Date : 21/12/2020
Spring Boot 9AM
Mr. RAGHU

Spring Boot # StopWatch

StopWatch (C):
This is a class given by Spring F/w , package:
org.springframework.util

To calculate time taken a for task/block/method/object cration..ect.

--Method--
start() : It will start time counting
start(taskName)
stop(): It will stop time counting

getTotalTimeMillis(): long
getTotalTimeSeconds():int

prettyPrint(): String

---

ctrl+shift+T Open any pre-defined class : StopWatch
ctrl+o View all members in class : variable,method,..

Time Factor Sacle

```
NANO_SCALE   = 1L;
MICRO_SCALE  = 1000L * NANO_SCALE;
MILLI_SCALE  = 1000L * MICRO_SCALE;
SECOND_SCALE = 1000L * MILLI_SCALE;
MINUTE_SCALE = 60L * SECOND_SCALE;
HOUR_SCALE   = 60L * MINUTE_SCALE;
DAY_SCALE    = 24L * HOUR_SCALE;
```

Example#1

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

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

@Component
public class TimeTestRunner implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
		//1. Create StopWatch Object
		StopWatch watch = new StopWatch();

		//2. start watch
		watch.start();

		//3. define logic
		for (int i = 0; i < Integer.MAX_VALUE; i++) {
```

```
                        Math.pow(i+1, 909856);
                }

                for (int i = 0; i < Integer.MAX_VALUE; i++) {
                        Math.pow(i+1, 909856);
                }

                for (int i = 0; i < Integer.MAX_VALUE; i++) {
                        Math.pow(i+1, 909856);
                }

                //4. Stop watch
                watch.stop();

                //5. printing details
                System.out.println("In Mill Sec " +
watch.getTotalTimeMillis());
                System.out.println("In Sec "
+watch.getTotalTimeSeconds());


        }

}
```

Example#2

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

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

@Component
public class TimeTestRunner implements CommandLineRunner {

        @Override
        public void run(String... args) throws Exception {
                //1. Create StopWatch Object
                //StopWatch watch = new StopWatch();
                StopWatch watch = new StopWatch("Time Test For
Loops"); //Watch#ID

                //2. start watch
                //watch.start();

                //3. define logic

                watch.start("Loop#1"); //Watch#taskName
                for (int i = 0; i < 999999999; i++) {
                        Math.pow(i+1, 909856);
                        Math.pow(i+1, 909856);
                }
                watch.stop();

                watch.start("Loop#2");
                for (int i = 0; i < 889999999; i++) {
                        Math.pow(i+1, 999956);
                }
```

```
                watch.stop();


                watch.start("Loop#3");
                for (int i = 0; i < 666699999; i++) {
                        Math.pow(i+1, 998856);
                        Math.pow(i+1, 998856);
                }
                watch.stop();

                //4. Stop watch
                //watch.stop();

                //5. printing details
                System.out.println(watch.prettyPrint());
                //System.out.println("In Mill Sec " +
watch.getTotalTimeMillis());
                //System.out.println("In Sec "
+watch.getTotalTimeSeconds());


        }

}
==========================================================================
==
*) Note:
*) For below code
watch.stop(); //watch stopped
watch.stop(); //exception
   IllegalStateException: Can't stop StopWatch: it's not running

watch.start();  // watch started
watch.start(); //exception
  IllegalStateException: Can't start StopWatch: it's already running

-------------------------------------------------------------
*) TimeUnit (Enum # Java 5)
It supports time factors and conversions

--------------------------_Example#3_----------------------
package in.nareshit.raghu.runner;

import java.util.concurrent.TimeUnit;

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

@Component
public class TimeFactorRunner implements CommandLineRunner {

        @Override
        public void run(String... args) throws Exception {
                //3 days-> hours
                System.out.println(TimeUnit.DAYS.toHours(3));
                //3 days-> mins
                System.out.println(TimeUnit.DAYS.toMinutes(3));
```

```
                //1 sec-> nano sec
                System.out.println(TimeUnit.SECONDS.toNanos(1));
                //10 min-> mill sec
                System.out.println(TimeUnit.MINUTES.toMillis(10));
        }
}
```

More about TimeFactors:
https://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html

---

Spring Boot Banner

When we start any Spring Boot application one logo is printed at console
called as Banner.

=> This Banner setup and printing is done when we run starer class.
=> We can even customize our code ie OFF Banner, modify data..etc

---Task Turn off Banner---

```
interface  Banner {
   enum Mode { OFF, CONSOLE, LOG } ;
}
```

---Modify starter class---

```
package in.nareshit.raghu;

import org.springframework.boot.Banner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringBoot2StopWatchApplication {

        public static void main(String[] args) {

//SpringApplication.run(SpringBoot2StopWatchApplication.class, args);

                SpringApplication sa = new
SpringApplication(SpringBoot2StopWatchApplication.class);
                sa.setBannerMode(Banner.Mode.OFF);
                sa.run(args);

        }

}
```

=> Default Banner.Mode.CONSOLE, banner is printed at console.

=> To provide our own banner file, you need to create one txt file under
    scr/main/resources folder. Bcoz Spring Boot has provided
    internal key:
         spring.banner.location=classpath:banner.txt

=> Here, classpath** means src/main/resources folder.

> Right click on src/main/resource folder > new > File
> Enter file name : banner.txt > Finish

=> Goto Banner Generator
  https://devops.datenkollektiv.de/banner.txt/index.html

 Enter Banner Text : [-----------]
 Choose Banner Font: [-----------]

 Copy Banner data and paste in banner.txt file (press ctlt+s and ctlr+F11)

Q) What are Command Line Arguments and VM/System Args?

A) Command Line Arguments:
   To pass input to application while running to main() method

  > java MainClassName.class val1 val2 val3 ...etc

  > java -jar <sampleJarName>.jar --main-class MainClassName val1 val2 val3 ...etc

 For Spring Boot Syntax: --key=val (option args)

 VM/System Args: Creating one variable at JVM/System level is called as
    VM Args, this is input to every application running in same VM.

 Syntax: -Dkey=val
 Read : System.getProperty(key):val

---

*)Note:
 We can give setup data to Spring Boot application (Properties/Arguments data)
 using below order:

 a) Command Line Arguments
    --key=val

 b) VM Arguments
   -Dkey=val

 c) YAML Arguments
   prefix:
      variable:<space>value

 d) Properties Arguments
   prefix.variable=value

=> Above data we can read using @Value (or) @ConfigurationProperties
=> To pass either Command Line Arguments or VM Arguments follow below steps

```
> Right click on main class/starter class
> Run as > Run Configuration
> Click on Arguments tab
> Enter details under
  Program  Arguments (Command Line Arguments)
   --my.app.export.data=CLA

  VM Arguments
   -Dmy.app.export.data=VMA

> Apply > Run

--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 DataInputTestRunner implements CommandLineRunner {

        @Value("${my.app.export.data}")
        private String exportData;

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

}
---------------------------------
*) also create : application.yml and application.properties
   with above key and check order.

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

Enum : set of possible values

Values : limited values(enum) / unlimited values

Exam --> Wrote -> Result (PASS, FAIL, ABSENT)

enum Result {
  PASS, FAIL, ABSENT
}

Gender --> MALE, FEMALE, OTHER

enum Gender {
  MALE, FEMALE, OTHER
}

public static final  String MALE="MALE";

numbers = 0,1,2........ (unlimited)
```

```
show(String s) {  }
show(Result r) { }
```

Enums Basics

https://www.youtube.com/c/NareshIT/search?query=enum