Runners in spring Boot Application

========

========

=>Runner is a spring bean of spring boot App that contains one time excution logics in its run(-) method. (callback method)

=>We do not call the run(-) of the Runner class manually ..It will be called by Spring Boot automatically as part of Application's startup prcoess that is started by SpringApplication.run(-) method Right after pre-instataition of singleton scope beans and completing the neccessary dependency Injections.

=>The Spring Boot App recognizes the spring bean class as the Runner class by seeing its XxxRunner (1) implementation.

of

Two types Runners

=>A good programmer of spring boot Application does not keep any user-defined code in main(-) method of main class.. rather he prefers placing testing code

code

or method invocation in the Runner class

==> A good programmer just places SpringApplication.run(-,-) method call

in main(-) of @SBA class.. All the logics of client app / testing the app will be placed in the run(-,-) method of the Runner class

note:: The method that is called by underlying server or container or JVM automatically for certain event is called callback method. eg:: servlet comp life cycle methods,

Spring boot Runners run method and etc..

a) CommandLineRunner (Spring bean class implementing org.springframework.boot.CommandLineRunner(1)) b) ApplicationRunner (Spring bean class implementing org.springframework.boot.ApplicationRunner(1))

with

What is the difference b/w placing static block in spring bean class and working Runner class?

static block of spring bean class static{...}

==========

(a) It is java feature, can be used in both spring and spring boot Apps

(b) Executes automatically when the IOC container Loads the spring bean calss

not

(c) Does allow to pass data/args from outside

(d) Non-static data can not be accessed in this static block

^

Runner class of Spring boot App

==========

(a) It is a feature of spring boot

Can be used only in spring boot app

(b) Executes automatically as part of Spring boot Application_startup activity performed by the

SpringApplication.run(-) method (mostly after pre-instantitions and injections)

(c) Allows to pass data/args to run(-) method

(command line args can be passoed)

(d) Allows us to access both static and non-static data

(e) usefull to place class level one-time executing logics.

(f) Static block does not support Exception Propagration

(e) Useful to place spring boot Application level

one time executing logics

(eg:: Executing the DB script during the Appliation startup to create db tablesin Db s/w (or) The client App logics)

(f) run(-) method Runner class supports the Exception Propagations

note:: if we place multiple Runner classes the logics of all runner classes execute only for

1 time during the application startup prcoess.

ApplicationRunner

==========

org.springframework.boot.ApplicationRunner(1)

In one Spring boot App/Project, we can place 0 or more runner classes

Modifier and Type boolean

command line args

|--->public void run(ApplicationArguments args) throws Exception

101 --name-raja --addrs=hyd

Non option args

=>This

run method allow to get cmd line args as catergorized args Method and Description

like option args(with name,value) and non option args(only value) containsOption(String name)

optional args (name=value is given)

`List<String>`

`Set<String>`

`List<String>`

String[]

CommnadLineRunner

Return whether the set of option arguments parsed from the arguments contains an option with the given nam
getNonOptionArgs()

Return the collection of non-option arguments parsed.

`getOptionNames()`

Return the names of all option arguments.

`getOptionValues (String name)`

Return the collection of values associated with the arguments option having the given name.

getSourceArgs()

Return the raw unprocessed arguments that were passed to the application.

==========

org.springframework.boot.CommandLineRunner(1)

|---->void run(String... args) throws Exception

=>Command line args will come directly..

No Categorization for cmd line args. (i.e we get all args as the non-option args only i.e no provision to cateroize them)

What is the difference b/w Command Line Runner and ApplicationRunner?

