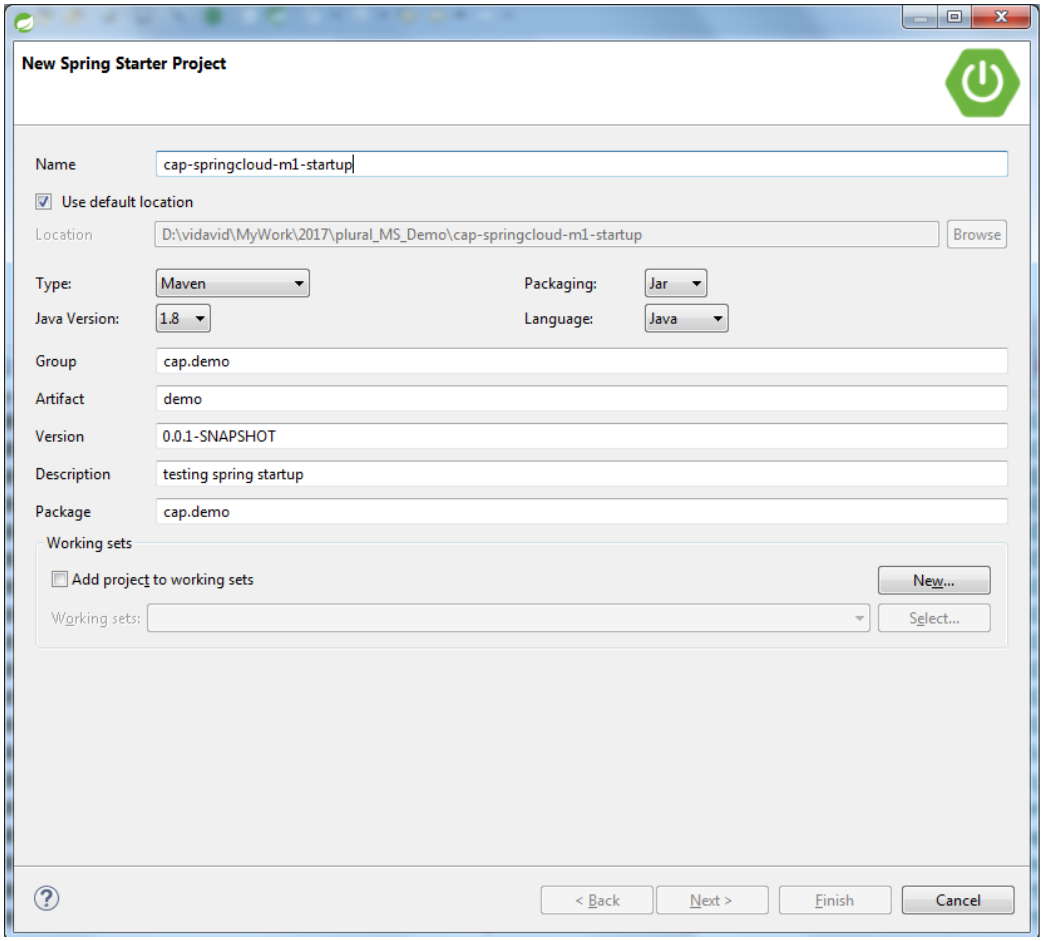
# Spring Boot

## LAB – 1

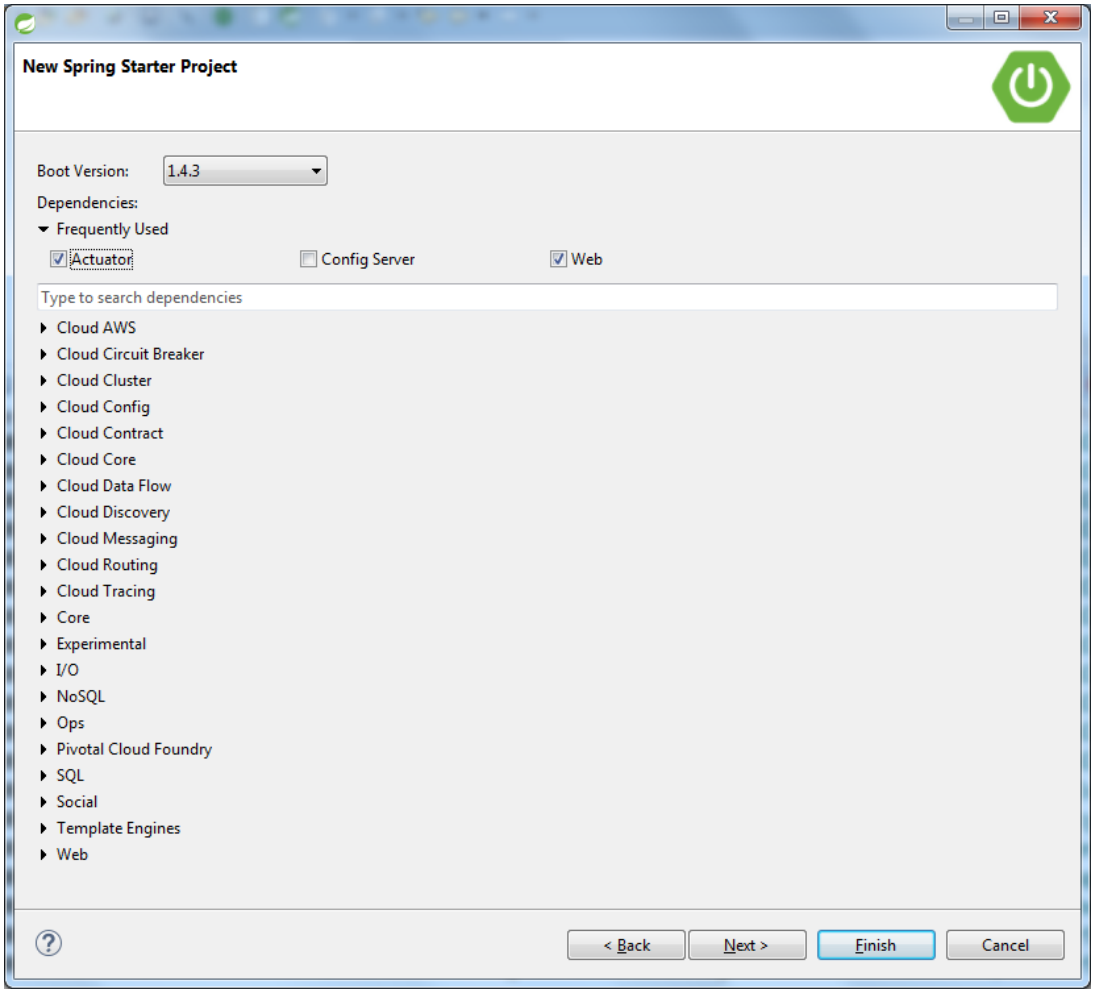
***Create one simple Spring Boot web application with resolving dependencies, edit properties file and show actuator components usage.***

**Steps:**

- Install SpringToolSuite 3.8.1.Release IDE.
- File → New → Spring Starter Project . And create project name as cap-springcloud-m1-startup with the following setup.




- In the next window add ConfigServer and Actuator Dependencies. And then click Finish. You will be getting one project **cap-springcloud-m2-startup** in the project explorer window.




- Open the Main Class under cap.demo package. Edit the main class as mentioned below:

**CapSpringcloudM1StartupApplication.java**

```
package cap.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class CapSpringcloudM1StartupApplication {
```

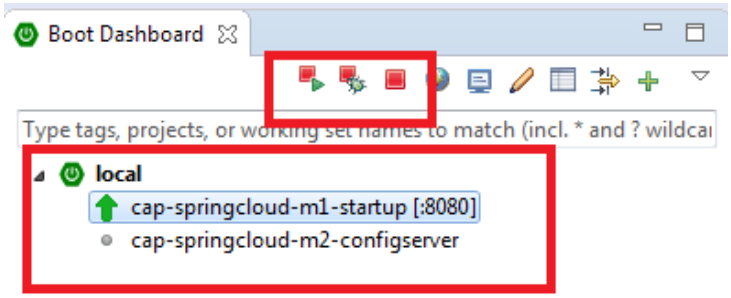
```
	public static void main(String[] args) {
		SpringApplication.run(CapSpringcloudM1StartupApplication.class, args);
	}

	@RequestMapping(value="/greetings",method=RequestMethod.GET)
	public String sayhello(){
		return "Hello! Welcome To Spring Boot!";
	}
}
```

**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>

	<groupId>cap.demo</groupId>
	<artifactId>demo</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>cap-springcloud-m1-startup</name>
	<description>testing spring startup</description>

