) 4.22.0.RELEASE

Spring Tools 4 is the next generation of Spring Boot tooling for your favorite coding enrivonment. Largely rebuilt from scratch, it provides world-class support... more info by VMware, EPL

spring Spring IDE Cloud Spring Tool Suite STS

Installs: 2.84M (25,254 last month)

(or)

Installed

Example App

=====

step1) keep the following software setup ready

JEE

Eclipse 2020+ version with STS plugin

(2024-06)

note:: STS (spring Tool suite) must be installed manually

By adding STS plugin to Eclispe IDE we can use

STS (Spring Tool Suite) IDE features in Eclipse IDE itself

Spring Tools 3 Add-On for Spring Tools 4 3.9.22.RELEASE

Spring Tools 3 Add-On for Spring Tools 4 End of life: This Spring Tools 3 Add-On for Spring Tools 4 is no longer

maintained or updated. The final and last release... more info by VMware, EPL

J2EE spring Spring IDE Cloud jee

Plugin in the Eclipse IDE

help menu---> eclipse market place ----> search for STS ----> select spring tool suite 3.9 --->next --->next -->accept terms

and conditions ---> ... ...

(restart eclipse IDE as it demands)

step2) create Spring boot starter Project giving the following details

File ---> new ---> spring starter Project --->

Service URL

Name

4.0

https://start.spring.io (url where actual spring boot project will be created) BootProj01-DependencyInjection (Project name)

Use default location Location

E:\Worskpaces\Spring\NTSPBMS616-Boot\BootProj01-Depender Browse

Maven (build tool) Packaging: Jar

Language:

Java

com.nit (company name) BootProj01-DependencyInjection (Project name) 0.0.1-SNAPSHOT (version)

Type:

Java Version:

17

java version

Group

Artifact

Version

Description Package

Working sets

Example App

com.nt

Add project to working sets

(optional)

***jar for standalone apps war for web applications Version SNAPSHOT :: Code is under**

developerment

version RELEASE :: Final code which is ready to release or already released (main class)

(Root package name where @SpringBootApplication anntation based class will come)

New...

-->next ----> select no starters (default are sufficient) ---> finish.

Eclipse IDE

Eclipse SDK (only for

standalone apps)

Eclipse JEE (best) (for all types of the apps)

1.Eclipse

supplied Plugins

=> use help menu ---> Install new software option

to install these plugins [eg:: GUI Builder plugin, Lombok API plugin and etc..]

(For Latest Eclipse JEE IDE downloading

2. Third party plugins for Eclipse IDE

=>use help menu ---> Eclipse market place option to install these plugins

[eg: STS Plugin, Gradle Buildship plugin, SonarQube plugin and etc..]

https://www.eclipse.org/downloads/packages/release/2023-12/r)

Always place main class in the default root package like com.nt

and remaining packages as the sub packages of the root package (com.nt)

So the @ComponnentScan annotation of @SpringBootApplication annotation

can scan and get stereo type annotations tidasseffectively from root pkg and its sub pkys

The every Spring boot Project that is created will

be inherited from the Parent boot project of the Maven Central repository by getting <parent> </parent>in pom.xml file

....

step3) observe the Generated directory structure of the Project

✓ BootProj01-DependencyInjection [boot]

> Spring Elements

src/main/java To place pkgs with source code (real code)

>com.nt

#src/main/resources

application.properties

| application.properties file to give various

src/test/java | To place pkgs with unit testing source code

>com.nt

> JRE System Library [JavaSE-17]

>

Maven Dependencies

✓

src

✓

main

java

> com

resources

> test

› target

WHELP.md

mvnw

src/main/java src/test/java src/main/resources

application.properties

maven generated

output files will be saved here.

mvnw.cmd

pom.xml (To given instructions maven tool)

<parent>

<groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.5</version> <relativePath/> repository --> </parent>

instructions to spring boot (overriding the default settings)

These are given quick access navigation points

pom.xml :: Project object model

application.properties/application.yml files are part of spring boot Project echo system so they will be recognized by IOC container automatically i.e we do not need any separate configuration

step4)

Develop the source (target and dependent classes as spring beans)

This called Maven inheritence

=>All the jars and plugins belonging to parent project will be inherited child project..

=> Based on this only we get all jar files from the spring boot starter parent project to our project

//SeasonFinder.java (target class)

```
package com.nt.sbeans;
import java.time.LocalDate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
@Component("sf")
public class SeasonFinder {
//bean property @Autowired
private LocalDate ldt;
```

makes the class

as spring bean

```
//b.method
public String findSeason() {
}
//get current month
int month=ldt.getMonthValue();
//show seasos name
if(month>=3 && month<=6) return "Summer Season";
else if(month>=7 && month<=10)
return "Rainy Season";
else
return "Winter Season";
```

Makes the IOC container

to seach for spring bean whose type is LocalDate and gets that object and injects that object ldt property.

