LAB- Cloud Config Server

Config Server

In this Section, you will be working on 08-01-spring-cloud-config-server-start

Open pom.xml and complete TODO-1

Open SpringCloudConfigServer.java and complete TODO-2

Open application.yml and observe how we have already configured git url for configuration

Also, Observe that server.port is configured as 4000

Now start the ConfigServer by running SpringCloudConfigServer.java .

Firstly, open my git repository at the following URL
`https://github.com/sivaprasadvalluru/springcloudtraining`

See the contents of
https://github.com/sivaprasadvalluru/springcloudtraining/blob/master/springcloudconfigclient.yml
and
https://github.com/sivaprasadvalluru/springcloudtraining/blob/master/springcloudconfigclient-prod.yml

Now give a request to http://localhost:4000/springcloudconfigclient/default and observe that you get contents of springcloudconfigclient.yml as json

Give a request to http://localhost:4000/springcloudconfigclient/prod and observe that you get the contents of springcloudconfigclient-prod.yml

In this Section, you will be working on 08-02-spring-cloud-config-client-solution

Open pom.xml and observe that spring-cloud-starter-config is configured as dependency .

Open application.yml and observe that we configured application name as springcloudconfigclient (same as the the yml file name in my git repository)

Also observe how we have configured cloud config server uri pointing to the url of cloud config server you started in previous step.

Open MyController.java and observe the code

Run ConfigTestApplication.java. Can you tell at which port will the embedded tomcat starts at ? Did we confgiure server.port in bootstrap.yml?

From which file in git will the the config client get the content?

Now Stop ConfigTestApplication.java and run it by passing -Dspring.profiles.active=prod

Can you now tell at which port will embedded tomcat start at ? From which file in git will the the config client get the content?

Give a request to http://localhost:5050/msg. You should see hello +value of test.message in **springcloudconfigclient-prod.yml**

Change the value of test.message in **springcloudconfigclient-prod.yml** in git and give request to http://localhost:5050/msg

Did you observe the changed message ?

We want the configuration changes to be loaded .

Annotate the Controller with @RefreshScope

Add spring-boot-actuator as dependency in pom.xml

```
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Now Stop ConfigTestApplication.java and run it by passing -Dspring.profiles.active=prod

give request to http://localhost:5050/msg

You should see hello +value of test.message in **springcloudconfigclient-prod.yml**

Change the value of test.message in **springcloudconfigclient-prod.yml** in git and give request to http://localhost:5050/msg

Did you observe the changed message ?

Open POSTMAN and make a post request to http://localhost:5050/actuator/refresh

Did the application context get refreshed?

Configure following in bootstrap.yml

```
management:
  endpoints:
    web:
      exposure:
        include:
        - refresh
```

Open POSTMAN and make a post request to http://localhost:5050/actuator/refresh

See the logs in console and observe that application context is refreshed.

Now make a request http://localhost:5050/msg and observe the chages made in git are reflected.