Date : 10/09/2020

Spring Boot 7AM

Mr. RAGHU

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

**Working with Properties File**

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

=> To read multiple key-val into code (Variables) use @ConfigurationProperties

=> @Value supports reading a at time one key- into one variable only.

=> \*\* To work with @ConfigurationProperties, we need to follow a prefix

(common word for all keys).

@ConfigurationProperties with

a. Primitives

b. List/Set/Array [1D collection]

c. Map/Properties [2D Collection]

d. Association Mapping (HAS-A with class)

========= a. Working with Primitives: =====================

#i. Define key-val in application.properties file using one common prefix

#ii. Provide common prefix at class level using @ConfigurationProperties(prefix="")

#iii. No.of keys exist in properties file must match with no.of variables in class

#iv. key name and variable name must match.

#v. Also provide set/get methods.

\*) If we do not follow above rules no error/exception is raised. But value is not provided

for variable.

--Example#1---

#1. Create on Spring Starter Project

Name : SpringBoot2ConfigPropsExOne

#2. Write one class with variables and set/get method

also apply @ConfigurationProperties(prefix="")

package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.stereotype.Component;

@Component

@ConfigurationProperties(prefix = "my.app")

public class MyDataRunner implements CommandLineRunner {

private Integer id;

private String code;

private Double version;

private String model;

@Override

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

System.out.println("FROM RUNNER");

System.out.println(this);

}

//set/get methods

> Source > Generat getters and setters > Select All > Generate

//toString

> Source > Generat toString > Generate

}

#3. provide key-val pairs with common prefix in application.properties

---application.properties--

my.app.id=100

my.app.code=ABCD

my.app.version=9.9

my.app.model=NIT

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

\*) Note: (it is optional)

Inside class at @ConfigurationProperties level, you can see one yellow color underline

that is a warning, if we place mouse over that it shows one option:

Add spring-boot-configuration-processor to pom.xml

If we choose this option, then our variables with prefix are considered as keys in

properties file.

in pom.xml below content is added

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

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

--Example#1 with configuration-processor---

#1. Create on Spring Starter Project

Name : SpringBoot2ConfigPropsExTwo

> Next > Search with 'configuration processor'

> Select checkbox 'Spring boot configuration processor'

> Finish

#2. Class with @ConfigProps

package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.stereotype.Component;

@Component

@ConfigurationProperties(prefix = "product.nit")

public class ProductDataRunner implements CommandLineRunner {

private Integer mid;

private String service;

private String grade;

@Override

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

System.out.println(this);

}

public Integer getMid() {

return mid;

}

public void setMid(Integer mid) {

this.mid = mid;

}

public String getService() {

return service;

}

public void setService(String service) {

this.service = service;

}

public String getGrade() {

return grade;

}

public void setGrade(String grade) {

this.grade = grade;

}

@Override

public String toString() {

return "ProductDataRunner [mid=" + mid + ", service=" + service + ", grade=" + grade + "]";

}

}

#3. ----application.properties--------

product.nit.grade=ABC

product.nit.mid=99

product.nit.service=XYZ

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

============ **b. List/Set/Array [1D collection]** ==============

\*) one variable that holds multiple values

In this case we need to provide key-val pair using below syntax in properties file

prefix.variable[index]=value

--Example#2 with configuration-processor---

#1. Create on Spring Starter Project

Name : SpringBoot2ConfigPropsEx1DColl

> Next > Search with 'configuration processor'

> Select checkbox 'Spring boot configuration processor'

> Finish

#2. Runner class

package in.nareshit.raghu.runner;

import java.util.Arrays;

import java.util.List;

import java.util.Set;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.stereotype.Component;

@Component

@ConfigurationProperties(prefix="my.app")

public class ProductDataRunner implements CommandLineRunner {

private List<String> colors;

private Set<String> codes;

private String[] models;

@Override

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

System.out.println(this);

}

//set/get methods

> Source > Generat getters and setters > Select All > Generate

//toString

> Source > Generat toString > Generate

}

#3.--- application.properties----

my.app.colors[0]=RED

my.app.colors[1]=GREEN

my.app.colors[2]=BLUE

my.app.codes[0]=A

my.app.codes[1]=B

my.app.codes[2]=C

my.app.models[0]=M1

my.app.models[1]=M2

my.app.models[2]=M3

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

\*) We can pass data using one short format ie

prefix.variable=val1,val2,val3...

In this case index concept is not going to work. Priority is

given for short format only.

----application.properties---

my.app.colors=R,G,B

my.app.colors[0]=RED

my.app.colors[1]=GREEN

my.app.colors[2]=BLUE

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

Q) What is the actual/main difference between List and Set?

A) Duplicates are not stored in Set<> Collection, where List accept.

:Student wrote exam:

marks --> List - 90,90,90

subjects --> Set - ENG, MAT, SCI

Task:

\*) Create variable of type boolean(false/true) and char types provide data.

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

**c. Map/Properties [2D Collection]**

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

--working with Map<k,v>---

#1. Create on Spring Starter Project

Name : SpringBoot2ConfigPropsMapEx

> Next > Search with 'configuration processor'

> Select checkbox 'Spring boot configuration processor'

> Finish

#2. Runner class

package in.nareshit.raghu.runner;

import java.util.Map;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.stereotype.Component;

@Component

@ConfigurationProperties(prefix="my.app")

public class MyPropsMapRunner implements CommandLineRunner {

private Map<String,Integer> edata;

@Override

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

System.out.println(this);

}

public Map<String, Integer> getEdata() {

return edata;

}

public void setEdata(Map<String, Integer> edata) {

this.edata = edata;

}

@Override

public String toString() {

return "MyPropsMapRunner [edata=" + edata + "]";

}

}

#3. application.properties

my.app.edata.ENG=95

my.app.edata.MAT=98

my.app.edata.SCI=99

my.app.edata.SCI=66

my.app.edata.sci=55