	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>1.4.3.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
		<java.version>1.8</java.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.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
</project>
```

- Add server.port = 8080 in application.properties file under resources floder.
- Go to BootDashboard window, select your project click run  icon.



- You can aslo run project by **Right Click → Run As →Spring Boot App**

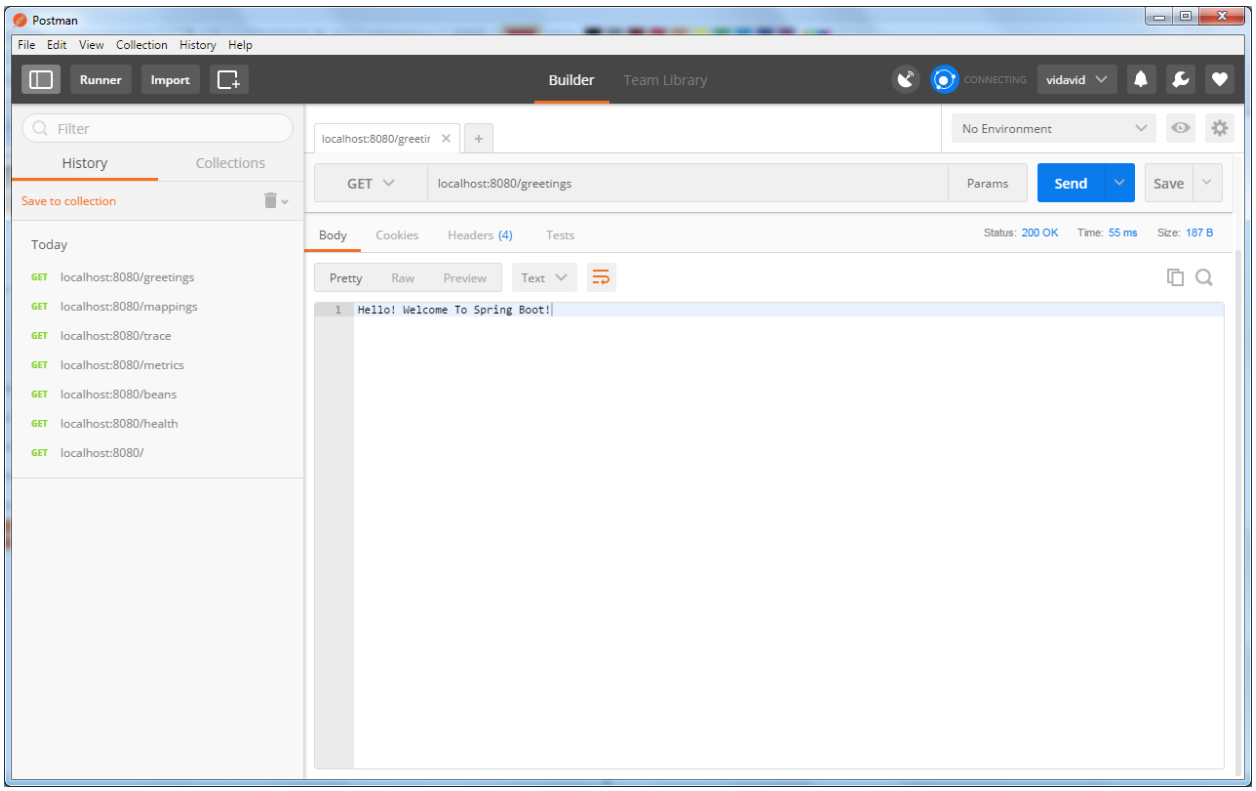
**Output:**

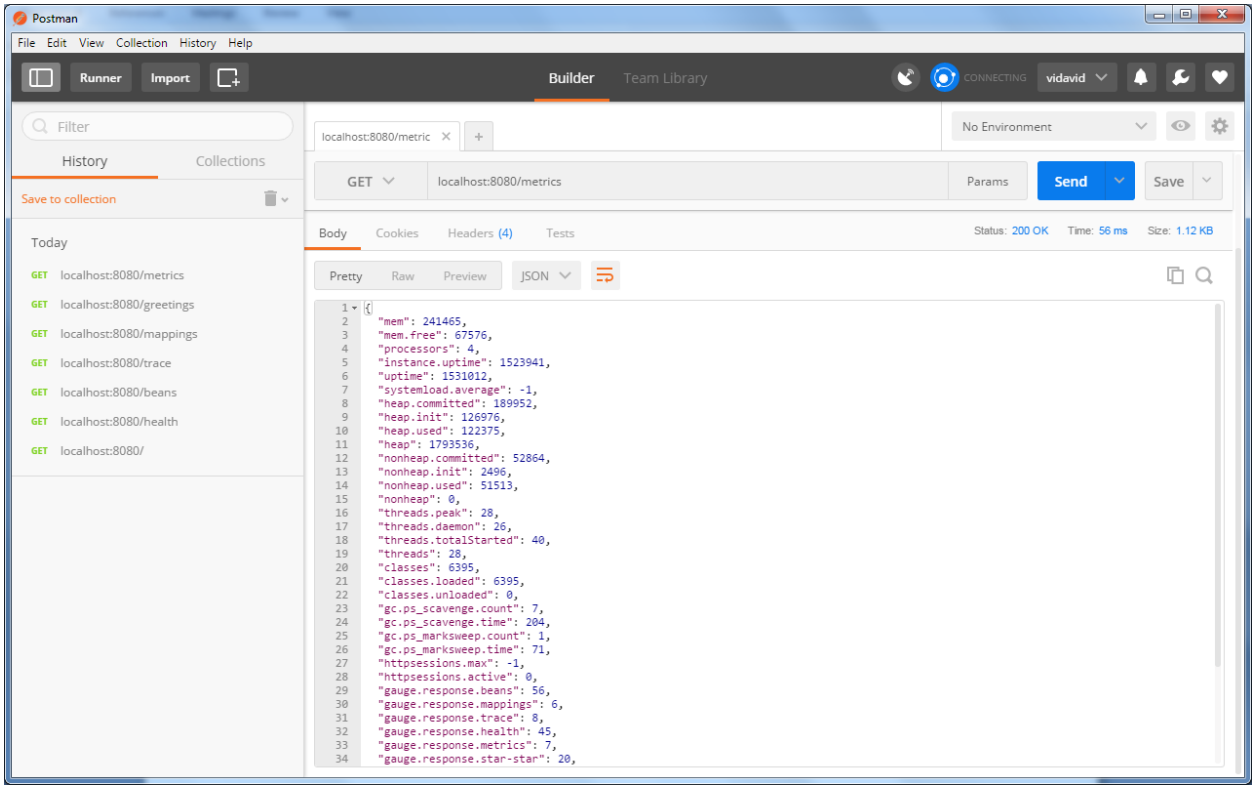
If you look at the console window you will be get the below log:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.4.3.RELEASE)

2017-01-10 10:47:39.902  INFO 3600 --- [           main] c.d.CapSpringcloudM1StartupApplication   : Starting
CapSpringcloudM1StartupApplication on LIN73000499 with PID 3600 (D:\vidavid\MyWork\2017\plural_MS_Demo\cap-springcloud-m1-
startup\target\classes started by vidavid in D:\vidavid\MyWork\2017\plural_MS_Demo\cap-springcloud-m1-startup)
2017-01-10 10:47:39.905  INFO 3600 --- [           main] c.d.CapSpringcloudM1StartupApplication   : No active profile set, falling back to default
profiles: default
2017-01-10 10:47:39.994  INFO 3600 --- [           main] ationConfigEmbeddedWebApplicationContext : Refreshing
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@7ec7ffd3: startup date [Tue Jan 10
10:47:39 IST 2017]; root of context hierarchy
2017-01-10 10:47:42.730  INFO 3600 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2017-01-10 10:47:42.755  INFO 3600 --- [           main] o.apache.catalina.core.StandardService   : Starting service Tomcat
2017-01-10 10:47:42.756  INFO 3600 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache
Tomcat/8.5.6
2017-01-10 10:47:42.997  INFO 3600 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
WebApplicationContext
public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.MetricsMvcEndpoint.value(java.lang.String)
2017-01-10 10:47:45.057  INFO 3600 --- [           main] o.s.b.a.e.mvc.EndpointHandlerMapping     : Mapped "{[/metrics ||
/metrics.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object
org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()
2017-01-10 10:47:45.058  INFO 3600 --- [           main] o.s.b.a.e.mvc.EndpointHandlerMapping     : Mapped "{[/trace ||
/trace.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object
org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()
2017-01-10 10:47:45.066  INFO 3600 --- [           main] o.s.b.a.e.mvc.EndpointHandlerMapping     : Mapped
"{[/env/{name:.*}],methods=[GET],produces=[application/json]}" onto public java.lang.Object
org.springframework.boot.actuate.endpoint.mvc.EnvironmentMvcEndpoint.value(java.lang.String)
2017-01-10 10:47:45.066  INFO 3600 --- [           main] o.s.b.a.e.mvc.EndpointHandlerMapping     : Mapped "{[/env ||
/env.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object
org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()
2017-01-10 10:47:45.068  INFO 3600 --- [           main] o.s.b.a.e.mvc.EndpointHandlerMapping     : Mapped "{[/dump ||
/dump.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object
org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()
2017-01-10 10:47:45.071  INFO 3600 --- [           main] o.s.b.a.e.mvc.EndpointHandlerMapping     : Mapped "{[/autoconfig ||
/autoconfig.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object
org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()
2017-01-10 10:47:45.239  INFO 3600 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2017-01-10 10:47:45.256  INFO 3600 --- [           main] o.s.c.support.DefaultLifecycleProcessor  : Starting beans in phase 0
2017-01-10 10:47:45.452  INFO 3600 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2017-01-10 10:47:45.458  INFO 3600 --- [           main] c.d.CapSpringcloudM1StartupApplication   : Started
CapSpringcloudM1StartupApplication in 6.15 seconds (JVM running for 7.823)
2017-01-10 10:47:58.931  INFO 3600 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring FrameworkServlet
'dispatcherServlet'
2017-01-10 10:47:58.931  INFO 3600 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet':
initialization started
```

```
2017-01-10 10:47:58.960  INFO 3600 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initialization completed in 29 ms
```

- Open Postman Application to check the rest web services, if you enter localhost:8080/greetings below screen will appear.
- Change your URL as mentioned below and analyse your output.
    - **localhost:8080/greetings**
    - **localhost:8080/mappings**
    - **localhost:8080/trace**
    - **localhost:8080/health**
    - **localhost:8080/beans**
    - **localhost:8080/env**
    - **localhost:8080/metrics**






**Learning:**

- From the example we learnt simple Spring Boot Web application config and actutaor components.

You have successfully completed this lab!

**LAB – 2**

***Create Spring Starter Project, Annotate the main Class and set the application properties. Add local configuration files and run the Spring boot application. Query all configurations with POSTMAN.***

**Steps:**

- Open SpringToolSuite 3.8.1.Release IDE.
- File → New → Spring Starter Project . And create project name as cap-springcloud-m2-configserver with the following setup.
- In the next window add **Spring Cloud Config and Actuator** Dependencies. And then click Finish. You will be getting one project **cap-springcloud-m2-configserver** in the project explorer window.
    - In the main class add **@EnableConfigServer** annotation.

**CapSpringcloudM2ConfigserverApplication.java**

```
package cap.demo;

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

@SpringBootApplication
@EnableConfigServer
public class CapSpringcloudM2ConfigserverApplication {

	public static void main(String[] args) {
		SpringApplication.run(CapSpringcloudM2ConfigserverApplication.class, args);
	}
}
```

- Create folder called **config** under resources, add **app1.properties, app2.properties** and **app3.properties** under config with the following greeting messages in properties file.
    - app1.properties
      **greeting=Hello!**
    - app2.properties
      **greeting=Howdy**
    - app3.properties
      **greeting=Hey There!**

- In applications.properties file add the following properties:

```
server.port=8888
spring.profiles.active=native
```

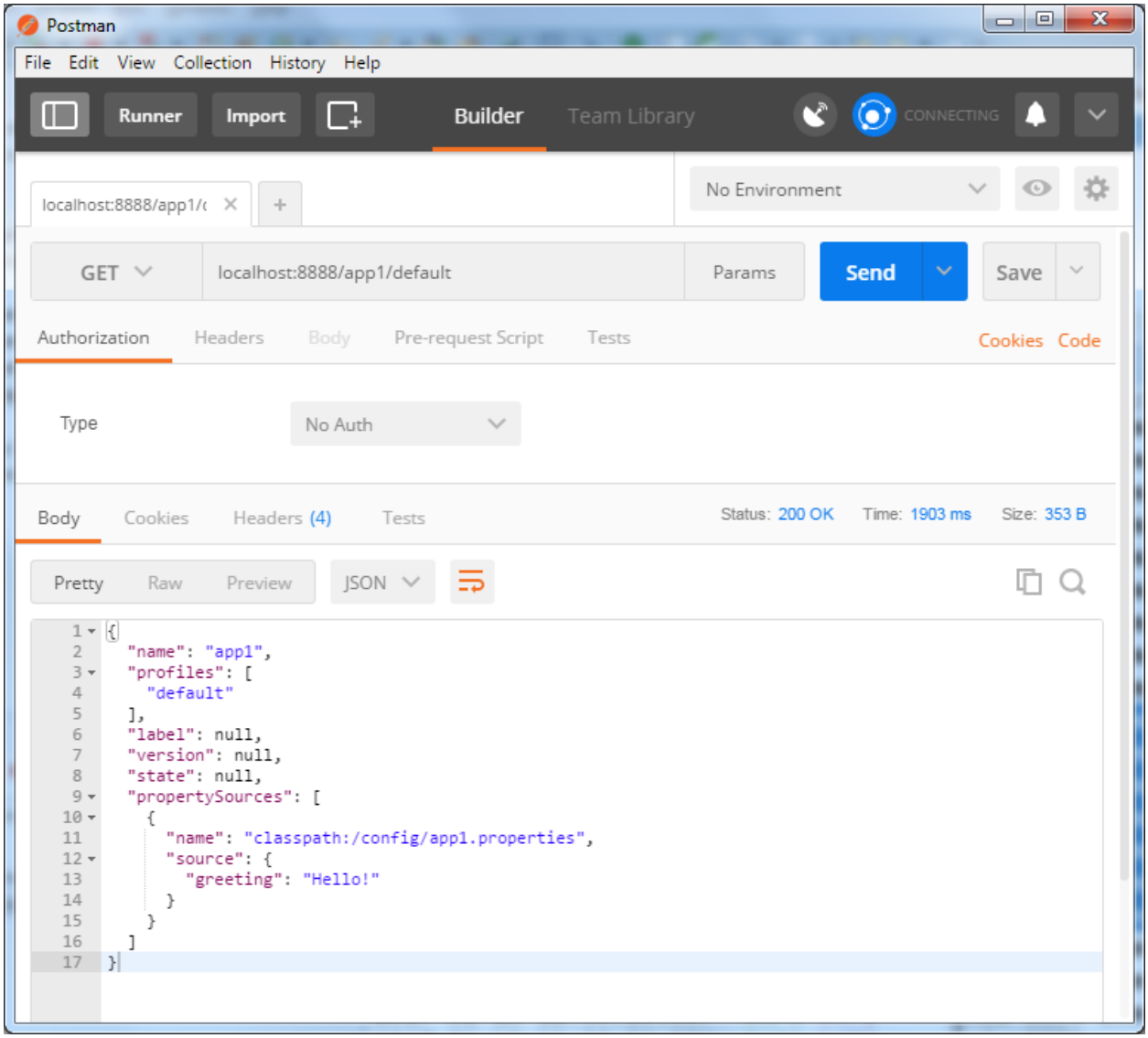
- Go to BootDashboard window, select your project cap-springcloud-m2-configserver and click run  icon.
- You can aslo run project by Right Click → Run As →Spring Boot App

