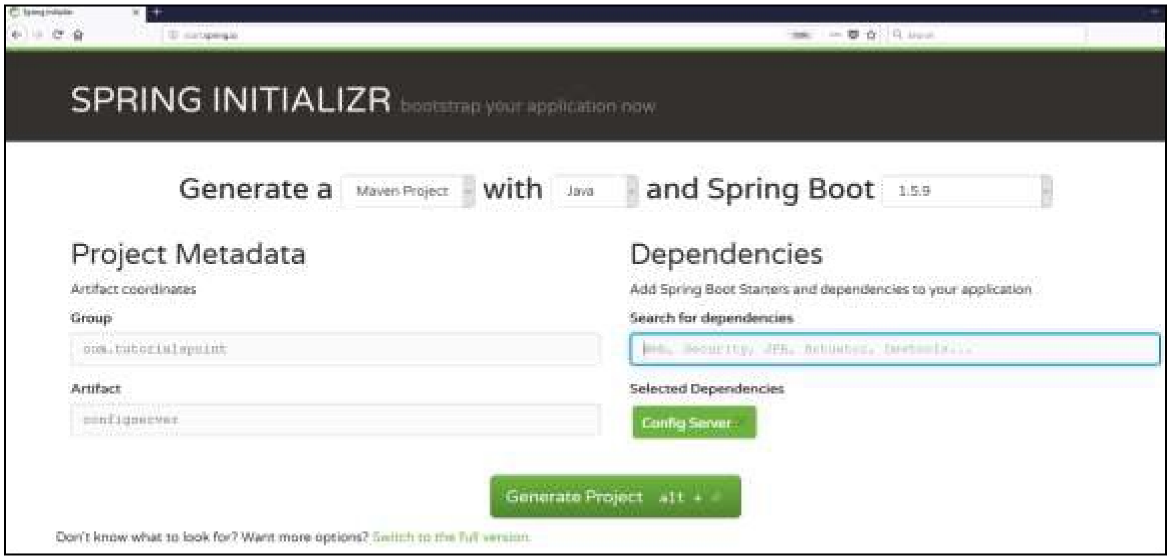
Spring Boot - Cloud Configuration Server

Spring Cloud Configuration Server is a centralized application that manages all the application related configuration properties. In this chapter, you will learn in detail about how to create Spring Cloud Configuration server.

Creating Spring Cloud Configuration Server

First, download the Spring Boot project from the Spring Initializer page and choose the Spring Cloud Config Server dependency. Observe the screenshot given below −



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 1.5.9

Project Metadata

Artifact coordinates

Group: com.tutorialspoint

Artifact: configserver

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies: Config Server

Generate Project alt + ⏎
Don't know what to look for? Want more options? Switch to the full version.

Now, add the Spring Cloud Config server dependency in your build configuration file as explained below −

Maven users can add the below dependency into the pom.xml file.

```
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-config-server</artifactId>
</dependency>
```

Now, add the **@EnableConfigServer** annotation in your main Spring Boot application class file. The **@EnableConfigServer** annotation makes your Spring Boot application act as a Configuration Server.

The main Spring Boot application class file is given below −

```
package com.tutorialspoint.configserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.config.server.EnableConfigServer;

@SpringBootApplication
@EnableConfigServer
public class ConfigserverApplication {
   public static void main(String[] args) {
      SpringApplication.run(ConfigserverApplication.class, args);
   }
}
```

Now, add the below configuration to your properties file and replace the application.properties file into bootstrap.properties file. Observe the code given below −

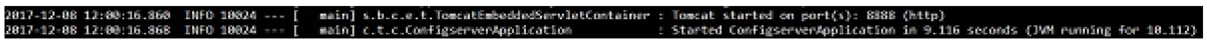
```
server.port = 8888
spring.cloud.config.server.native.searchLocations=file:///C:/configprop/
SPRING_PROFILES_ACTIVE=native
```

Configuration Server runs on the Tomcat port 8888 and application configuration properties are loaded from native search locations.

Now, in **file:///C:/configprop/,** place your client application - **application.properties** file. For example, your client application name is config-client, then rename your application.properties file as config-client.properties and place the properties file on the path **file:///C:/configprop/**.

The code for config-client properties file is given below −
welcome.message = Welcome to Spring cloud config server

Run the Application
Now, the application has started on the Tomcat port 8888 as shown here −



```
2017-12-08 12:00:16.860  INFO 10024 --- [   main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8888 (http)
2017-12-08 12:00:16.868  INFO 10024 --- [   main] c.t.c.ConfigserverApplication            : Started ConfigserverApplication in 9.116 seconds (JVM running for 10.112)
```

Now hit the URL http://localhost:8888/config-client/default/master on your web browser and you can see your config-client application configuration properties as shown here.

