Q) i have 10 spring beans, how can we make only 5 spring beans participating in the pre-instantiation?

Ans1) Take all the 10 spring beans as the singleton scope spring beans but enable @Lazy(true) on 5 spring beans, but make sure that these 5 spring beans are not dependent to other 5 singleton scope spring beans

(or)

Ans2) Take 5 spring beans as singleton scope spring beans and other spring beans as the prototype spring beans but make sure that the prototype scope spring bean are not dependent to singleton scope spring beans

StereoType Annotations

=> The spring supplied multiple annotations having similar functionality are called Stereotype annotations i.e all these annotations perform similar operations with minor changes

note:: Boys dressing is called stereo type dressing, girls dressing is not called stereo type dresssing

a) @Component ----> makes the java class as the spring bean

b) @Service ----> makes the java class as the spring bean cum service class c) @Repository ---> makes the java class as the spring bean cum DAO class/ Repository class/ Persistence class

d) @Controller ---> makes the java class as the spring bean cum web controller class having the ability to take the requests from the clients

e) @Configuration ----> makes the java class as the spring bean cum

and etc..

@ Component

Configuration class

=> The java class that contains the b.logic in methods is called service class

=> b.logic means dealing with calculations, analyzations, validations, sorting, filtering and etc,,

=>@Service spring bean automatically gets TxMgmt support => The java class that contains the persistence logic

(CURD operations logics) is called DAO class /Repository class

DAO class ---> Data Access Object class

@Repository based DAO class makes the IOC container to convert underlyingly generated jdbc exceptions to spring/spring boot exceptions TxMgmt (Transaction Management)

=> It is the process of executing certain logics by applying do every thing or nothing principle

eg:: transfer money operation needs two sub operations

a) withdraw amount from source account b) deposite the amount into dest account

needs to execute by enabling do everything or nothing

@Service

@Repository

@Controller

@Configuration

In Real projects, we develop the project as the Layered App i.e we keep different logics in different java classes and we make them participating in the communication.. In this situation,we take the support of these

special stereo type annotation to make the java classes as the spriing beans cum special classes of layered app.

Client app

(presentation logics/ UI Logics)

(@Controller) controller class Д (monitoring logics)

(@Service) ->service class Д (b.logic)

(@Repository) DAO class गु (persistence logic -- jdbc code)

DB s/w

others like

@RestController @RestControllerAdvice

note:: In any spring project of 100% code configuration we pass inputs to IOC container using @Configuration class

@Service/@Repository/@Controller/@Cofiguration = @Component++

(These annotations make the java classes as the spring beans cum special classes)

note: All stereotype annotation classes will be scanned by @ComponentScan annotation of the @Configuration class to recognize and use them as normal or special spring beans

properties file

==========

=>The text file that maintains the entries in the form of key-value pairs is called properties file =>This file is also called ResourceBundle file

=> This file can have any extension, but the recomanded extension is .properties

=> The keys can have multiple nodes separated with "." symbols

Can u place multiple stereo type annotaitons for a java class configuration as spring bean? Ans) No, Not possible, if we provide different bean ids

yes, possible, if we provide same bean ids

@Component("wmg") @Service("wmg") public class Wish MessageGenerator{

}

...

//valid code

@Component("wmg1") @Service("wmg") public class Wish MessageGenerator{ //invalid code

}

In a Layered App of spring can change the roles of stereotype annotations?

Ans) Yes, but not recommended becoz the readability of the code and maintainance of the code will be distrubed

Taking @Service for DAO class is bad practice Taking @Repository for Service class is bad practice

Info.properties

per.id=101

-per.name=raja

per.addrs=hyd

per.mobileno=988989998

note:: we can write comments in properties file using # symbol

keys

values

note:: The properties file that is configured on top of one spring bean class using @PropertySource annotation can be used in all the spring bean' classes of the project

=>To configure the properties with the spring App using @PropertySource Annotation

@PropertySource(value="<name and location of the properties file>")

On the top of

spring bean class

=> we can use @Value Annotation to read the values from the properties file and to inject them to spring bean Properties

gives raja

In spring bean class @Value("${per.name}") private String pname;

gives hyd

@Value("${per.addrs}")

private String paddrs;

