Date : 17/09/2020

Spring Boot 7AM

Mr. RAGHU

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

\*)In Spring boot, application execution entrypoint is: main(String[])

\*) we can pass inputs at runtime to main() method. Those are called as CLA(Command Line Arguments)

Example:

package com.app;

public class Test {

public static void main(String[] args) {

System.out.println(args[0]);

System.out.println("done");

}

}

#compile: javac Test.java

#run : java Test SAM ABC 123

\* Here [SAM,ABC,123] String array is given as input to main() method.

Q) How can we pass main() method inputs using STS/Eclipse?

A) Write Main class and print args

> Right click on code > Run As > Run Configuration

> Click on Arguments tab

> Enter values inside 'program argumnents' > Enter values > Apply > Run

Ex:2

package com.app;

import java.util.Arrays;

public class Test {

public static void main(String[] args) {

System.out.println(args[0]);

System.out.println(Arrays.asList(args));

System.out.println("done");

}

}

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

\*) In Spring Boot, starter class [main() method class], will call all runners one by one.

While executing runner(ex: CommandLineRunner) main() method also pass the input ie args.

ie args are given from main() method to every runner.

--Ex--

#1. Create one Starter Project

name : SpringBoot2ClaTest

#2. Runner class

package in.nareshit.raghu.runner;

import java.util.Arrays;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class ClaTestRunner implements CommandLineRunner {

@Override

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

System.out.println("DONE");

System.out.println(Arrays.asList(args));

}

}

#3. Execution

> Right click on main() class > Run As > Run Configuration

> Click on Arguments tab

> Goto Program arguments > Enter values > Apply > Run

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

**Arguments in Spring Boot**

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

main() method gets arguments while executing application which are called as CLA

(CommandLineArguments). In case of Spring boot these CLA args are two types

a. Option Args : (--key=val) : These behaves like input to application

b. NonOptionArgs: (command/word) : These behaves like task/work to application.

\*) Here, both option and non-option args are stored in one array only.

--Ex--

#1. Create one Starter Project

Name: SpringBoot2ClaTestTwo

#2. Runner

package in.nareshit.raghu.runner;

import java.util.Arrays;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class ClaTestRunner implements CommandLineRunner {

@Override

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

System.out.println("DONE");

System.out.println(Arrays.asList(args));

}

}

#3. Execution: pass below format values as Command Line Args

--title=NIT dbDump exportEmp --code=ABC

> Apply > Run

Q) Whay option-args and what is the use?

A) Generally properties file stores data in key-val format, that is given to Spring container

and we can read using @Value and @ConfigProps.

Here, we can pass same key-val at runtime using Option-Args(High Priority)

same value can be read using @Value and @ConfigProps.

--Ex#3--

#1. Create new Spring Boot application

#2. Write one Runner class

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 ClaTestRunner implements CommandLineRunner {

@Value("${title}")

private String title;

@Override

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

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

System.out.println(title);

}

}

#3. application.properties

title=LOCAL-TITLE

#4. Run As > Run Config > Arguments > enter value like

--title=NIT

> Apply > Run

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Spring Boot - Profiles**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*) Types of RuntimeEnvironments in IT

DevEnv -- Developer Machine for coding

QAEnv -- Tester Machine for Testing

UATEnv -- Client-Test(User Acceptance Test) Before purchase product

Production- Realtime Application deployed server.

\*) If we write code using Spring boot and this application is moved from one

RuntimeEnvironment(one System to another System) to another, then code never gets modified

only changes comes in properties files (not even all, few which are required).

\*) Profiles concept is used to handle Environment Specific key-val loading.

1. Profiles using properties file

2. Profiles using YAML File

1. Profiles using properties file:

\*) Create properties file for every environment (1 propertis file - 1 RuntimeEnv)

and select properties file at runtime(Using Option Args).

Naming rule:

application-[profileName].properties

examples:

application.properties [No profile Name given=> default Profile]

application-qa.properties [profile: qa]

application-uat.properties [profile: uat]

application-prod.properties [profile: prod]

\*)Activate profile at runtime using below key

--spring.profiles.active=<profileName>

example:

--spring.profiles.active=qa

==Example#=

#1. create spring boot application

Name : Springboot2ProfilesFirstApp

#2. application.properties

dbname=Oracle-Dev

#3. create new file : application-qa.properties

dbname=mySQL-QA

#4. Runner class

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 ProfileTestRunner implements CommandLineRunner {

@Value("${dbname}")

private String dbn;

@Override

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

System.out.println("FROM CLR : " + dbn);

}

}

#5. Execution: Run As > Run Configuration > Arguments

--spring.profiles.active=qa

> Apply > Run

\*) If we did not provide any profile name, then default profile[application.properties]

is selected.