Date : 09/09/2020

Spring Boot 7AM

Mr. RAGHU

--------------------------------------------

\*) basePackage for Component Scanning:-

Spring Container will search/find/detect classes from basePackage only.

Even if we follow all rules and wrote classes outside of basePackage then

those classes are not detected by Spring container, finally no object is created.

-> basePackage means one common package name for all classes.

-> Spring container will search for classes in given package and its sub-package also.

-> \*\* By default Starter class package behaves as basePackage by Spring boot.

--Example---------------------------------------

#1

Startre class package : in.nareshit.raghu

classes packages selected/not

A in.nareshit not

B in.nareshit.raghu selected

C in.nareshit.test not

D in.nareshit.raghu.test selected

#2. We can even override default basePackage rule by providing

@ComponentScan("packageName") [or]

@ComponentScan(

{

"pack1",

"pack2",

"pack3",

...

}

)

-> Need to write at starter class, then starter class package is no longer considered.

Starter class package: in.nareshit.raghu

@ComponentScan("com.app")

classes packages selected/not

A com.app selected

B com.app.test selected

C app.com.one not

D in.nareshit.raghu not

Ex#3

@ComponentScan({"com.app","in.nareshit.raghu","app.com"})

classes packages selected/not

A com.app selected

B com.app.test selected

C app.com.one selected

D in.nareshit.raghu selected

Q) Who will provide @ComponentScan in Spring Boot?

A) @SpringBootApplication annotation provides @ComponentScan by default with

starter class/main class package name.

-----------------------------------------------------

Starter class package Name : in.nareshit.raghu

classes package name ??

MessageARunner com.app.test NO

MessageBRunner in.nareshit.raghu YES

MessageCRunner in.nareshit.raghu.runner YES

MessageDRunner in.nareshit NO

\*) If you want to execute all above runners then provide below line at starter class:

@ComponentScan({"in.nareshit","com.app"})

--code--

a) Starter class

package in.nareshit.raghu;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

//@ComponentScan("in.nareshit")

@ComponentScan({"in.nareshit","com.app"})

public class SpringBoot2ClrExThreeApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBoot2ClrExThreeApplication.class, args);

}

}

b) Runner classes

package com.app.test;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class MessageARunner implements CommandLineRunner {

@Override

public void run(String... args) throws Exception {

System.out.println("SAMPLE-A");

}

}

--

package in.nareshit.raghu;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class MessageBRunner implements CommandLineRunner {

@Override

public void run(String... args) throws Exception {

System.out.println("SAMPLE-B");

}

}

--------

package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class MessageCRunner implements CommandLineRunner {

@Override

public void run(String... args) throws Exception {

System.out.println("SAMPLE-C");

}

}

-------

package in.nareshit;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class MessageDRunner implements CommandLineRunner {

@Override

public void run(String... args) throws Exception {

System.out.println("SAMPLE-D");

}

}

==========================================================================

Q) Can we define multiple runners in one application?

A) YES. They are executed in spring boot selected order.

Q) Can we define our own order of execution for Runners?

A) YES. Using @Order annotation.

Lowest value is high priority in execution.

...-9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9...

--Example code--

Runners:

package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;

import org.springframework.core.annotation.Order;

import org.springframework.stereotype.Component;

@Component

@Order(-85)

public class MessageRunnerA implements CommandLineRunner {

@Override

public void run(String... args) throws Exception {

System.out.println("FROM-A");

}

}

--

package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;

import org.springframework.core.annotation.Order;

import org.springframework.stereotype.Component;

@Component

@Order(20)

public class MessageRunnerB implements CommandLineRunner {

@Override

public void run(String... args) throws Exception {

System.out.println("FROM-B");

}

}

--

package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;

import org.springframework.core.annotation.Order;

import org.springframework.stereotype.Component;

@Component

@Order(105)

public class MessageRunnerC implements CommandLineRunner {

@Override

public void run(String... args) throws Exception {

System.out.println("FROM-C");

}

}

Execution order:\_\_\_\_\_\_\_\_\_

\*) Note: If two runners are having same order then execution order depends on their

loading (into Container, it may look like name based order-but not correct).

\*) If a runner has no order, gets executed last.

========================================================================

**Working with Properties file**

========================================================================

\*) Properties means Storing data in key-value format.

ex:

driver=Oracle

url=jdbc:orcl

\*) keys are case-sensitive (eid,EID are different)

driver=Oracle

Driver=Oracle

-> Both are different

\*) If same key is provided multiple time with different value

then last combinaton is taken into code.

driver=Oracle

driver=MySQL

driver=H2

-> Finally selected is driver=H2

\*) In general, Spring f/w uses @PropertySource annotation to load one .properties file

into Spring container. A memory created 'Environment' [key-val store]

\*) Coming to Spring boot, by default 'application.properties' file is loaded into

Spring container, with internal code like

@PropertySource("classpath:application.properties")

Note\*) classpath means src/main/resources

---Read key-val data in code---

a. @Value annotation [Single key reading]

b. \*\* @ConfigurationProperties [Multiple keys reading]

-------------------------------

a. @Value annotation: This annotation is used to read one key data into one variable

Syntax: @Value("${key}") //Supports even type conversion.

--Example--

#1. Create one starter project

Name: SpringBoot2PropsExOne

#2. application.properties

# My Project details

app-id=101

app-id=550

title=NIT

version=3.2

#3. Runner class code

package in.nareshit.raghu.runner;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class MyPropsRunner implements CommandLineRunner {

//1 key data --> 1 variable

@Value("${app-id}")

private int id;

@Value("${title}")

private String ttl;

@Value("${version}")

private double ver;

@Override

public void run(String... args) throws Exception {

System.out.println(this);

//System.out.println(id+"-"+ttl+"-"+ver);

}

@Override

public String toString() {

return "MyPropsRunner [id=" + id + ", ttl=" + ttl + ", ver=" + ver + "]";

}

}

\*) Duplicates are not recomanded by Spring boot (but allowed), if we define duplicate

keys with different data then last combination is selected.

\*) If given key in @Value is not present in properties file then Spring boot throws

IllegalArgumentException: Could not resolve placeholder 'app-id' in value "${app-id}"

\*) If we provide wrong data type data, then NumberFormatException: For input string: "ABCDEF"

will occure.

ex: app-id=ABCDEF

\*) Syntax is @Value("${key}"), if we do not follow syntax properly then given one is

considered as vaue not as key. Try to assign same as value to variable

(which may give NumberFormatException)

@Value("{title}")

Spring ttl; // here ttl="{title}"

@Value("$app-id")

int id ; //NumberFormatException

---------------------------------------------------