Date : 14/12/2020
Spring Boot 9AM
Mr. RAGHU
---

*) Runners in Sping Boot:-
To execute any logic only one time when application is getting started.

=> Runners are used in setup logic/test purpose/batch processing.

*) There are two types of Runner in Boot.
a) CommandLineRunner (I)
b) ApplicationRunner (I)

-------------Eclipse/STS Shortcuts----------------------------
ctrl+shift+T (open type) => Enter full/partial class name
=> double click on matching name

ctrl+O (Overview) => To see all variables/methods in class

ctrl+L (Goto line) => Enter line number (ex: 30) press enter key.

ctrl+shift+O (Imports)

ctrl+F11 (or) Ctrl+Fn+F11 (To run main class)

** Programmer has to define one class that implements CommandLineRunner and override run() method
also add @Component

** Spring Boot Starter class(main class) will find all runner classes and executes/calls them.!

============code=====================================
S#1 Create one Spring Starter Project
> File > New > Spring Starter Project > Enter details

Name : SpringBoot2RunnersExOne
Package : in.nareshit.raghu

> Next > Next > Finish

S#2 Create a class under basepackage

```
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

//ctrl+shift+O
@Component
public class MyRunner implements CommandLineRunner {

        //ctrl+space for code snippets
        @Override
        public void run(String... args) throws Exception {
```

```
                System.out.println("FROM 1ST RUNNER CODE");
        }
}
```

CommandLineRunner + @Value

@Value annotation is used to read data from properties files
ie from one key to one variable.

If we have 20 variables in class then 20 times we should write @Value.
[not a good approch for multiple variables]

--code---

#1 Create Spring Starter Project
Name : SpringBoot2RunnerValueExTwo
Package : in.nareshit.raghu

#2 application.properties

```
my.app.id=10
my.app.code=ABC
my.app.cost=8900
```

#3.Runner class

```
package in.nareshit.raghu.runner;

//ctrl+shift+O
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class DataRunner implements CommandLineRunner {

        @Value("${my.app.id}")
        private Integer id;

        @Value("${my.app.code}")
        private String code;

        @Value("${my.app.cost}")
        private Double cost;

        @Override
        public void run(String... args) throws Exception {
                //on print any reference variable toString
                System.out.println(this);
                //System.out.println(id+"-"+code+"-"+cost);
        }

        //source Menu > Generat toString > ToString()
        public String toString() {
                return "DataRunner [id=" + id + ", code=" + code + ",
cost=" + cost + "]";
        }
```

```
}

#4. Open main class and press : ctrl+F11
================================================================================
                Bulk Loading : @ConfigurationProperties

=>This is also called as 'Bulk Loading' or reading multiple keys
  from properties files to our variables at a time.

--Rules--
a) Define keyname using 'prefix.variableName' in properies file

b) Provide that common prefix at our class level using annotation
   @ConfigurationProperties(prefix="")

c) Define variables in our class and generate set/get methods
   and also toString(optional).

 > Source Menu > Gerenetae Getters and Setters > Select All > Generate

--code--
Name : SpringBoot2RunnerConfigPropsExOne
Package: in.nareshit.raghu


#2 application.properties
my.app.id=99
my.app.code=ABC
my.app.cost=9089

#3. Runner class code
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 DataRunner implements CommandLineRunner {

        private Integer id;
        private String code;
        private Double cost;

        @Override
        public void run(String... args) throws Exception {
                System.out.println(this);
        }

        //Source Menu > generate Getters and Setters
        // > Select All > Generate
        public Integer getId() {
                return id;
```

```
	}
	public void setId(Integer id) {
		this.id = id;
	}

	public String getCode() {
		return code;
	}

	public void setCode(String code) {
		this.code = code;
	}

	public Double getCost() {
		return cost;
	}

	public void setCost(Double cost) {
		this.cost = cost;
	}

	@Override
	public String toString() {
		return "DataRunner [id=" + id + ", code=" + code + ",
cost=" + cost + "]";
	}

}
```

Q) If key is not present in properties file then trying to read same key, what will happen?

A) @Value --- Exception
@ConfigurationProperties - default value (null/0/0.0/false)

Internally ConfigProps uses set method. Logic looks like
object.setVariable(key)
ex: ob.setCode(code); //auto parsing supported

Q) If prefix/variables not matching in bulk loading, then what will happen?
A) @ConfigurationProperties will never throw eny error/exception.
It will read only matching variables data. Other values are ignored
So, it holds default for no-matching keys/variables.

After Login to Gmail => https://classroom.google.com/