Date : 17-May-21
Spring Boot 9AM
Mr. RAGHU

Workspace:-
https://www.mediafire.com/file/0qcxdqeyp9ehasq/SpringCloud_9AM_17052021_RAGHU.zip/file

*) Native Config Server:-
=> Here, Native indicates local machine.
=> In case of Dev Environment, it is recomanded to use.
=> Until App goes to Production/UAT , we may not get actual GIT link.
   In that case we can use Native Config Server.

*** It is never used in Production/UAT Environments.
===coding steps===
1. Native Config Serever

Name : SpringCloudNativeConfigServer
Dep : Config Server

> application.properties
```
server.port=8888
# This is for external config server
#spring.cloud.config.server.git.uri=

# This is for native config server
#spring.cloud.config.server.native.search-locations=file:D:/abcd
spring.cloud.config.server.native.search-locations=classpath:mydata
spring.profiles.active=native
```

*) Create one folder mydata under src/main/resources folder
 > Right click on 'src/main/resources' folder
 > new > folder > Enter name 'mydata'
 > Finish

*) Create properties file under mydata folder
> Right click on mydata > new > file > enter name
application.properties
> Finish.

```
 [project-name]
    |
    |--src/main/resources
         |- mydata
             |- application.propreties   [native config file]
```

*) in this application.propreties , add you all MS# common keys
  my.app.title=FROM-LOCAL-CONFIG

---

2) Eureka Application
Name : SpringCloudCSEurekaServer
Dep : Eureka Server

=> At starter class: @EnableEurekaServer

```
=> Properties
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
-----------------------------------------------
3) MS Application
Name : SpringCloudCSEmployeeService
Dep  : Web, EurekaDiscovery Client, Config Client

=> At starter : @EnableEurekaClient
=> properties
server.port=8686
spring.application.name=EMPLOYEE-SERVICE
eureka.client.service-url.defaultZone=http://localhost:8761/eureka

=> RestController
package in.nareshit.raghu.rest;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/emp")
public class EmployeeRestController {

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

        public String showData() {
                return "Hello FROM MS# " + code;
        }
}

================Execution Order=========================
1. Eureka Server
2. Config Server
3. MS# app
http://192.168.0.2:8686/actuator/info
http://192.168.0.2:8686/emp/data

*) Note: When we run our MS# application with Config Client Dependency
   look at first line at console

ConfigServicePropertySourceLocator:
    Fetching config from server at : http://localhost:8888

*) Here, For Every MS# one Parent is Spring Cloud that has code for
  Config Client. We just added dependncy in MS#.
=========================================================================
=========
@RefreshScope :- This process is used to avoid restart of MS# and
Config Server

*) Each and every key in Spring Application (Even Spring Boot and MS#)
  is stored inside Environment(I)[org.springframework.core.env]
  Impl class is : StandardEnvironment(C)
```

```
  which stores data in key=val of .properties, .yml, OS properties,
  JVM Properties...etc

  By using method getProperty(key):val

--Example------
package in.nareshit.raghu;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Component;

@Component
public class TestData implements CommandLineRunner {

        @Autowired
        private Environment env;

        public void run(String... args) throws Exception {
                System.out.println(env.getClass());
                System.out.println(env.getProperty("my.app.title"));
        }

}
--application.properties--
my.app.title=SAMPLE-NIT
-------------------------
*) Work Flow:-
 If we make HTTP call 'POST' with URL /actuator/refresh, then
 RefreshScope(C) is trigger for ApplicationContext refresh event
 'ContextRefreshedEvent' there it gets latest Environment from Parent
 call and merged with our Environment memory.


*) @RefreshScope  : It gets modified keys from config server to MS#
                    only if HTTP Post Request is made from MS#

*) We are using POSTMNA Tool initially, now to automate HTTP call
   writing code using 'RestTemplate# post' to make call

*) Scheduling:- execute Request call POST call in loop
  ie get latest data every night 12 AM.
  cron = 0 0 0 * * *

--Either in MS# or diff project (web)------------
Name : SpringBoot2WebTest
Dep  : Web

> At starter class : @EnableScheduling
--Test--
package in.nareshit.raghu;

import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
```

```
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

@Component
public class RefreshMemoryProps {

        @Scheduled(cron = "0 24 10 * * *")
        public void run() throws Exception {
                RestTemplate rt = new RestTemplate();
                HttpHeaders header = new HttpHeaders();
                header.setContentType(MediaType.APPLICATION_JSON);
                HttpEntity<String> entity = new HttpEntity<String>("
{}", header);

rt.exchange("http://192.168.0.7:8686/actuator/refresh",
HttpMethod.POST, entity, String.class);
                System.out.println("DONE");
        }
}
---------------------------------
```