External Git Configuration in Spring-Cloud-Config server

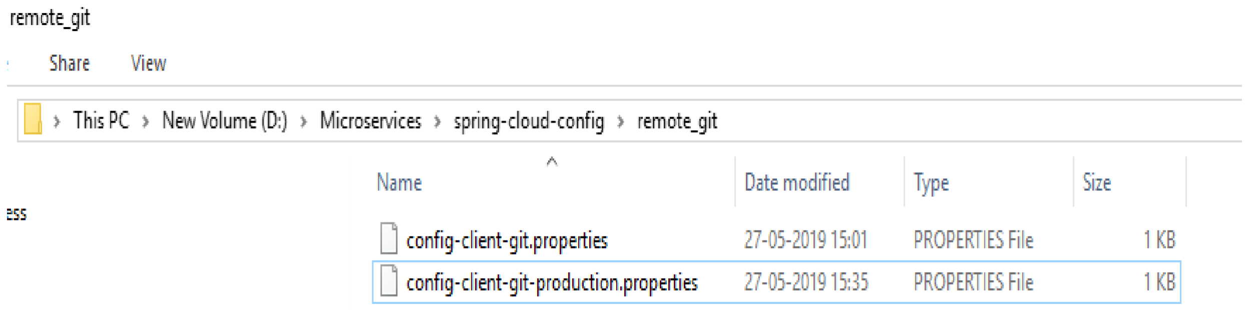
Create the configuration file in the git repositories.
1. File name should be Spring-Cloud-Config-Client application name i.e.
    bootstrap.properties
    `spring.application.name=config-client-git`

config-client-git.properties	- Default when no active profile
config-client-git-production.properties	- Active profile is Production

    `spring.profiles.active=production`

application.properties

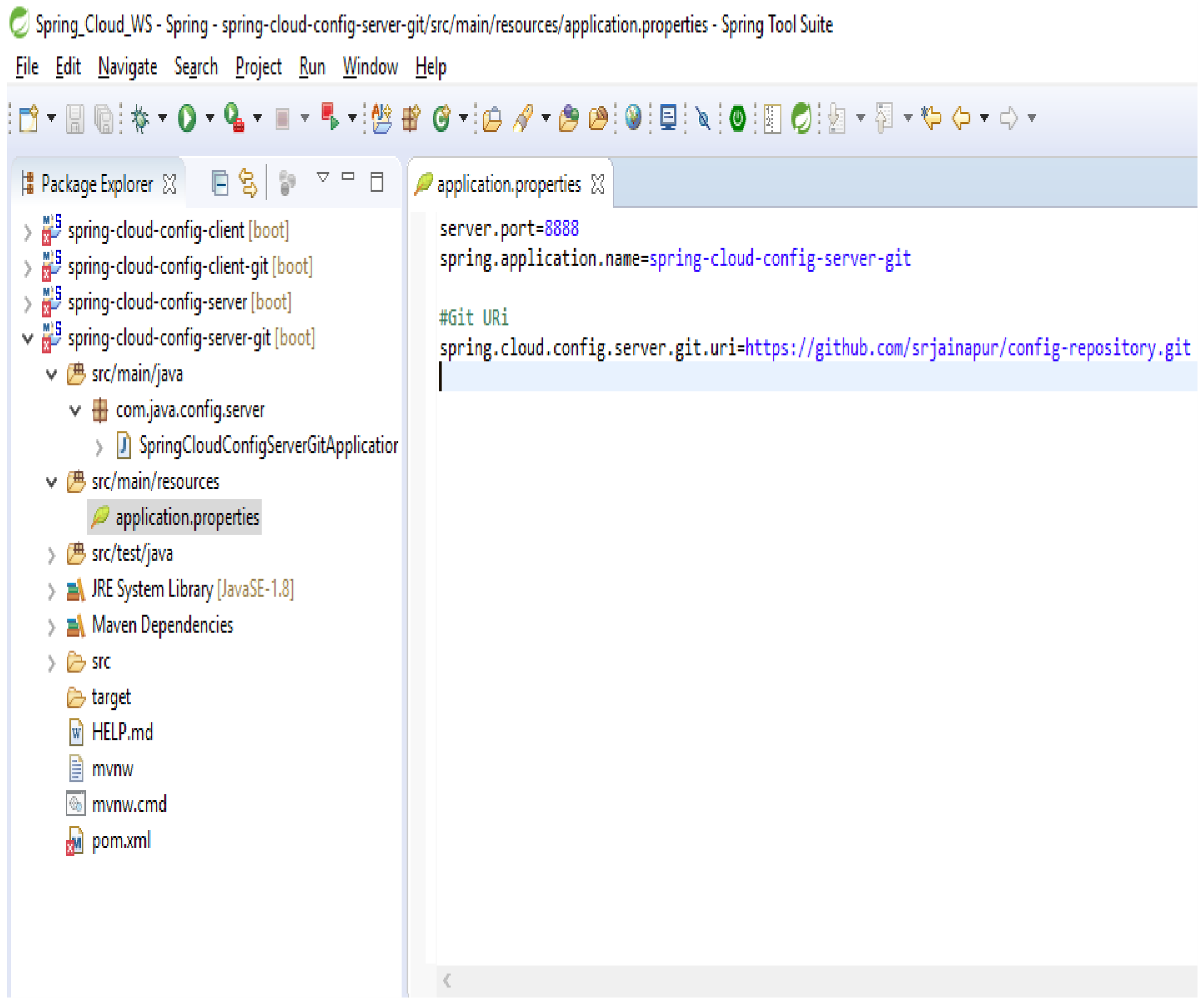
```
server.port=8888
spring.application.name=spring-cloud-config-server-git

#Git URi
spring.cloud.config.server.git.uri=https://github.com/srjainapur/config-repository.git
```


