Spring Cloud Configuration Server is a centralized application that manages all the application related configuration properties

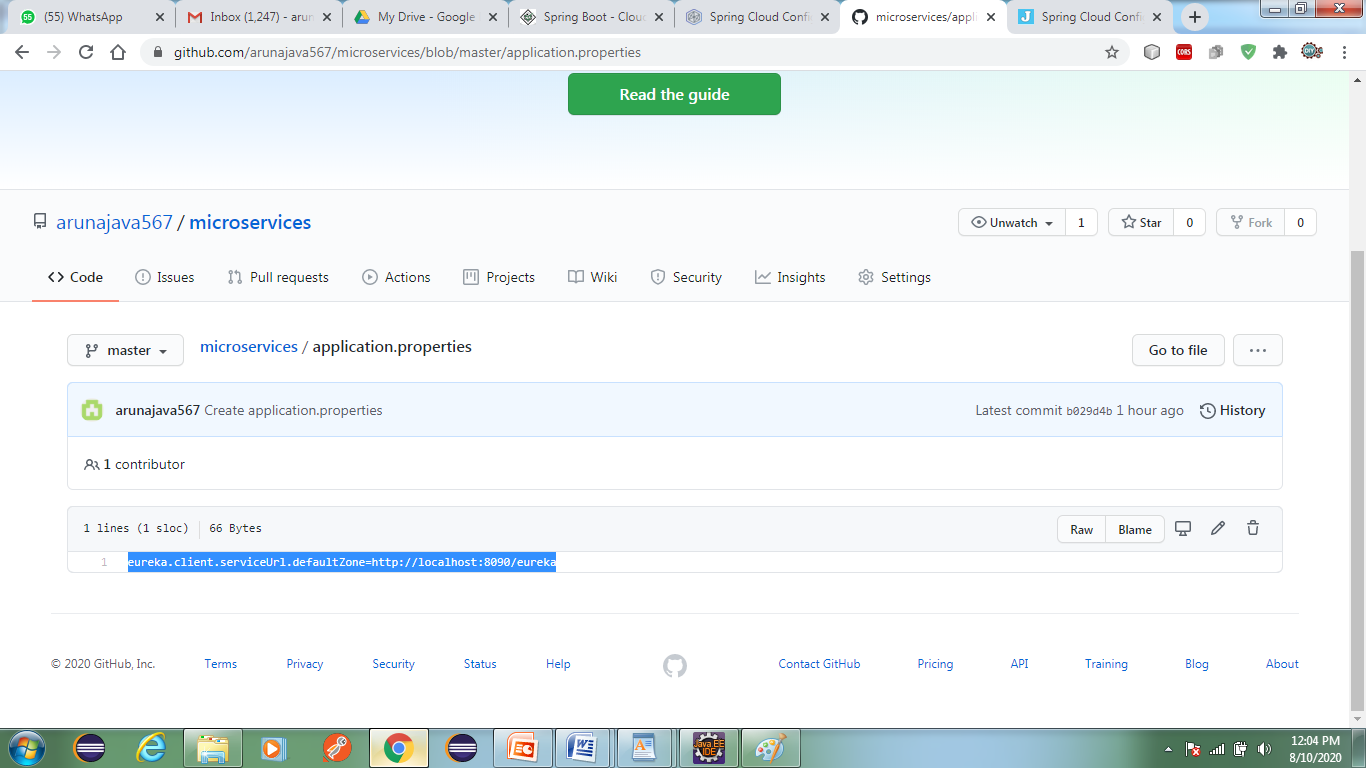
* One of the features is to maintain application configurations outside the application.
* This will enable configuration change like db properties,profiling properties etc without restarting microservice application.

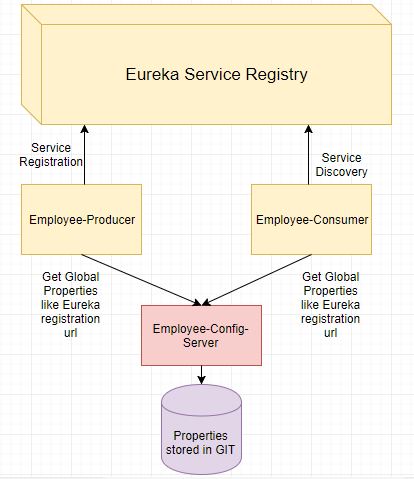
Annotation to enable ConfigServer

@EnableConfigServer

* Db properties
* Different profiles can be maintained like dev,prod

Config properties can be configured into local git repository or remote git repository.





We will create a config client and use properties stored in the Spring config server.  Please make sure spring config server is up and running.