${<key>} ---> place holder -- represents

the value that is going to come

from the properties file or from other places

(It is like placing? symbol in the SQL Query)

=>@Value is the multipurpose annotation, which can be used for different operations

directly

a) To Inject simple values to primitive/String bean properties directly

@Value("raja")

eg:: private String pname;

b) To inject the values collected from the properties file to primitive /String spring bean properties

@Value(" ${per.name}")

eg: private String pname;

injects the value collected from

the properties file by submitting the key

c) To inject the system property values to primtive/String bean properties

@Value("${os.name}")

private String os_name;

Value("${os.version}")

private String os_ver;

sion

os.name, os.ver and etc.. are the fixed system properties

d) To inject env.. variable values to primitive/String bean properties

@Value("${Path}") private Stirng pathData;

Path, ClassPath, JAVA_HOME and etc.. are called env.. variables The key in the ${...} is called Place holder that represents name of the value that going to be injected to spring bean properties

What is difference b/w @Value and @Autowired annotations?

Ans) To Inject one spring bean class obj to another spring bean class obj's HAS-A property take the support of @Autowired (For Injecting spring bean)

In target spring bean

@Autowired

private IEngine engg; //HAS-A property

(Spring bean to Spring bean Injection)

To inject simple values collected from different places (like properites file,system properies and etc..)

to the primitive /String spring bean properties take the support of @Value annotation

@Value(" ${per.name}")

eg: private String pname;

Injecting simple values to SpringBean properties

Working with Properties file

===

=>@PropertySource is the class level annotation to configure properties file with Spring Application

=> @Value is the field level annotation to inject the values collected from the properties file, system properties,

env variables with Spring Bean's primitive and String properties

=> Comments in properties file can be writtern using # symbol

Different System properties In Java environment

1.1. Runtime Environment Properties

`java.home`

`java.library. path`

`java.class.path`

`java.ext.dirs`

`java.version`

`java.runtime.version`

`JRE home directory, e.g., "C:\Program Files\Java\jdk1.7.0_09\jre".`

JRE library search path for search native libraries. It is usually but not necessarily taken

from the environment variable PATH.

JRE classpath e.g., '.' (dot - used for current working directory).

JRE extension library path(s), e.g, "C:\Program

`Files\Java\jdk1.7.0_09\jre\lib\ext; C:\Windows\Sun\Java\lib\ext".`

`JDK version, e.g., 1.7.0_09.`

JRE version, e.g. 1.7.0_09-b05.

1.2. File System Properties

file.separator

symbol for file directory separator such as 'd:\test\test.java'. The default is '\' for windows or '/' for Unix/Mac.

symbol for separating path entries, e.g., in PATH or CLASSPATH. The default is ';' for windows or ':' for Unix/Mac.

path.separator

line.separator

symbol for end-of-line (or new line). The default is "\r\n" for windows or "\n" for Unix/Mac OS X.

1.3. User Properties

user.name

the user's name.

user.home

the user's home directory.

user.dir

the user's current working directory.

note:: In real projects, we use properties file(s) to inject technical values to Spring /Spring boot application like jdbc properties (jdbc driver class name, url, db username, db password and etc..)

//PersonInfo.java

Example App

IOCProj09-Working_With_Properties_File

src/main/java

com.nt.client

<

> PropertiesFileTest.java

com.nt.commons

#com.nt.config

>

AppConfig.java com.nt.sbeans

> PersonInfo.java

>

src/test/java

JRE System Library [JavaSE-17]

>

Maven Dependencies

>

src

> target

pom.xml

Info.properties

```
# Personal Info
per.name=raja per.addrs=hyd
per.id=1001
```

note:: u can take any thing as the keys and vlaues

```
package com.nt.sbeans;
import org.springframework.beans.factory.annotation.Value; import
org.springframework.context.annotation.PropertySource; import
org.springframework.stereotype.Component;
@Component("pInfo")
@PropertySource(value = "com/nt/commons/Info.properties") public class PersonInfo {
//injecting the values of properties file to Spring bean properties @Value("${per.id}")
private Integer pid;
@Value("${per.name}")
private String pname;
@Value("${per.addrs}")
private String addrs;
// injecting the direct values to spring bean properties @Value("9898989899")
private Long mobileNo;
// injecting the System property values
@Value("${os.name}")
private String os_name; @Value("${os.version}")
private String os_ver;
// Injecting env.. variable values @Value("${Path}")
private String path_data;
//toString()
```