remote_git

Share View

This PC › New Volume (D:) › Microservices › spring-cloud-config › remote_git

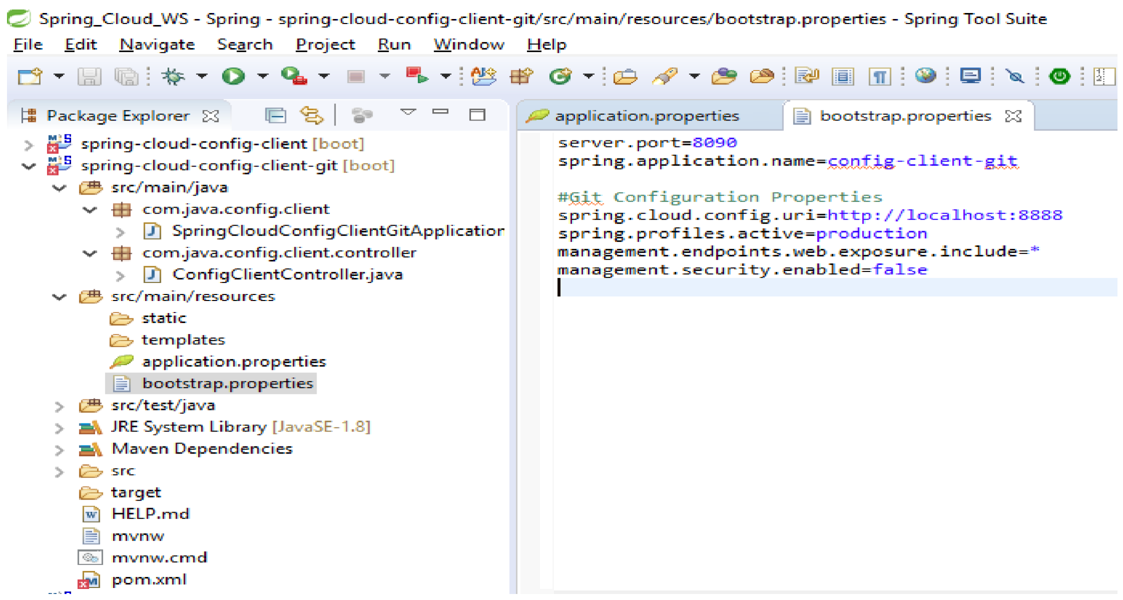
Name	Date modified	Type	Size
config-client-git.properties	27-05-2019 15:01	PROPERTIES File	1 KB
config-client-git-production.properties	27-05-2019 15:35	PROPERTIES File	1 KB


Spring_Cloud_WS - Spring - spring-cloud-config-server-git/src/main/resources/application.properties - Spring Tool Suite

File Edit Navigate Search Project Run Window Help
Package Explorer
spring-cloud-config-client [boot]
spring-cloud-config-client-git [boot]
spring-cloud-config-server [boot]
spring-cloud-config-server-git [boot]
src/main/java
com.java.config.server
SpringCloudConfigServerGitApplicatior
src/main/resources
application.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
application.properties

```
server.port=8888
spring.application.name=spring-cloud-config-server-git

#Git URi
spring.cloud.config.server.git.uri=https://github.com/srjainapur/config-repository.git
```