=>Both are one time execution classes /spring bean having run(-) method ..But the way they get cmd line args to the run (-) is different

.form of

CommandLineRunner gets the given cmd line args in the form of String[] where as ApplicationRunner gets the given cmd line args in the ApplcationArguments object where we can categorize the cmnd line into optional and non-optional args

//CommandLineRunner example

=======

TestRunner.java

```
package com.nt.runner;
import java.util.Arrays;
import org.springframework.boot.Command Line Runner; import org.springframework.stereotype.Component;
@Component
public class TestRunner implements CommandLineRunner {
```

Run as ---> run cfgs arguments tab

---->

Program arguments:

101 --name-raja --addrs=hyd

non option option option

arg

arg1 arg2

```
}
}
@Override
public void run(String... args) throws Exception {
System.out.println("Runner to Test...."+Arrays.toString(args));
```

ApplicationRunner example

package com.nt.runner;

import org.springframework.boot.ApplicationArguments; import org.springframework.boot.Application Runner; import org.springframework.stereotype.Component;

@Component

public class Test1Runner implements Application Runner {

@Override

}

output:: Runner to Test....[101, --name-raja, --addrs=hyd]

public void run(ApplicationArguments args) throws Exception { System.out.println("non option args values::"+args.getNonOptionArgs());

System.out.println("Option arg names and values::");

|--->101

for(String name:args.getOptionNames()) {

}

System.out.println(name+"-----> "+args.getOptionValues(name));

addrs

hyd

name

raja

args[0]

args[1]

args[2]

Variables...

Run as ---> run cfgs

---->

arguments ta b

Program arguments:

101-name-raja --addrs-hyd

non option option option arg1 arg2

arg

oulput: non option args values::[101] Option arg names and values::

addrs----->[hyd]

name-----> [raja]

note: In real layered Applications we inject service Impl class object/controller obj to Runner class and we invoke methods on that object in the run(-) of the Runner class

note:: ComandLineRunner, Application Runner are

two

two indenpendent interfaces. (There is no relationship b/w these interfaces)

note:: Mostly used Runner in the industry is Command Line Runner note:: Both Command Line Runner and

Application Runner interfaces are Functional Interfaces

(The java interface that is having single abstract method)

=> A good programmer of spring boot Application

does not keep any user-defined code in main(-) method

of main class.. rather he prefers placing testing code

code

or method invocation in the Runner class

==> A good programmer just places SpringApplication.run(-,-) method call

in main(-) of @SBA class.. All the logics of client app/testing the app will be

placed in the run(-,-) method of the Runner class

Example app

==========

```
//main class
package com.nt;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class BootlocProj03RealtimeDiMiniProjectApplication {
public static void main(String[] args) {
SpringApplication.run(BootlocProj03 RealtimeDiMiniProjectApplication.class, args);
}//main
}//class
//MiniProjectTestRunner.java
package com.nt.runners;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.Command Line Runner;
import org.springframework.stereotype.Component;
import com.nt.controller.StudentOperationsController;
import com.nt.model.Student;
@Component
public class MiniProjectTestRunner implements CommandLineRunner {
@Autowired
private StudentOperationsController controller;
public MiniProjectTestRunner() {
}
@Override
```

}

}

System.out.println("MiniProjectTestRunner::0-param constrcutor");

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

System.out.println("MiniProjectTestRunner.run()");

try {

List<Student> list=controller.processStudentsData("hyd","vizag", "chennai"); list.forEach(st->{

System.out.println(st);

});

}

catch(Exception e) {

e.printStackTrace();

}

Q) Though, we are not adding any starters to Spring boot project, How does it gets multiple jar files

as the maven/gradle dependencies to the Project?

Ans) In every Spring boot Project, One parent spring boot project will be imported from the maven central repo using <parent> tag as show below

Variables...

<parent>

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

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

=>This project is readymade project available in the

maven central repository and will be inherited to every spring boot project giving its common dependencies, properites and plugins to the child project

=> Based on the parent spring boot project's version, lots of things will come to our spring boot project which is nothing but a child project

(a) Basic spring, spring boot jar files and dependent, relavent jar files (b) required plugins

(c) required Project properties

To feel all these things look at pom.xml file of spring boot starter parent .The procedure is

go to mvnrepository.com ---> search for spring boot starter parent ---> select the starter ---> select the version (3.3.0) ---> clink on pom.xml hyperlink

=> In one spring boot application we can place multiple Runners .. if do not want to execute specific Runner then remove @Component on top of that runner (Atleast u can comment that @Component class)

=> By default the multiple Runner will execute in random order

=> To provide our choice execution order to these multiple Runner classes we can use

either @Order(<n>) or Ordered(1) implementation on Runner classes.