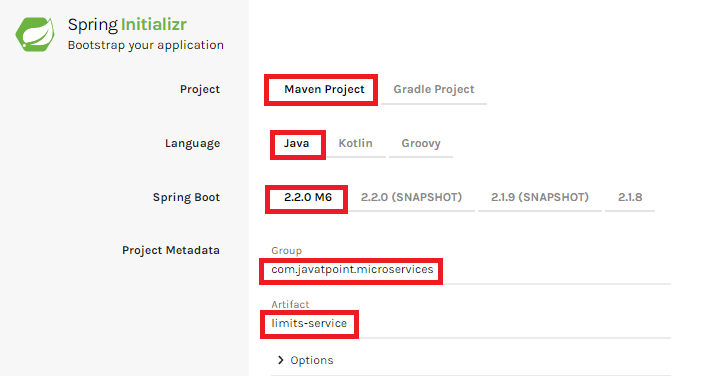
Creating a Simple Microservice

**Step 1**: Create a Maven project using Spring Initializr <https://start.spring.io/>

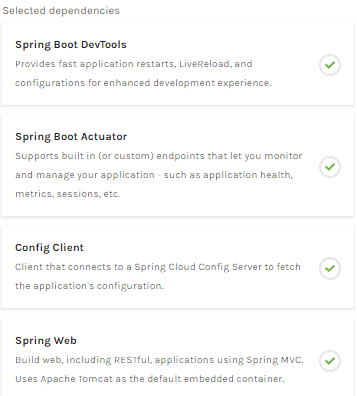
**Step 2**: Choose the Spring Boot version **2.2.0 M6** or higher version. Do not choose the snapshot version.

**Step 3**: Provide the **Group** name. In our case c**om.diy**

**Step 4**: Provide the **Artifact id**. We have provided **limits-service**.



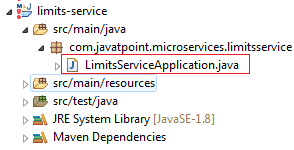
**Step 5**: Add the following dependencies: **Spring Web, Spring Boot DevTools, Spring Boot Actuator, Config Client**.



**Step 6**: Click **on Generate the project** button. A **zip** file will download, extract it into the hard disk.

**Step 7**: Now, open the **eclipse**. Import the created maven project. It takes some time to download the required files.

**Step 8**: Once the project is downloaded, go to **src/main/java**. Open the **LimitsServiceApplication**.



**Step 9**: Now run the **LimitsServiceApplication.java** as Java Application.

**It started the Tomcat on port(s) 8080 (http).**

Now we will add couple of services in the above project. For this we will have to follow the following steps:

**Step 1**: Open **application.properties** file and write the following code:

1. spring.application.name=limits-service      //name of application

**Step 2**: Create a class file with name **LimitsConfigurationController.java** in the folder src/main/java under the package **com.javatpoint.microservices.limitsservice** and write the following code:

1. **package** com.javatpoint.microservices.limitsservice;
2. **import** org.springframework.web.bind.annotation.GetMapping;
3. **import** org.springframework.web.bind.annotation.RestController;
4. **import** com.javatpoint.microservices.limitsservice.bean.LimitConfiguration;
5. @RestController
6. **public** **class** LimitsConfigurationController
7. {
8. @GetMapping("/limits")
9. **public** LimitConfiguration retriveLimitsFromConfigurations()
10. {
11. **return** **new** LimitConfiguration(1000, 1);
12. }
13. }

**Step 3**: Create a class file with name **LimitConfiguration.java** in the folder **src/main/java** under the package **com.javatpoint.microservices.limitservice.bean** and write the following code:

1. **package** com.javatpoint.microservices.limitsservice.bean;
2. **public** **class** LimitConfiguration
3. {
4. **private** **int** maximum;
5. **private** **int** minimum;
6. //no-argument constructor
7. **protected** LimitConfiguration()
8. {
9. }
10. //generating getters
11. **public** **int** getMaximum()
12. {
13. **return** maximum;
14. }
15. **public** **int** getMinimum()
16. {
17. **return** minimum;
18. }
19. //genetrating constructor using fields
20. **public** LimitConfiguration(**int** maximum, **int** minimum)
21. {
22. **super**();
23. **this**.maximum = maximum;
24. **this**.minimum = minimum;
25. }
26. }

Type the **localhost:8080/limits** in the browser and press enter, we get the JSON response as output.

**Output**

{

maximum: 1000,

minimum: 1

}

Adding services to the application.properties

In the previous program, we will modify the code according to the requirement.

Now we call the **limits-service** from the **application.properties** file. In this file, we are configuring a couple of values.

1. limits-service.minimum=99
2. limits-service.maximum=9999

There is a better approach in Spring Boot to read values from the configuration using the annotation **@ConfigurationProperties**.

**Step 1**: Create a class with name **Configuration.java** in the folder **src/main/java** under the package **com.javatpoint.microservices.limitservice**.

**Step 2**: Add the annotations **@Component** and **@ConfigurationProperties**.

**Step 3**: Declare two variables **minimum** and **maximum**.

**Step 4**: If we are using the Configuration file, we need to generate getters and setters.

The Configuration.java file look like this.