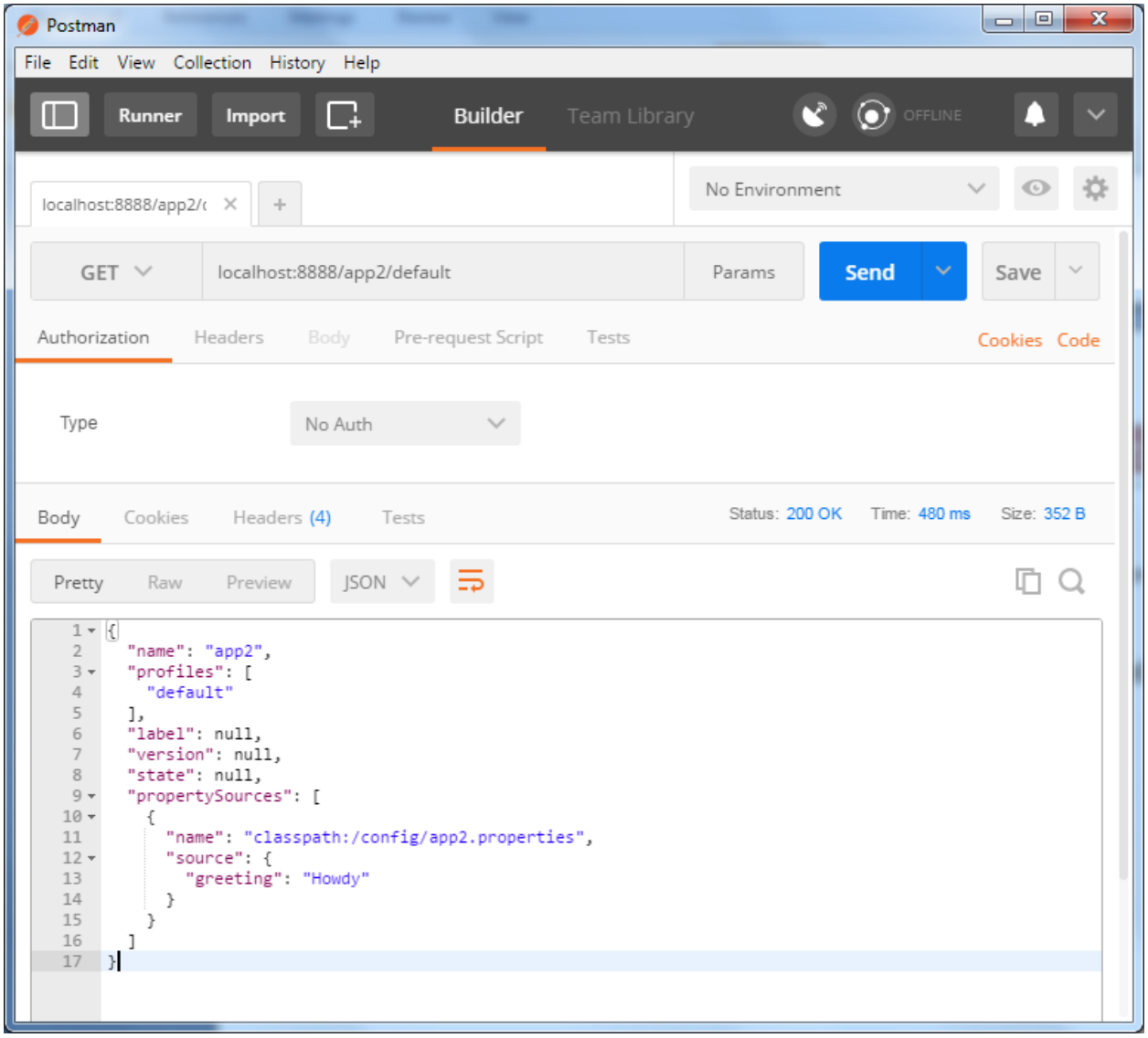
**Output:**

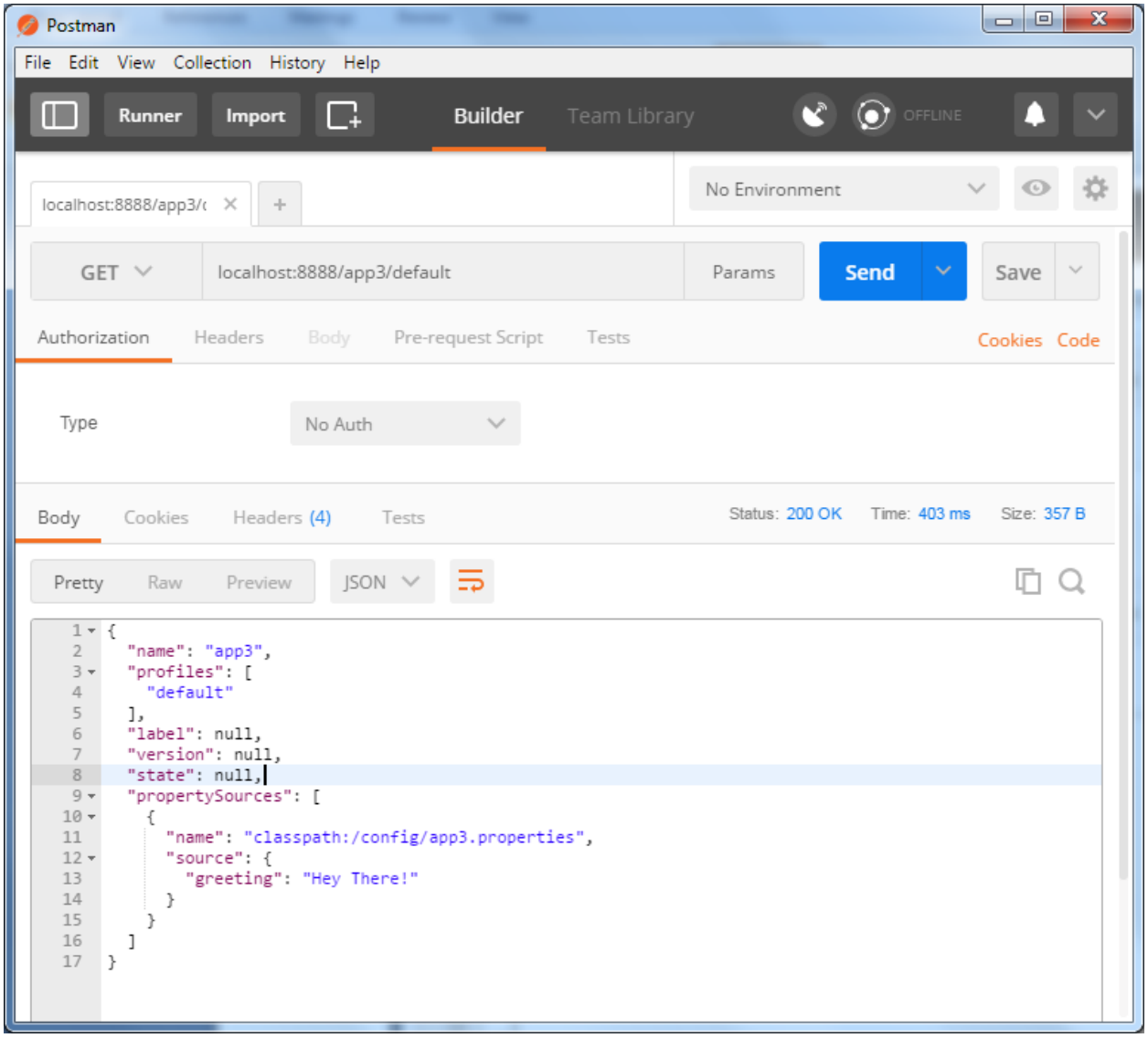
- You will log details in console window.

```
2017-01-10 12:35:36.994  INFO 5472 --- [        main] o.s.j.e.a.AnnotationMBeanExporter      : Located managed bean 'restartEndpoint': registering with JMX server as MBean
[org.springframework.cloud.context.restart:name=restartEndpoint,type=RestartEndpoint]
2017-01-10 12:35:37.012  INFO 5472 --- [        main] o.s.j.e.a.AnnotationMBeanExporter      : Located managed bean 'refreshScope': registering with JMX server as MBean
[org.springframework.cloud.context.scope.refresh:name=refreshScope,type=RefreshScope]
2017-01-10 12:35:37.022  INFO 5472 --- [        main] o.s.j.e.a.AnnotationMBeanExporter      : Located managed bean 'configurationPropertiesRebinder': registering with JMX server as MBean
[org.springframework.cloud.context.properties:name=configurationPropertiesRebinder,context=37c7595,type=ConfigurationPropertiesRebinder]
2017-01-10 12:35:37.040  INFO 5472 --- [        main] o.s.j.e.a.AnnotationMBeanExporter      : Located managed bean 'refreshEndpoint': registering with JMX server as MBean
[org.springframework.cloud.endpoint:name=refreshEndpoint,type=RefreshEndpoint]
2017-01-10 12:35:37.206  INFO 5472 --- [        main] o.s.c.support.DefaultLifecycleProcessor  : Starting beans in phase 0
2017-01-10 12:35:37.713  INFO 5472 --- [        main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8888 (http)
2017-01-10 12:35:37.725  INFO 5472 --- [        main] .CapSpringcloudM2ConfigserverApplication : Started CapSpringcloudM2ConfigserverApplication in 10.442 seconds (JVM running for 15.035)
```

- Now you can open Postman application, you will get below output




Postman
File Edit View Collection History Help
Runner
Import
Builder
Team Library
OFFLINE
localhost:8888/app2/(
No Environment
GET
localhost:8888/app2/default
Params
Send
Save
Authorization
Headers
Body
Pre-request Script
Tests
Cookies
Code
Type
No Auth
Body
Cookies
Headers (4)
Tests
Status: 200 OK
Time: 480 ms
Size: 352 B
Pretty
Raw
Preview
JSON
{
"name": "app2",
"profiles": [
"default"
],
"label": null,
"version": null,
"state": null,
"propertySources": [
{
"name": "classpath:/config/app2.properties",
"source": {
"greeting": "Howdy"
}
}
]
}




**Learning:**

- From the example we learnt how to configure local configuration files(app1,app2 and app3).