New

Old (Legacy)

=>Once @Order is specified having priority value or Ordered (1) is implemented having priority value .the.Runners will execute according to the prirority value with out worrying about Runner type

and their class names

=>High value indicates low priority

=>Low'ificates high priority

=>if same priority value is given then they will go by

Negetive values are

accepted as the priority values

randomorder

note:: The Soloty value is Integer MAXValue (2147483648) (Integer. MAX_VALUE

```
@Component
@Order(10)
public class AltertServiceRunner1 implements CommandLineRunner {
@Override
public void run(String... args) throws Exception {
System.out.println("AltertServiceRunner1.run().. The cmd line args are");
for(String arg:args)
System.out.println(arg);
```

Runner

XRunner

@Order value

10 (IV)

YRunner ZRunner

-20 (1)

1 (II)

ARunner

6 (111)

Runner

@Order value

XRunner (AR) YRunner (CR)

10

10

Executes in Random Order

ZRunner (CR)

10

ARunner (AR)

Runner

10

@Order value

XRunner

YRunner (AR) ZRunner ARunner (CR)

<no value>

10 (ii)

<no value>

1 (i)

Executes in Random Order (iii, iv)

Runner

XRunner(CR)

YRunner(AR) ZRunner(AR) ARunner(CR) BRunner(CR)

CRunner (AR)

DRunner(AR)

on

@Order value

10 (III)

-20 (1)

<no vlaue>

6

6

<no value>

6.

Conclusion priority order

=> Low value to high value of @Order

(iv) Thsese things in random order

(II) These things in Random Order

[if multiple runners having same prority the

then they will be executed in random order)

=> No @Order Runners

(Thee Runners will execute in random order)

We can alsfority value by implementing Ordered (1) as shown below

@Component

public class AltertServiceRunner1 implements CommandLine Runner,Ordered {

```
@Override
public void run(String... args) throws Exception {
for(String arg:args)
System.out.println("AltertServiceRunner1.run().. The cmd line args are");
System.out.println(arg);
}
@Override
public int getOrder() {
return 15;
}
}
r
```

A) if we provide two different priority values using @Oder and Ordered(1) for a Runner class then which will be taken?

```
ed
```

Ans) Order(1) 's getOrder() method retuned value

will be taken as the prirority value

```
@Component
@dder(-20)
public class AltertServiceRunner1 implements Command Line Runner,Ordered {
@Override
public void run(String... args) throws Exception {
System.out.println("AltertServiceRunner1.run().. The cmd line args are");
for(String arg:args)
System.out.println(arg);
}
@Override
public int getOrder() {
return 15;
}
}
```

This runner class

prority value is :: 15

Debugging the Application using Eclipse IDE

=>Debugging is useful to feel application's code execution line by line

=> Using debugging, we can identify the errors, fix the errors

=> Using Debugging, we can feel the data flow

Home >> org.springframework.boot » spring-boot-starter-parent » 3.3.0

License

Tags

Spring Boot Starter Parent » 3.3.0

Parent pom providing dependency and plugin management for applications built with Maven

HomePage

Apache 2.0

spring framework starter

https://spring.io/projects/spring-boot

Date

May 23, 2024

Files

pom (12 KB) View All

Repositories Ranking

Central

#12390 in MvnRepository (See Top Artifacts)

=>Using Debugging, we can understand the code written by others (very useful in project maintainenace)

=>Using Debugging, we can enable monitoring /watching on the variable data, objects data, and expressions Important terminologies in Debugging

a) breakpoint: the point that is placed in certain line number of the code from which the developer can control and feel the

flow of execution (we can place multiple break points in one app/file)

b) step into (F5) :: To get current caller method definition code

c) step over (F6) :: To execute the method call code with out getting into definitation

d) step return (F7): returns the control from current method definitation to caller method

e) jump to break point (f8) :: goes to next break point from the current position