1. **package** com.javatpoint.microservices.limitsservice;
2. **import** org.springframework.boot.context.properties.ConfigurationProperties;
3. **import** org.springframework.stereotype.Component;
4. @Component
5. @ConfigurationProperties("limits-service")
6. **public** **class** Configuration
7. {
8. **private** **int** maximum;
9. **private** **int** minimum;
10. **public** **void** setMaximum(**int** maximum)
11. {
12. **this**.maximum = maximum;
13. }
14. **public** **void** setMinimum(**int** minimum)
15. {
16. **this**.minimum = minimum;
17. }
18. **public** **int** getMaximum()
19. {
20. **return** maximum;
21. }
22. **public** **int** getMinimum()
23. {
24. **return** minimum;
25. }
26. }

**Step 5**: Now move to **LimitsConfigurationController.java** file and modify the code. In this we will use Configuration.

1. **package** com.javatpoint.microservices.limitsservice;
2. **import** org.springframework.beans.factory.annotation.Autowired;
3. **import** org.springframework.web.bind.annotation.GetMapping;
4. **import** org.springframework.web.bind.annotation.RestController;
5. **import** com.javatpoint.microservices.limitsservice.bean.LimitConfiguration;
6. @RestController
7. **public** **class** LimitsConfigurationController
8. {
9. @Autowired
10. **private** Configuration configuration;
11. @GetMapping("/limits")
12. **public** LimitConfiguration retriveLimitsFromConfigurations()
13. {
14. //getting values from the properties file
15. **return** **new** LimitConfiguration(configuration.getMaximum(), configuration.getMinimum());
16. }
17. }

Now refresh the browser page. It shows the JSON format of the updated values which are configured in **application .properties** file.

**Output**

{

maximum: 999,

minimum: 99

}

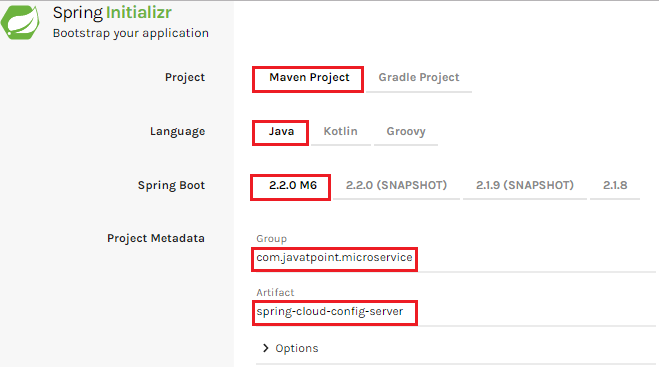
Setting up Spring Cloud Config Server

**Step 1:** Create a Maven project using Spring Initializr <https://start.spring.io/>

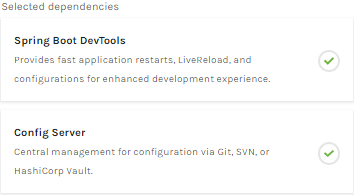
**Step 2:** Choose the Spring Boot version **2.2.0 M6** or higher version. Do not choose the snapshot version.

**Step** **3:** Provide the **Group** name. In our case, **com.javatpoint.microservices.**

**Step 4:** Provide the **Artifact id**. We have provided **spring-cloud-config-server.**



**Step 5:**Add the **Spring Boot DevTools**and**Config Server**dependencies**.**



**Step 6:** Click on**Generate the project** button. A zip file will download, extract it in the hard disk.

**Step 7:** Now, open the **eclipse.** Import the downloaded maven project. It will download the required files.

In the next step, we will create a simple Git repository and configure the spring cloud config server to pick up the values from the particular Git repository. We need to install the local Git.

Installing Git and creating a local repository

**Step 1:**Download Git from <https://git-scm.com/> and install it.

**Step 2:**Create a Git repository and store the files that we want to be able to configure a limits-service. We will try to access them from the spring-cloud-config-server. Open the Git bash and type the following commands:

Creating a new directory:

1. mkdir git-localconfig-repo
2. cd git-localconfig-repo/

Initializing a new Git repository:

1. git init

It initializes an **empty**git repository.

**Step 3:**Now move to the **spring-cloud-config-server** project and add a link to the specific folder.

1. Right-click on the **spring-cloud-config-server**project**.**
2. Click on **Build Path**->**Configure Build Path**…
3. Select the **Source** tab.
4. Click on **Link Source** and browse the folder **git-localconfig-repo**.
5. Right click on the folder-> **New** -> **Other** -> **File** -> **Next** -> Provide the file name**: limits-service-properties**-> **Finish**.
6. Now write the following code in the properties file:
7. limits-service.minimum=8
8. limits-service.maximum=888

**Step 4:**Configure the user name and user email:

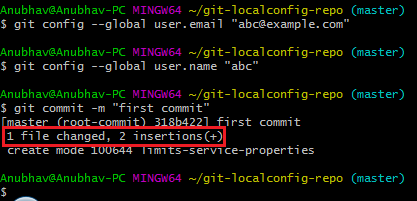
1. git config -global user.email abc@example.com
2. git config -global user.name "abc"

The command commits any file we have added with the git add command and also commits any files we have changed since then.

1. git add -A

Now execute the command to commit the changes in the repository. It records or snapshots the file permanently in the version history.

1. git commit -m "first commit"



We can see that a file is changed with two new instructions. These instructions are changed in the local repository.