You have successfully completed this lab!

**LAB – 3**

***Create config Server for GitHub Files.***

**Steps:**

Open SpringToolSuite 3.8.1.Release IDE.

- File → New → Spring Starter Project . And create project name as cap-springcloud-m2-configserver-git with the following setup.
- In the next window add Web and Actuator Dependencies. And then click Finish. You will be getting one project **cap-springcloud-m2-configserver-git** in the project explorer window.
  - In the main class add **@EnableConfigServer** annotation.
  - Use git hub " https://github.com/caprepo/springCloud-MS.git " repository files for configuration.
    - You can also refer https://github.com/caprepo/springCloud-MS-Perf.git repository.

**CapSpringcloudM2ConfigserverGitApplication.java**

```
package cap.demo;

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

@SpringBootApplication
@EnableConfigServer
public class CapSpringcloudM2ConfigserverGitApplication {

	public static void main(String[] args) {

	SpringApplication.run(CapSpringcloudM2ConfigserverGitApplication.class,
args);

	}

}
```

**application.yml**

```
---
server:
  port: 8888
spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/caprepo/SpringCloud-MS.git
```

```
           #username: caprepo
           #password: caprepo123
          search-paths:
           - 'station*'
```

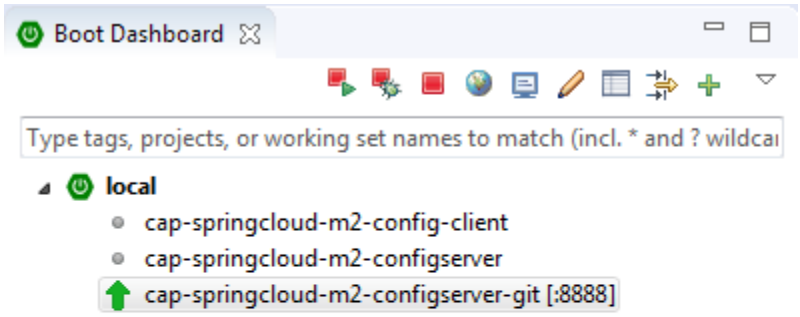
**pom.xml**

```
<?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>

      <groupId>cap.demo</groupId>
      <artifactId>demo</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <packaging>jar</packaging>

      <name>cap-springcloud-m2-configserver-git</name>
      <description>Spring Boot Project config Server with GIT</description>

      <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>1.4.3.RELEASE</version>
            <relativePath/> <!-- lookup parent from repository -->
      </parent>

      <properties>
            <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
            <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
            <java.version>1.8</java.version>
      </properties>

      <dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-actuator</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework.cloud</groupId>
                  <artifactId>spring-cloud-config-server</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-security</artifactId>
            </dependency>
```

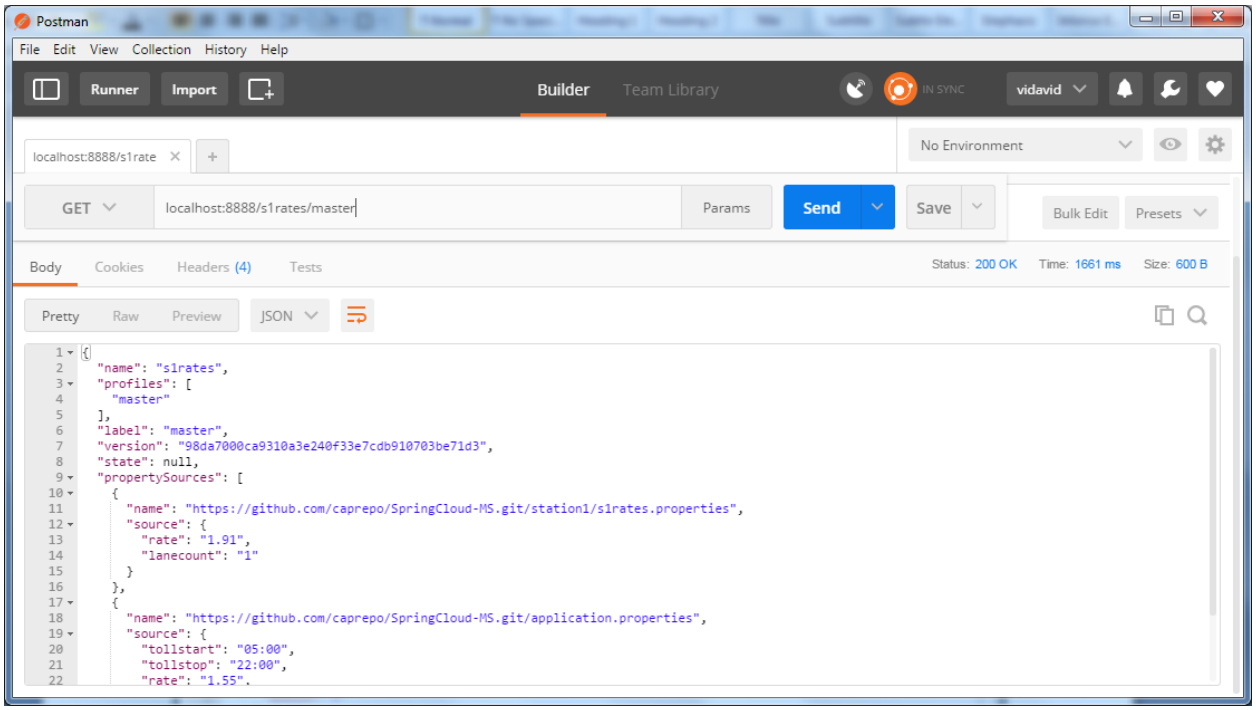
```
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<dependencyManagement>
		<dependencies>
			<dependency>
				<groupId>org.springframework.cloud</groupId>
				<artifactId>spring-cloud-dependencies</artifactId>
				<version>Camden.SR3</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>
```

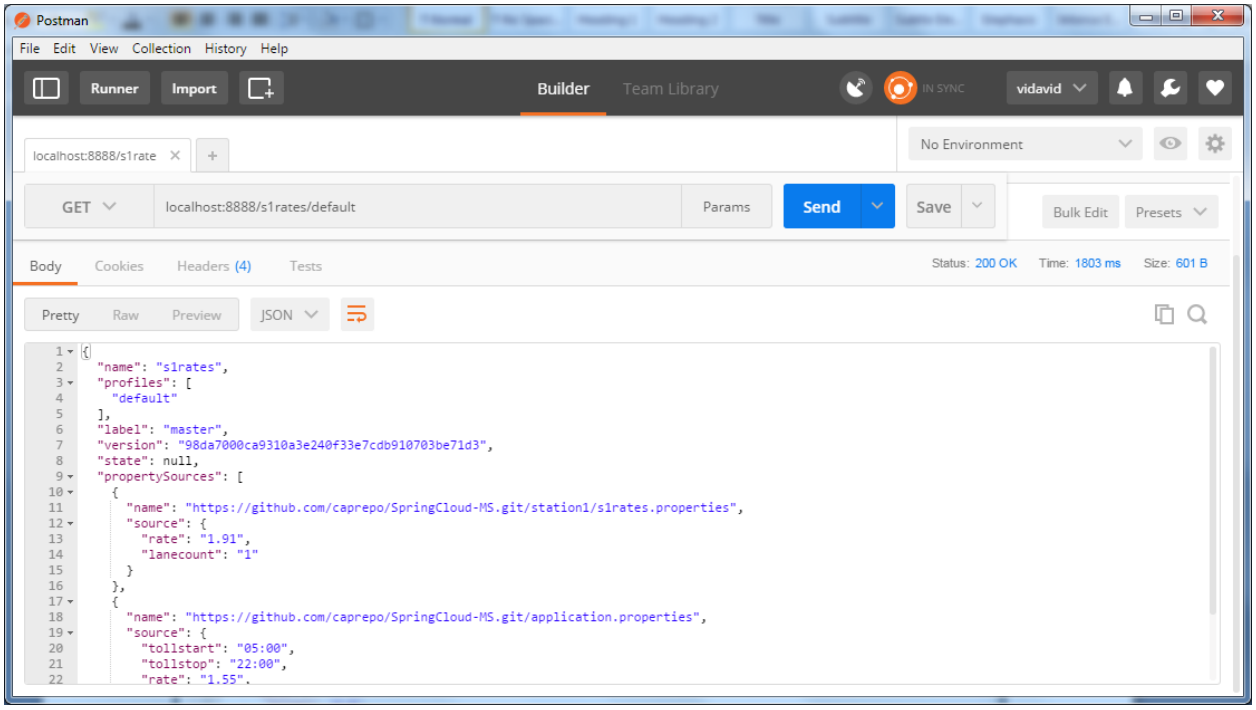
**Output:**

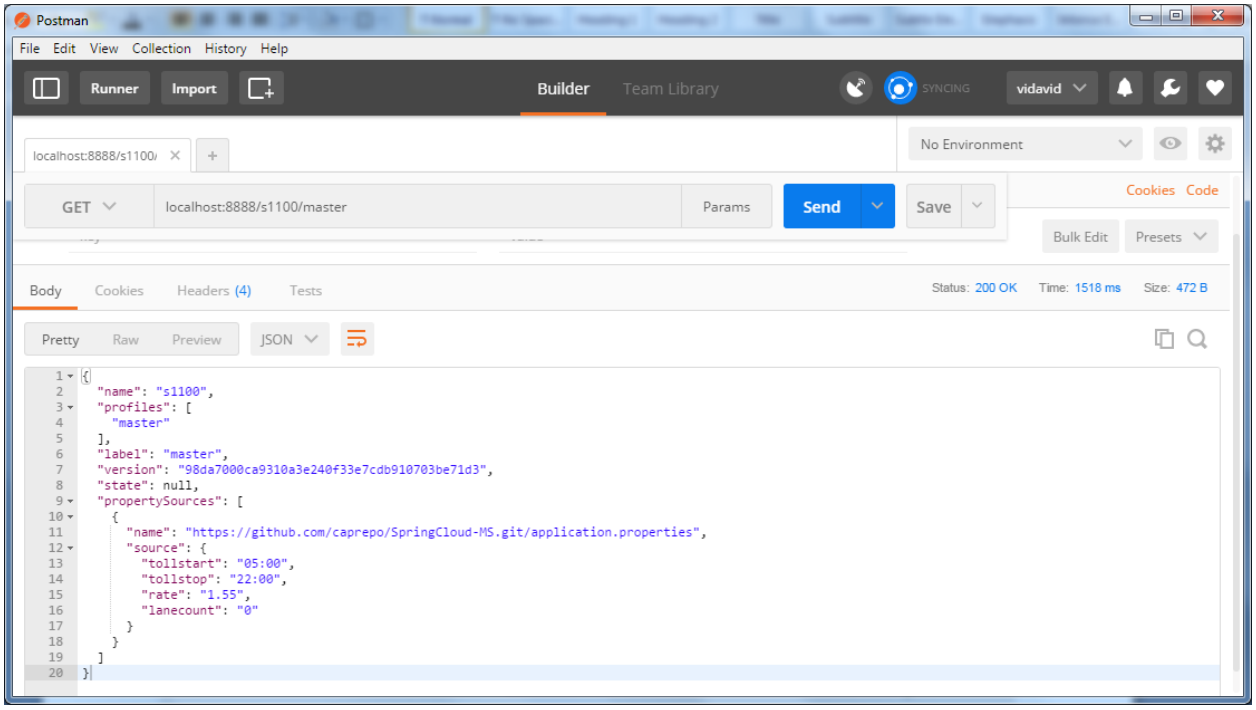
**Go to boot dashboard and start the server.**




**Once the server started , open your postman application**






**Learning:**

- From the example we learnt how to configure Spring Boot Config Server with Git Hub files.