Spring-Cloud-Client



Spring_Cloud_WS - Spring - spring-cloud-config-client-git/src/main/resources/bootstrap.properties - Spring Tool Suite
File Edit Navigate Search Project Run Window Help
Package Explorer
application.properties
bootstrap.properties
spring-cloud-config-client [boot]
spring-cloud-config-client-git [boot]
src/main/java
com.java.config.client
SpringCloudConfigClientGitApplicatior
com.java.config.client.controller
ConfigClientController.java
src/main/resources
static
templates
application.properties
bootstrap.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml

```
server.port=8090
spring.application.name=config-client-git

#Git Configuration Properties
spring.cloud.config.uri=http://localhost:8888
spring.profiles.active=production
management.endpoints.web.exposure.include=*
management.security.enabled=false
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.1.5.RELEASE</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.java.config.client</groupId>
<artifactId>spring-cloud-config-client-git</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>spring-cloud-config-client-git</name>
<description>Demo project for Spring Boot</description>

<properties>
<java.version>1.8</java.version>
<spring-cloud.version>Greenwich.SR1</spring-cloud.version>
</properties>

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

```
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-config</artifactId>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>

<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>${spring-cloud.version}</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>

</project>
```

Note :

1. Create properties based on the name of client application i.e if name of Spring-Cloud-Config-Client application name in bootstarp.properties is **config-client-git** then file names will be as follows

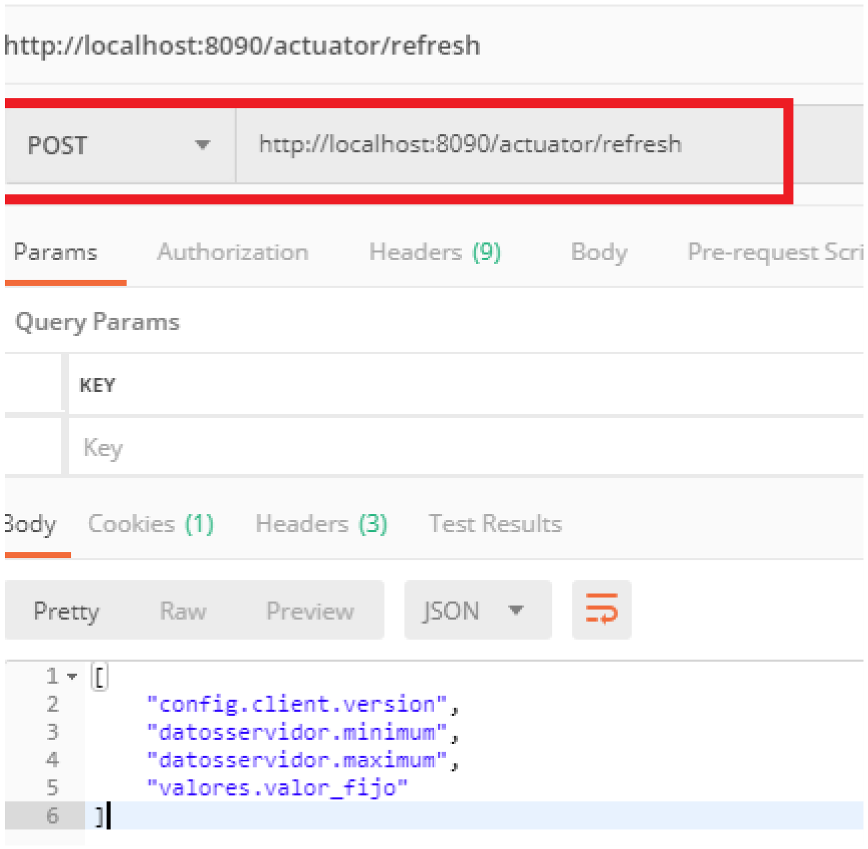
   bootstarp.properties
   ```
   spring.application.name=config-client-git
   ```

config-client-git.properties	- When there are no Active profile
config-client-git-production.properties	- When Active Profile is production
config-client-git-development.properties	- When Active Profile is development

2. Commit these files in the GIT repositories using following commads
   - git init
   - git add **.**
   - git commit -m "**first commit**"
   - git remote add origin https://github.com/srjainapur/temp.git
   - git push -u origin master
3. After committing the file access the application using url http://localhost:8090//getProperty
4. Now change the property value in Active profile file and committ the changes into git.

5. Now before hitting the client URL again we need to hit the **refresh** url to refelect the changes **http://localhost:8090/actuator/refresh**

   Note : It should be POST method



http://localhost:8090/actuator/refresh

POST http://localhost:8090/actuator/refresh

```
[
    "config.client.version",
    "datosservidor.minimum",
    "datosservidor.maximum",
    "valores.valor_fijo"
]
```

6. Now after refresh if you hit again http://localhost:8090//getProperty you will get the updated values without restarting any application