Drop to frame: very useful to repeat the method execution from the beginning either with old values or with new values

note:: Mostly used option in real time to understand the code

flow is not "Run As" option.. it is always "debug as" option

Procedure to perform

==========

step1) keep Mini Project App ready

step2) keep one break point in the first line of main(-) method definitition

step3) Run the App using Debug As option

Use F5, F6, F7 options to feel the flow

Suggetions:: if the method call is pre-defined method call then go for F6

if the method call is user-defined method call then go for F5

if want to execute same method definitation for more times use "drop to frame" option in the middle of the method execution

if code flow has gone to InMemory proxy classes code and u want to come out of it then use "F8" (resume) option to go

to next break point

Spring Boot Core FAQS

=====

=====

Q) what is the difference b/w Spring and Spring boot?

Q) what spring boot formulae w.r.t to spring?

is

Q) What is Auto Configuration in Spring boot?

Q) What is the benefit of Auto Configuration feature?

Q) What is starter in spring boot?

Q) How does starter is different from jar file?

Q) Explain what is brown filed project and what is green field Project?

Q) Explain @Spring BootApplication annotation?

Q) What is difference b/w @Value and @ConfigurationProperties annotation?

Q) What is Layered app?

Q) How does Dependency Injection and Lookup concepts will be used in realtime?

Q)

What is JDBC con pool

Q) How JDBC con pool and DataSource are interrelated?

Q) what DataSource algorithm in Spring boot App?

Q) Explain various popular DataSources

Q) What is the difference b/w properties file and yml file

Q) What are profiles in spring boot/spring app

Q) In many ways we can activate the profiles?

Q) Explain @Profile annotation

Q) Explain runners and their use cases in realtime?

Q) What is fifference CommandLineRunner and ApplicationRunner?

Q) Explain about Spring boot starter parent project?

Q) What is the benefit of making every spring boot project as the child project

of Sprng boot starter parent project?

Q) What is the difference b/w Program Args and VM args in run configurations of the Eclipse IDE ?

Q) What is the difference b/w Non-Option Args and Option Args of Command Line Args ?

Q) Explain different starters that are given spring boot?

Q) How can make certain pre-defined class not participating in AutoConfiguration activity ?

Runner class utilization in the MiniProject

==========

In main class

//Client App

==========:

```
package com.nt;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class BootlocProj12MiniProject RealtimeDiApp {
public static void main(String[] args) {
SpringApplication.run(BootlocProj12MiniProject RealtimeDiApp.class,args);
}//main
}//class
```

//Runner class

```
package com.nt.runners;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Component;
import com.nt.controller.Payroll MgmtOperationsController;
import com.nt.model.Employee;
@Component
public class MiniProjectTestRunner implements CommandLineRunner { @Autowired
private Payroll MgmtOperationsController controller;
@Override
public void run(String... args) throws Exception {
System.out.println("MiniProjectTestRunner.run()");
// read inputs from the enduser
Scanner sc=new Scanner(System.in);
System.out.println("Enter Desg1 ::");
String desg1=sc.next();
System.out.println("Enter Desg2 ::");
```

```
String desg2=sc.next();
System.out.println("Enter Desg3 ::");
String desg3=sc.next();
//invoke the b.methods
List<Employee> list-controller.showEmployeesByDesgs(desg1,desg2,desg3);
//display result
System.out.println("Employees belonging to "+desg1+" "+desg2+" "+desg3+" are ");
list.forEach(emp->{
System.out.println(emp);
});
}
}
```

=> Calling ctx.getBean(-) having bean id to get spring bean class obj ref

comes under Dependency Lookup operation

=> using @Autowired in spring bean classes to assign/inject one spring bean class obj to another spring bean class obj

comes under Dependency Injection

=> By Adding Runner class support to the spring boot projects.. we can completely eleminate Dependency Lookup operations from

the spring boot Projects

=> Runners are introduced from Spring boot i.e they do not work in Spring framework and they work only in Spring boot

framework