You have successfully completed this lab!

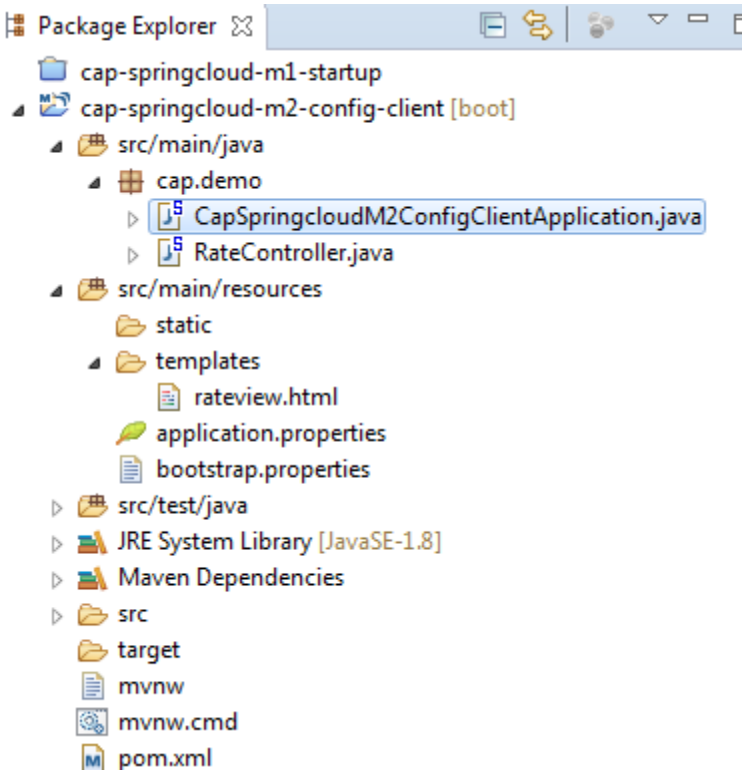
**LAB – 4**

***Create spring boot client application which will consume configuration from server which has been start in your last Lab.***

**Steps:**

Open SpringToolSuite 3.8.1.Release IDE.

- File → New → Spring Starter Project . And create project name as cap-springcloud-m2-configserver-git with the following setup.
- In the next window add Web ,Actuator, thymeleaf and Config-client and Dependencies. And then click Finish. You will be getting one project **cap-springcloud-m2-config-client** in the project explorer window.
  - Use git hub " https://github.com/caprepo/springCloud-MS.git " repository files for configuration.
    - You can also refer https://github.com/caprepo/springCloud-MS-Perf.git repository.




**CapSpringcloudM2ConfigClientApplication.java**

```
package cap.demo;

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

```
@SpringBootApplication
public class CapSpringcloudM2ConfigClientApplication {

	public static void main(String[] args) {
		SpringApplication.run(CapSpringcloudM2ConfigClientApplication.class, args);
	}
}
```

RateController.java

```
package cap.demo;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class RateController {
	@Value("${rate}")
	String rate;

	@Value("${lanecount}")
	String lanecount;

	@Value("${tollstart}")
	String tollstart;

	@RequestMapping("/rate")
	public String getRate(Model map){
		map.addAttribute("rateamount", rate);
		map.addAttribute("lanes",lanecount);
		map.addAttribute("tollstart",tollstart);

		return "rateview";
	}
}
```

rateview.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
```

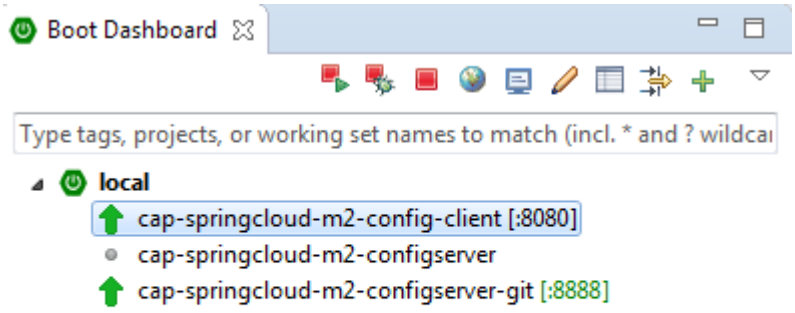
```
<title> Microservices Training: Config Client</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous"></link>
</head>
<body>
<div class="row">
        <div class="col-md-2"></div>
        <div class="col-md-8">
                <h1>Microservices Training: Spring Cloud Config Client</h1>
          <p th:text="'Your rate is: ' + ${rateamount} + ', number of lanes is ' + ${lanes} + ', toll start
time is ' + ${tollstart} + ' and encrypted value is '+ ${connstring} +'!'" />
        </div>
  <div class="col-md-2"></div>
</div>
</body>
</html>
```

application.properties

```
spring.application.name=s1rates
spring.profiles.active=default
spring.cloud.config.uri=http://localhost:8888
```

**Output:**

- Start the server first and start the client application as well.




- Once started check the below URL in the browser:

- http://localhost:8080/rate

- Modify the application.properties file and include bootstrap.properties file as mentioned below:

**application.properties**

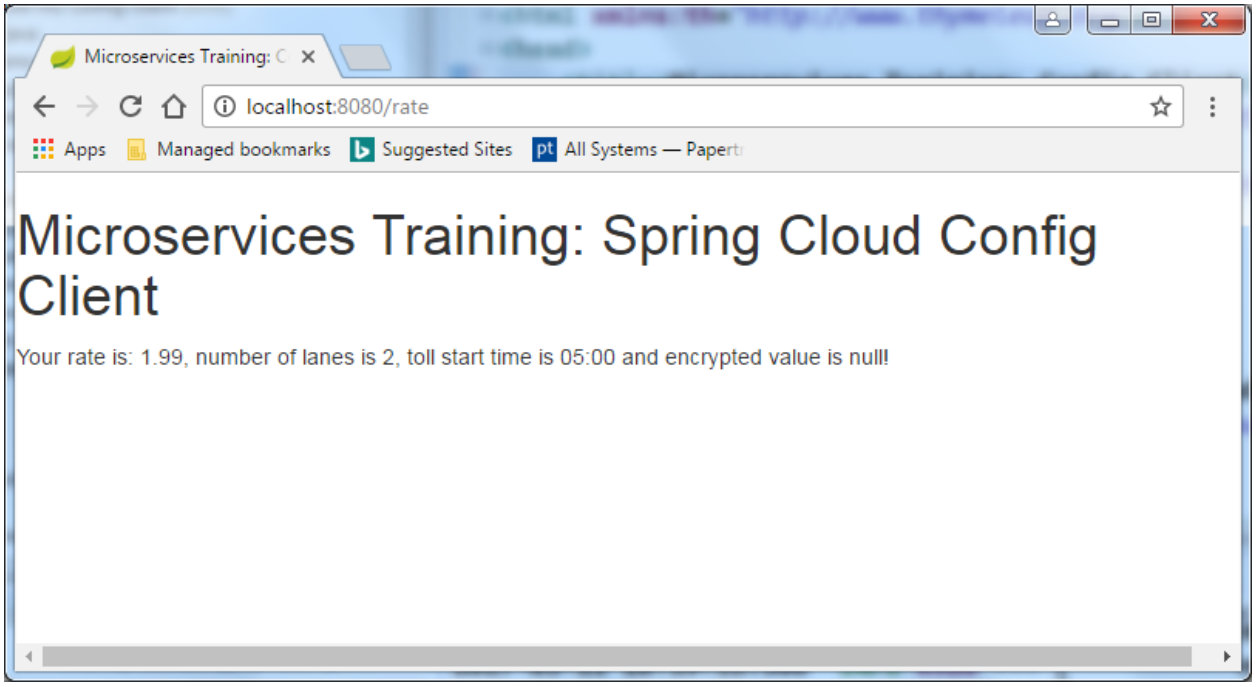
spring.application.name=s1rates

**bootstrap.properties**

spring.profiles.active=qa

spring.cloud.config.uri=http://localhost:8888

- Restart your application and refresh your browser




**Learning:**

- From the example we learnt how to access server data from client application.

You have successfully completed this lab!

**LAB – 5**

***Enable spring security with your previous assignment.***

**Steps:**

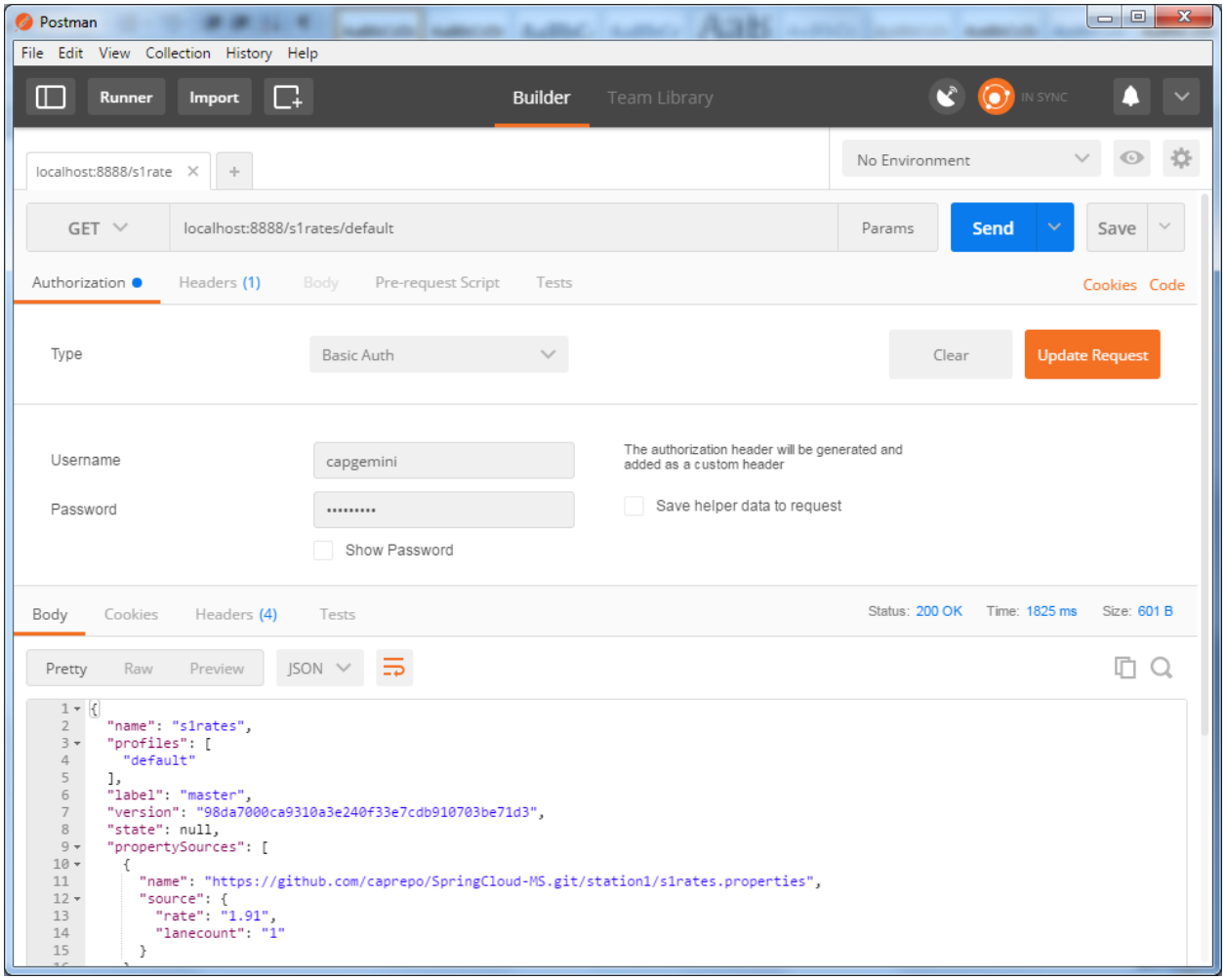
- Open **cap-springcloud-m2-configserver-git** project.
- Add spring security dependency in pom.xml file.