=> java.time.LocalDate is jdk supplied pre-defined class..

So the spring's IOC container does not give this class as

the Spring bean through AutoConfiguration.. So use @Bean method of @SpringBootApplication class (main class)

to make java.time.LocalDate class as the spring bean

=>Always place the package of main class where @SpringBootApplication annotation is placed as one level up compare to tother packages so

all the classes with stereo type annotations will be scanned automatically

to make them as spring beans becoz of @ComponentScan annotation of @SpringBootApplication.

```
}
//main class
package com.nt;
(This class main class cum @Configuration class)
import java.time.LocalDate;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ConfigurableApplicationContext; import
org.springframework.context.annotation.Bean;
import com.nt.sbeans.Season Finder;
@SpringBootApplication
public class BootProj01DependencyInjectionApplication {
@Bean(name="ld")
```

public LocalDate createDate() { return LocalDate.now();

public static void main(String[] args) { //get IOC container

internally usses

note:: In spring boot Apps, we do not take separate Configuration classes becoz the main class having @SpringBootApplication itself acts as the configuration class

AnnotationConfigApplicationContext

class to create the IOC container by taking

current given class as the confuguration class

ApplicationContext ctx-SpringApplication.run(BootProj01DependencyInjectionApplication.class, args);

//get Target spring bean class obj

SeasonFinder finder=ctx.getBean("sf",SeasonFinder.class);

//invoke the b.method

String seasonName=finder.findSeason();

System.out.println("Season Name::"+seasonName);

//close the IOC container

((ConfigurableApplicationContext) ctx).close();

}

X Delete

step5)

run the application

Right click on main class surce file ---> run as ---> Java App

› BootProj01Deper Copy Qualified Name

com.nt.sbeans

> SeasonFinder.jav

#src/main/resources

application.properti

src/test/java

>com.nt

Paste

Ctrl+V Delete

ate createDate() { alDate.now();

Remove from Context

Ctrl+Alt+Shift+Down

Build Path

>

ervers Data Source Explore

Source

Alt+Shift+S>

cylnjectionApplication [Java A

JRE System Library [Jav

Refactor

Alt+Shift+T >

Maven Dependencies

src

Import...

✓ main

Export...

✓

java

1 Run on Server

> com

Refresh

F5

2 Java Application

✓ resources

Close Project

application.pr

m2 3 Maven build

> test

target WHELP.md

mvnw mvnw.cmd

Mpom.xml

Close Unrelated Project

m2 4 Maven build...

References

>

m2 5 Maven clean

Declarations

>

m2

6 Maven generate-sourc

Mark as Deployable

m2

7 Maven install

m2

8 Maven test

2 Coverage As

▸ Run As

Run Configurations...

org.springframework.boot.SpringApplication

Class that can be used to bootstrap and launch a Spring application from a Java main method. By default class will perform the following steps to bootstrap your application: (start)

Create an appropriate ApplicationContext instance (depending on your classpath) (IOC container) Register a CommandLine PropertySource to expose command line arguments as Spring properties ✔Refresh the application context, loading all singleton beans

•

`Trigger any CommandLine Runner beans`

In most circumstances the static run (Class, String []) method can be called directly from your main method to

bootstrap your application:

La Piuviens

TIY Servers

Teman Dala Source Lapiorer rroperties Consure privyless

<terminated> BootProj01-DependencyInjection - BootProj01 DependencyInjectionApplication [Spring Boot App] C:\Program File 2024-04-05T21:51:51.927+05:30 INFO 6128 --- [BootProj01-DependencyInjection] [

2024-04-05T21:51:51.935+05:30 INFO 6128 --- - [BootProj01-DependencyInjection] [SeasonFinder:: 0-param constructor

BootProj01 DependencyInjectionApplication.createDate()

2024-04-05T21:51:52.671+05:30 INFO 6128 --- - [BootProj01-DependencyInjection] [SeasonFinder.findSeason()

output is Summer Season

Assignment1:: Develop the app to generate the wishmessage based on the current hour of the day
Assignment2:: Develop the app to generate the message based on the current weekday or weekend day

✓ ApplicationContext

> > ApplicationContextAssertProvider<C>

• ConfigurableApplicationContext

✓

AbstractApplicationContext

✓ AbstractRefreshableApplicationContext

> ^ AbstractRefreshableConfigApplicationContext

GenericApplicationContext

> AnnotationConfigApplicationContext

|---> enjoy ur week end

I

|----> happy working hours

=> Every Spring Boot Project created by the Programmer will become child project to "spring-boot-starter-parent" project which is collected from Maven central repository.. So from this "spring-boot-starter-parent" project lots of jar files will be inherited automatically our project ..