while taking primitive properties in spring bean class or java class or java bean class

it is recomanded to take wrapper class type becoz they hold "null" when no value

is given .which represent no value.. where as simple data type style properties hold 0,0.0 as the default value.. which represent some value

```
@Override
public String toString() {
return "PersonInfo [pid=" + pid + ", pname=" + pname + ", addrs=" + addrs + ", mobileNo=" + mobileNo
+", os_name=" + os_name + ", os_ver=" + os_ver + ", path_data=" + path_data + "]";
}
```

//AppConfig.java

```
package com.nt.config;
```

```
import org.springframework.context.annotation.ComponentScan; import
org.springframework.context.annotation.Configuration;
@ComponentScan(base Packages = "com.nt")
@Configuration
public class AppConfig {
}
```

In old version of spring, we used to get exception when we pass the wrong key in the @Value annotation

=>In new version of spring, we are not getting exception when we pass the wrong key in the @Value annotation rather we can see the injection place holder value as it is

```
//Client App
package com.nt.client;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import com.nt.config.AppConfig;
import com.nt.sbeans.PersonInfo;
public class PropertiesFileTest {
}
}
public static void main(String[] args) {
//create IOC container
AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext(AppConfig.class); //get Spring bean class obj ref
PersonInfo info=ctx.getBean("pInfo", PersonInfo.class);
System.out.println(info);
```

The IOC container internally maintains few ready made Spring Bean class objs like Enviromnent object.. This object gets info from

the configured properties files, system properties, env.. variable values and sends the spring bean class obj properties based on the keys placed in the place holders of the @Value Annotation

to

This object can be injected to other sporing bean using @Autowired annotation, can be accessed from the client App using ctx.getEnvironment() method

In Client App

```
//get access to Environment object
Environment env=ctx.getEnvironment();
System.out.println("os.name ::"+env.getProperty("os.name"));
```

Info.properties

```
# Personal Info
per.id=1001
per.name=raja
per.addrs=hyd
```

System properties

...

env variable values

...

In another Spring bean class

==========

package com.nt.sbeans;

=========

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.core.env.Environment;

import org.springframework.stereotype.Component;

@Component("pInfo1")

public class PersonInfo1 {

@Autowired

private Environment env;

public void showData() {

System.out.println("os.name::"+env.getProperty("os.name"));

System.out.println(" per.id key value::"+env.getProperty("per.id"));

}

}

IOC container Environment obj

Spring Bean class obj

@value("<key>}")

... spring bean properties

✓ PropertyResolver

✓ Environment

✓

ConfigurableEnvironment

✓

AbstractEnvironment

StandardEnvironment

=>Environment is not the obj of Environment(1) .. It is the obj of a class that

implements Enviroment(I). In spring 6.x version the class name if Envoronment obj is "StandardEnvironmnet"

Q) How to configure multiple spring bean cfg files in our spring app?

Ans) @PropertySource({"com/nt/commons/Info1.properties","com/nt/commons/Info.properties"})

info1.properties

per.id=1010

per.addrs=hyd

info.properties

per.id=1001

per.name=raja per.billamt=56788.89

Q) if multiple properties files that are configured with spring app are having same keys and with different values then the @Value annotation having that key injects which value as the final value?

Ans) The lastly configured properties file (in @PropertySource Annotation) key maintained value will be picked up for injecting the value to the spring bean property

In spring bean class

@Value("${per.id}")

private int pid:

gets 1001 as the value

from info.properties (2nd file)

In AppConfig.java

info1.properties

per.id=1010

per.addrs=hyd

@PropertySource({"com/nt/commons/Info1.properties","com/nt/commons/Info.properties"})

1

2

=> In xml driven configurations, we can use <context:PropertyPlaceHolder location="com/nt/commons/Info.properties"/>

info.properties

per.id=1001 per.name=raja

per.billamt=56788.89