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

- And modify your application.yml file as mentioned below:

```
---
server:
  port: 8888
security:
 basic:
  enabled: true
 user:
  name: capgemini
  password: capgemini
spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/caprepo/SpringCloud-MS.git
           #username: caprepo
           #password: caprepo123
          search-paths:
           - 'station*'
```

- Start your server. Open post man app, under authorization mention username and password and send the request.

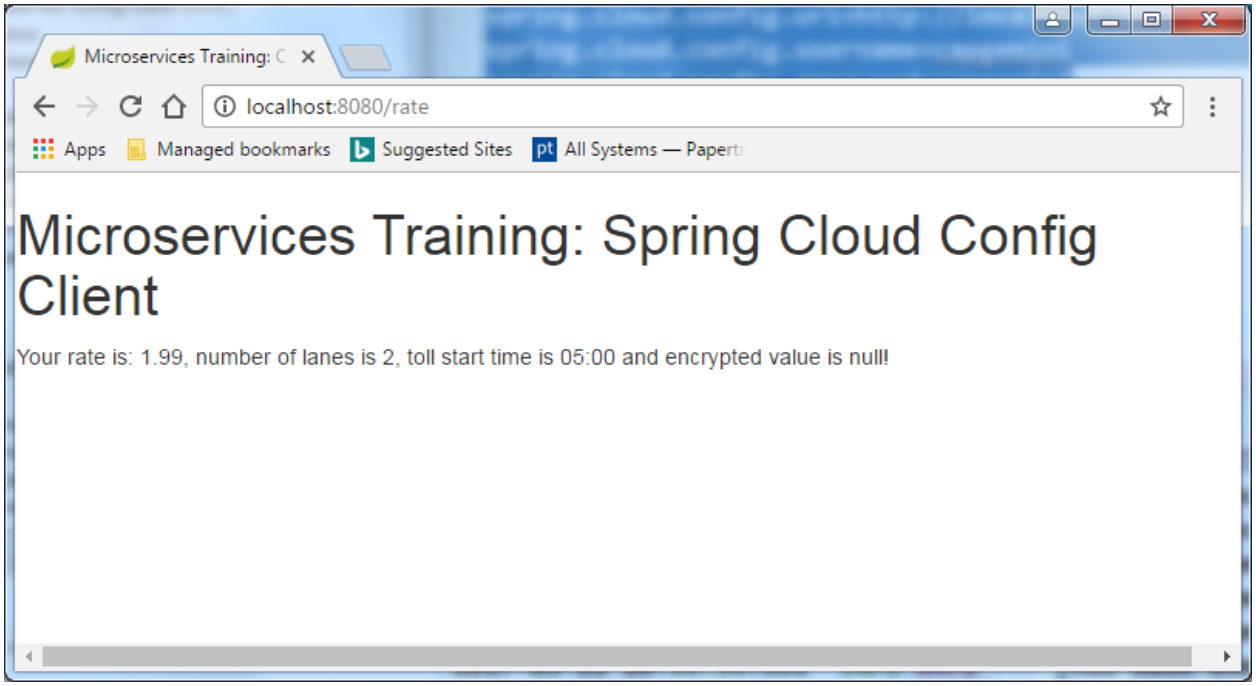



- Similarly open cap-springcloud-m2-config-client open boorstrap.properties file and add the followings:

```
spring.profiles.active=qa
spring.cloud.config.uri=http://localhost:8888
spring.cloud.config.username=capgemini
spring.cloud.config.password=capgemini
```

- Restart the application and check your output in the browser window.

**Output:**




**Learning:**

- From the example we learnt how to configure secured Spring Boot Config Server and client.

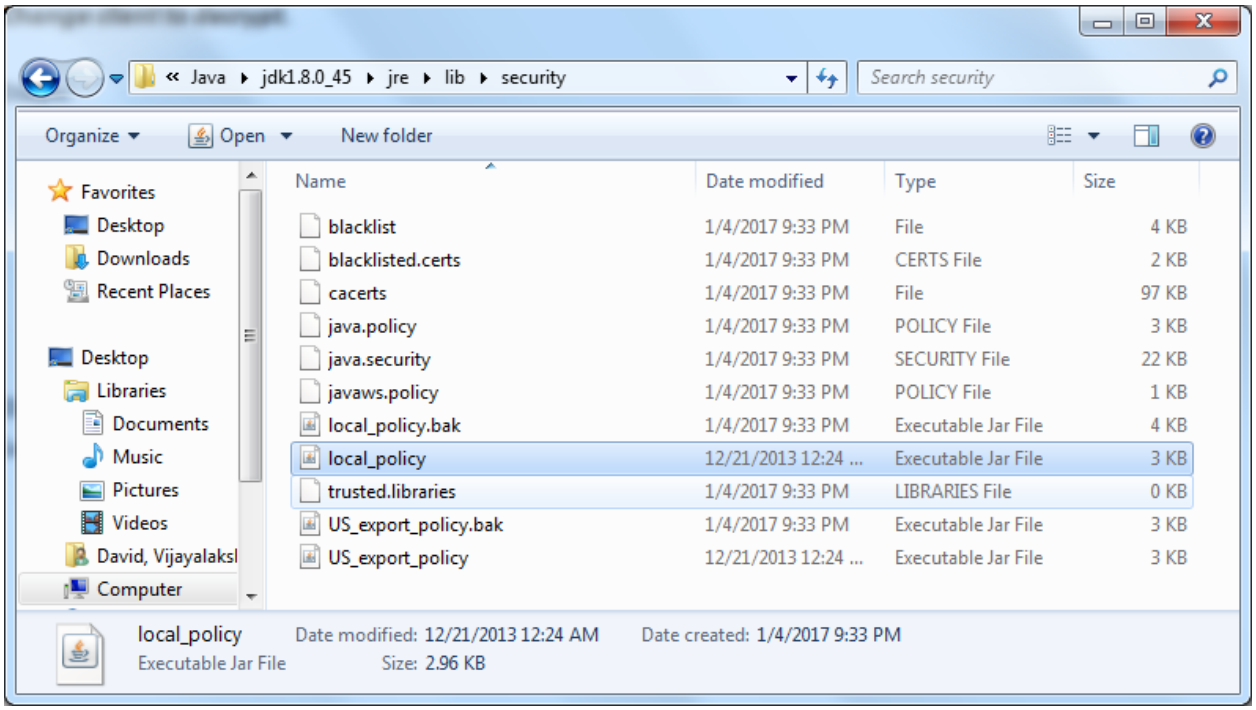
You have successfully completed this lab!

## LAB – 6

***Download full-strength JCE***
***Add key to bootstrap file***
***Generate encrypted value and add to properties file***
***Retrieve configuration via API***
***Test client app with server-side decrypted value***
***Update server to require client-side decryption***
***Change client to decrypt.***

### Steps:

- Use previously created cap-springcloud-m2-configserver-git and cap-springcloud-m2-config-client projects.
- Download the JCE from the below link:
  - http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html
- Copy local_policy and US_export_policy into your jdk → jre → security folder



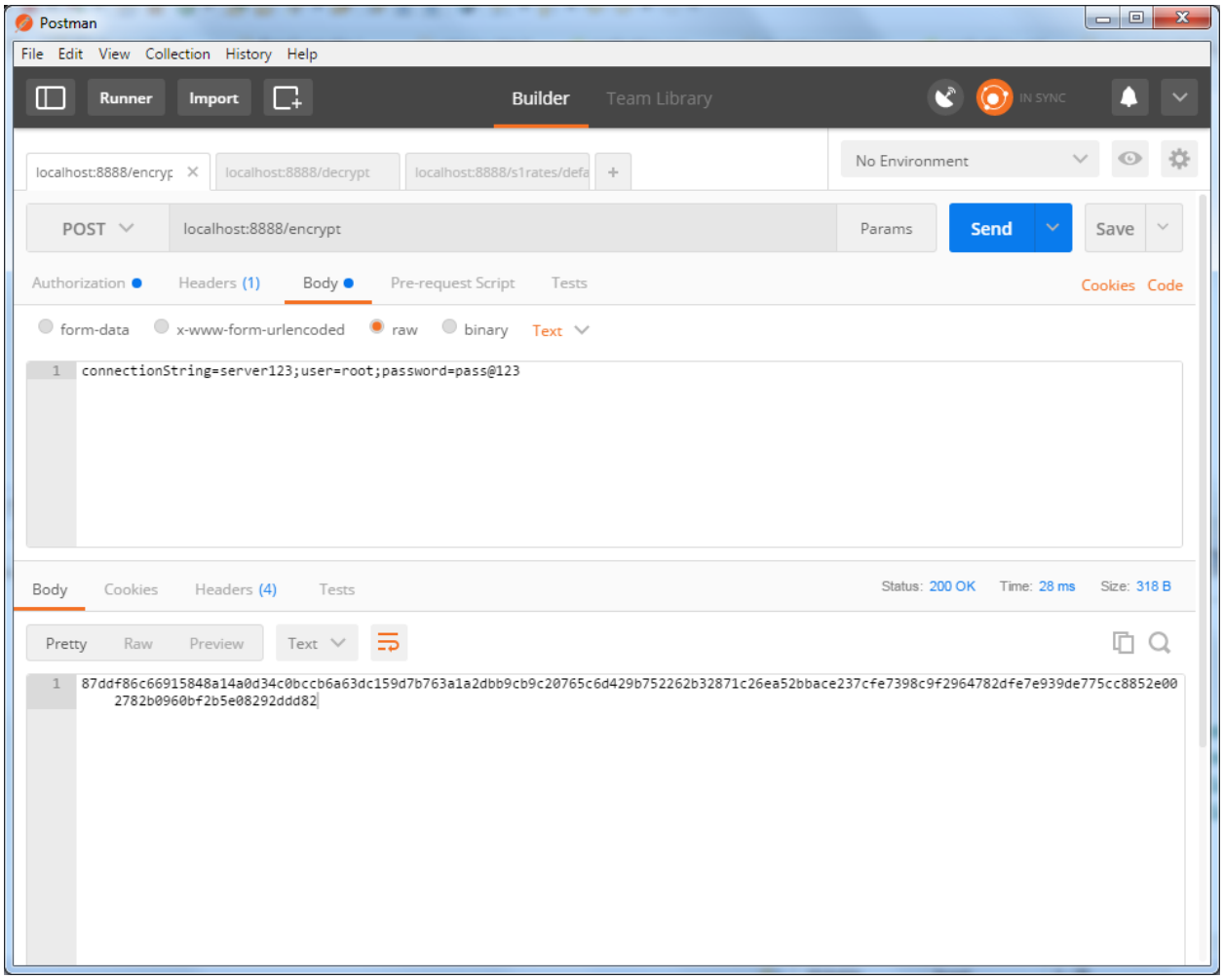

- In the server project add bootstrap.properties file as mentioned below:

**boorstrap.properties**

encrypt.key=ABCDEFGHIJKLMNOPQRSTUVWXZY

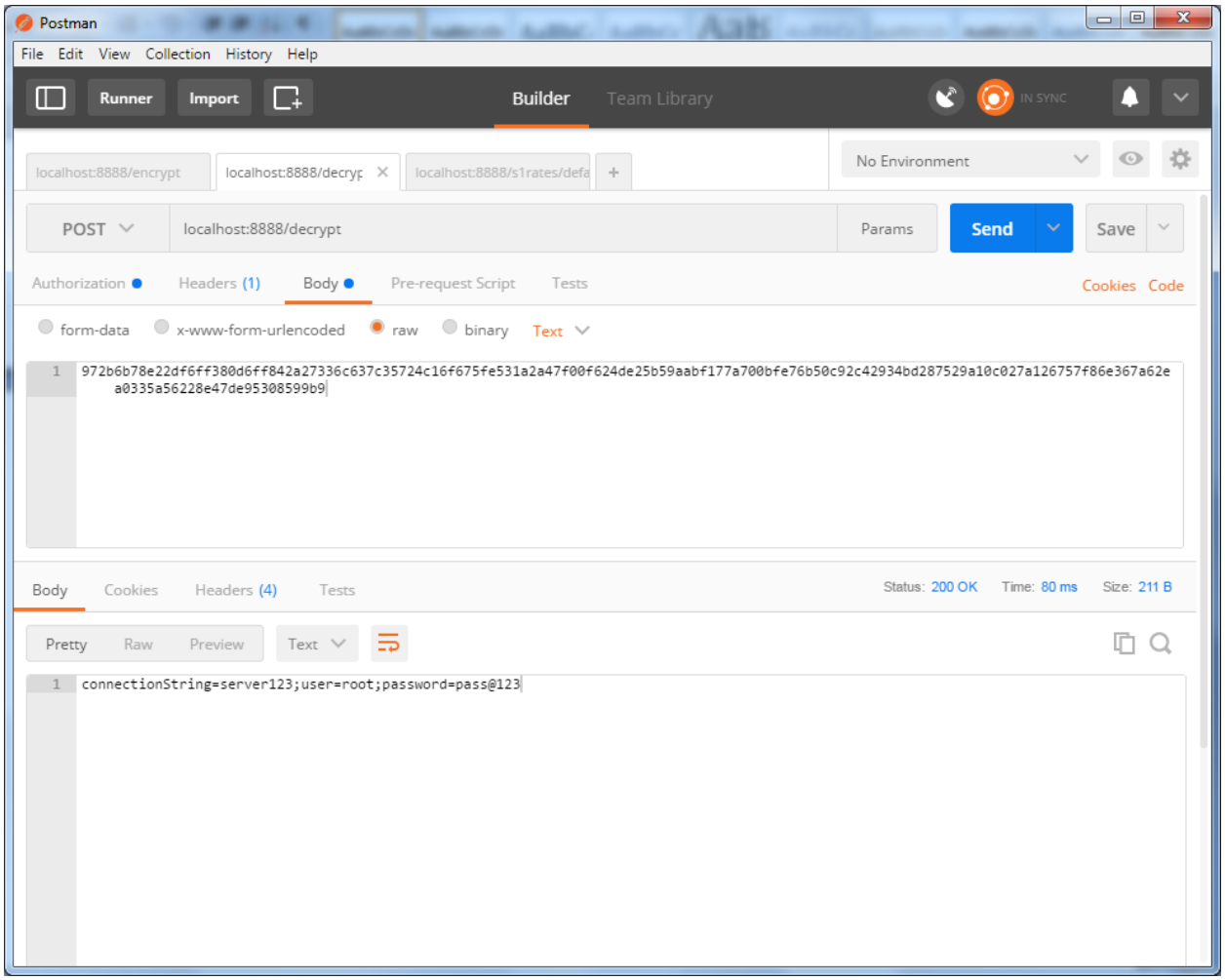
**Output:**

- Mention the below link in postman
    - c



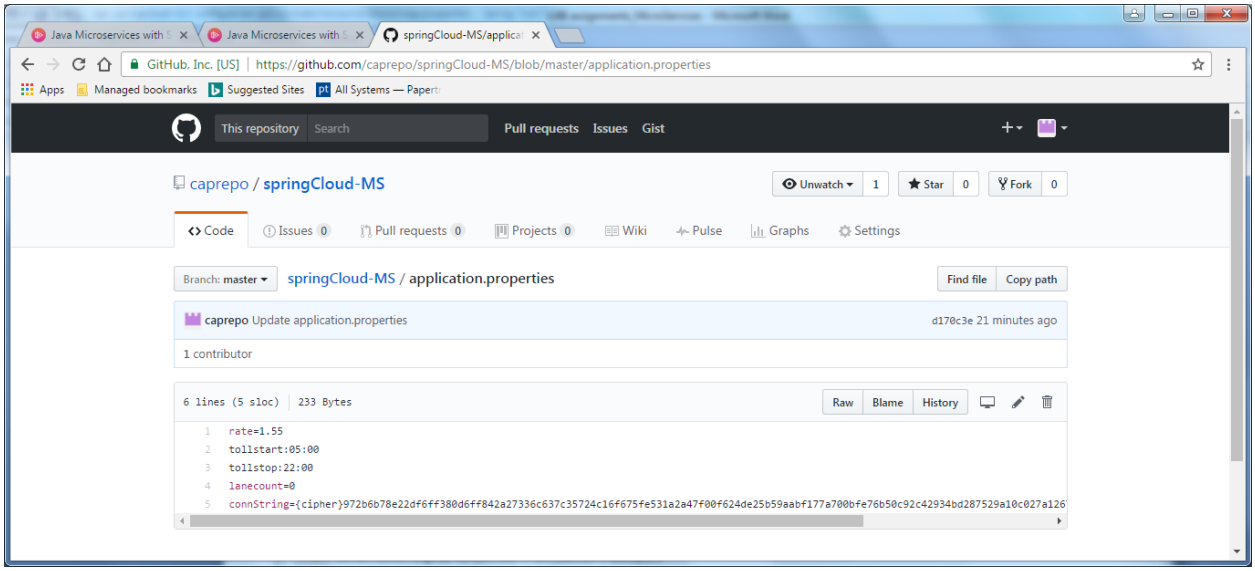
- Similarly mention the decrypt link:
  - http://localhost:8888/decrypt
    - Provide Basic Auth → username :capgemini; password→ capgemini
    - Under Body → mention encrypted value

**972b6b78e22df6ff380d6ff842a27336c637c35724c16f675fe531a2a47f00f624de25b59aabf177a700bfe76b50c92c42934bd287529a10c027a126757f86e367a62ea0335a56228e47de95308599b9**




- Now open git hub repository called “springCloud-MS” modify application.properties file with encrypted connection String :

connString={cipher}972b6b78e22df6ff380d6ff842a27336c637c35724c16f675fe531a2a47f00f624de25b59aabf177a700bfe76b50c92c42934bd287529a10c027a126757f86e367a62ea0335a56228e47de95308599b9




- Now open your client application change bootstrap.properties, RateController.java as mentioned below:

**bootstrap.properties**

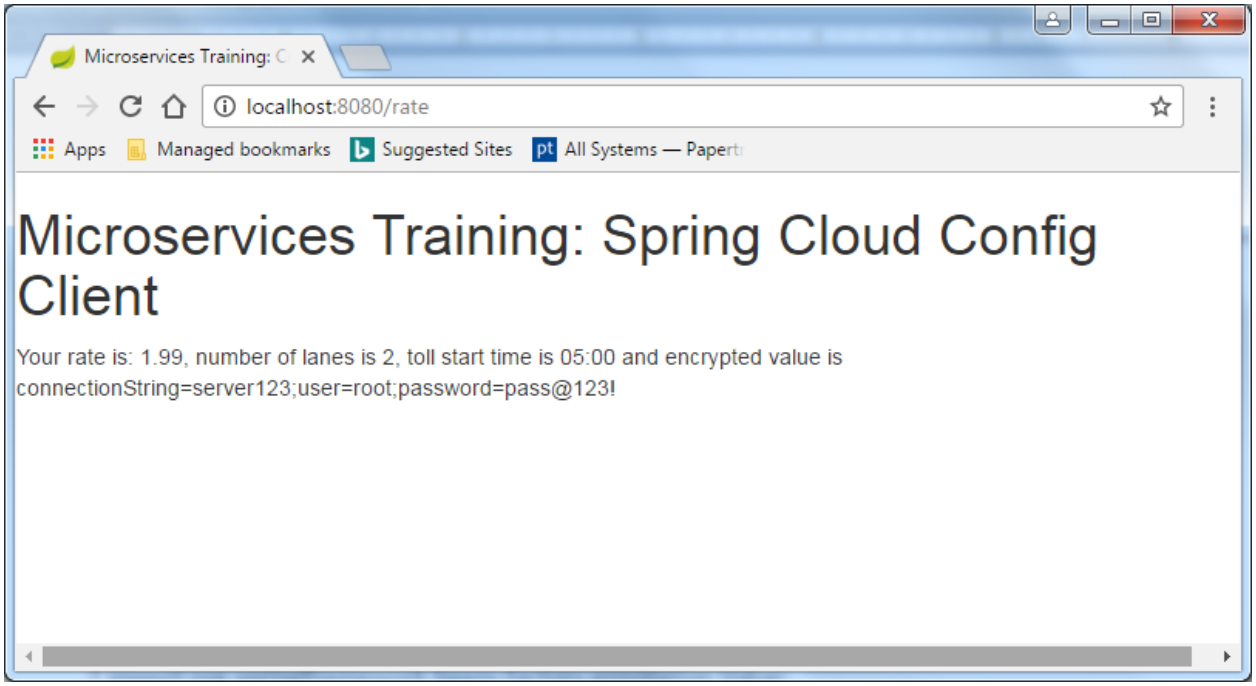
```
spring.profiles.active=qa
spring.cloud.config.uri=http://localhost:8888
spring.cloud.config.username=capgemini
spring.cloud.config.password=capgemini
encrypt.key=ABCDEFGHIJKLMNOPQRSTUVWXZY
```

**RateController.java**

```
package cap.demo;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

@Controller

public class RateController {

        @Value("${rate}")
```

```
	String rate;


	@Value("${lanecount}")
	String lanecount;


	@Value("${tollstart}")
	String tollstart;


	@Value("${connString}")
	String connString;


	@RequestMapping("/rate")
	public String getRate(Model map){
		map.addAttribute("rateamount", rate);
		map.addAttribute("lanes",lanecount);
		map.addAttribute("tollstart",tollstart);
		map.addAttribute("connString", connString);
		return "rateview";
	}


}
```

- Finally check your browser you will get updated value:




**Learning:**

- From the example we learnt how to configure Spring Boot Config Server with Git Hub files.

You have successfully completed this lab!

**LAB – 7**

***Add RefreshScope to controller***
***Start server and client apps***
***Change a property in GitHub***
***Trigger client refresh***
***See new value without requiring a restart.***

**Steps:**

- Do the following changed in client application
- Add actuator dependency in pom.xml

**pom.xml**

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

**bootstrap.prperties**

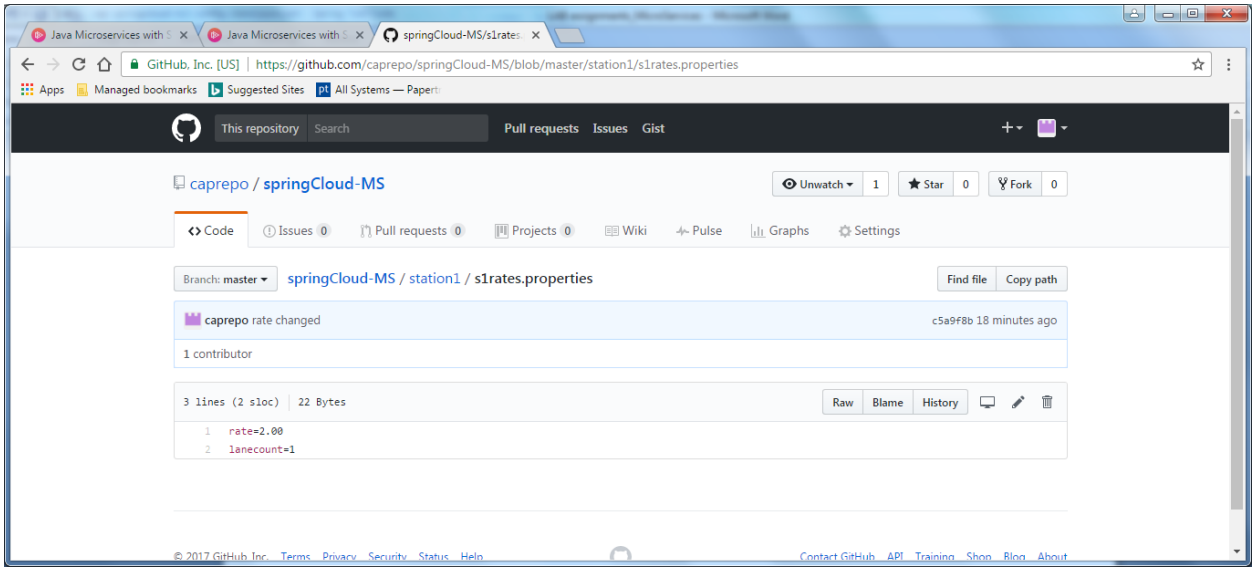
```
spring.profiles.active=default
spring.cloud.config.uri=http://localhost:8888
spring.cloud.config.username=capgemini
spring.cloud.config.password=capgemini
encrypt.key=ABCDEFGHIJKLMNOPQRSTUVWXZY
```

**RateController.java**

```
package cap.demo;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RefreshScope
public class RateController {
    @Value("${rate}")
    String rate;

    @Value("${lanecount}")
    String lanecount;
```

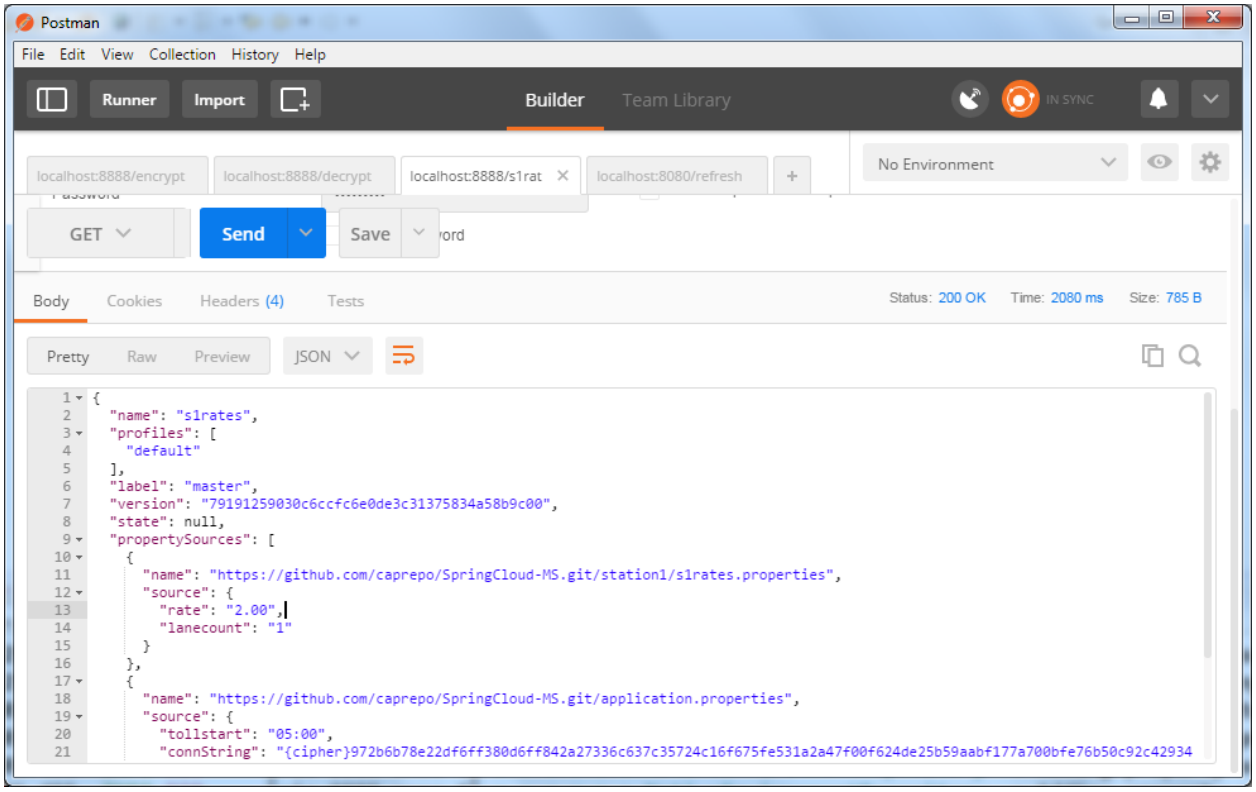
```
        @Value("${tollstart}")
        String tollstart;

        @Value("${connString}")
        String connString;

        @RequestMapping("/rate")
        public String getRate(Model map){
                map.addAttribute("rateamount", rate);
                map.addAttribute("lanes",lanecount);
                map.addAttribute("tollstart",tollstart);
                map.addAttribute("connString", connString);
                return "rateview";
        }

}
```

- Now start server and client application.
- Open Git hub modify the springCloud-MS repository → station1 →s1rates.properties file , rate as 2.00

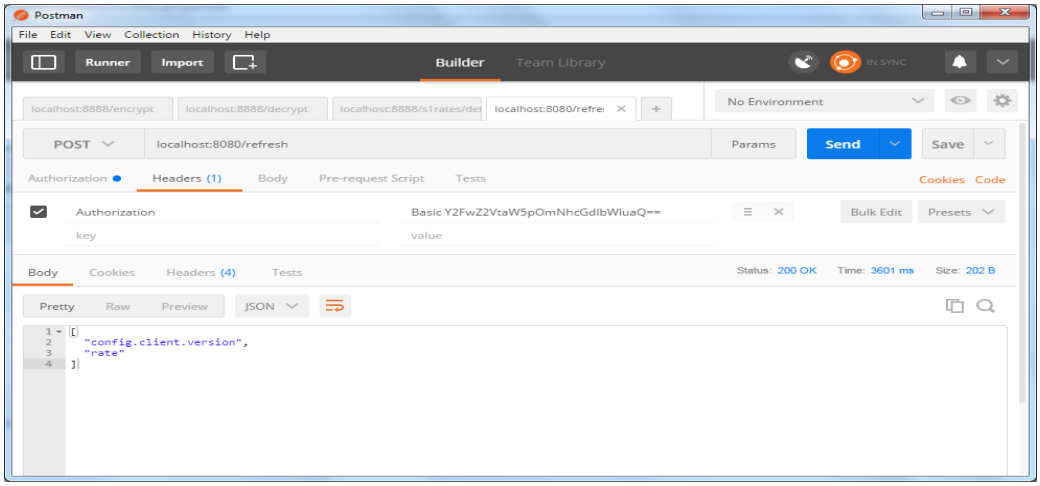



**Output:**

- Once change the value open post man
- Enter http://localhost:8888/s1rates/default you will get updated rate as 2.00




- If you open browser you will not get the changes. Again open postman with url
  - http://localhost:8080/refresh
- And then open your browser do refresh. Will get the updated rate.




**Learning:**

- From the example we learnt how to auto refresh the client page when the configuration changes.

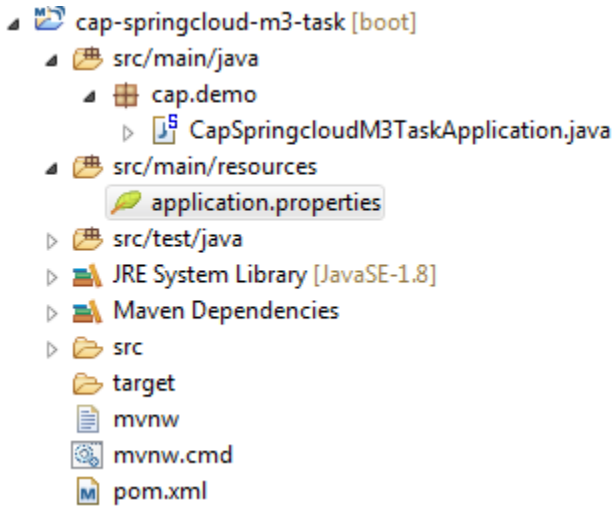
You have successfully completed this lab!

**LAB – 8**

***Create a new Spring Boot project for Toll Processing task***
***Annotate primary class***
***Add task logic as commandLineRunner***
***Execute task and observe results***

**Steps:**

- Create new Spring boot project in the name of cap-springcloud-m3-task.
- Add cloud task under cloud core while creating the project.




- Add the below code:

**CapSpringcloudM3TaskApplication.java**

```
package cap.demo;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.task.configuration.EnableTask;
import org.springframework.context.annotation.Bean;


@SpringBootApplication
@EnableTask
public class CapSpringcloudM3TaskApplication {

	public static void main(String[] args) {
		SpringApplication.run(CapSpringcloudM3TaskApplication.class, args);
	}

	@Bean
	public TollProcessingTask tollProcessingTask(){
```

```
		return new TollProcessingTask();
	}

	public class TollProcessingTask implements CommandLineRunner{

		@Override
		public void run(String... strings) throws Exception {
			//parameters stationid, license plate ,timestamp
			if(null!=strings){
				System.out.println("Parameter Length is:" +
strings.length);

				String stationId=strings[1];
				String licensePlate=strings[2];
				String timestamp=strings[3];

				System.out.println("Station Id is :" + stationId +", plate
is: "+  licensePlate +
					", Timestamp: " + timestamp);
			}
		}

	}
}
```

**application.properties:**

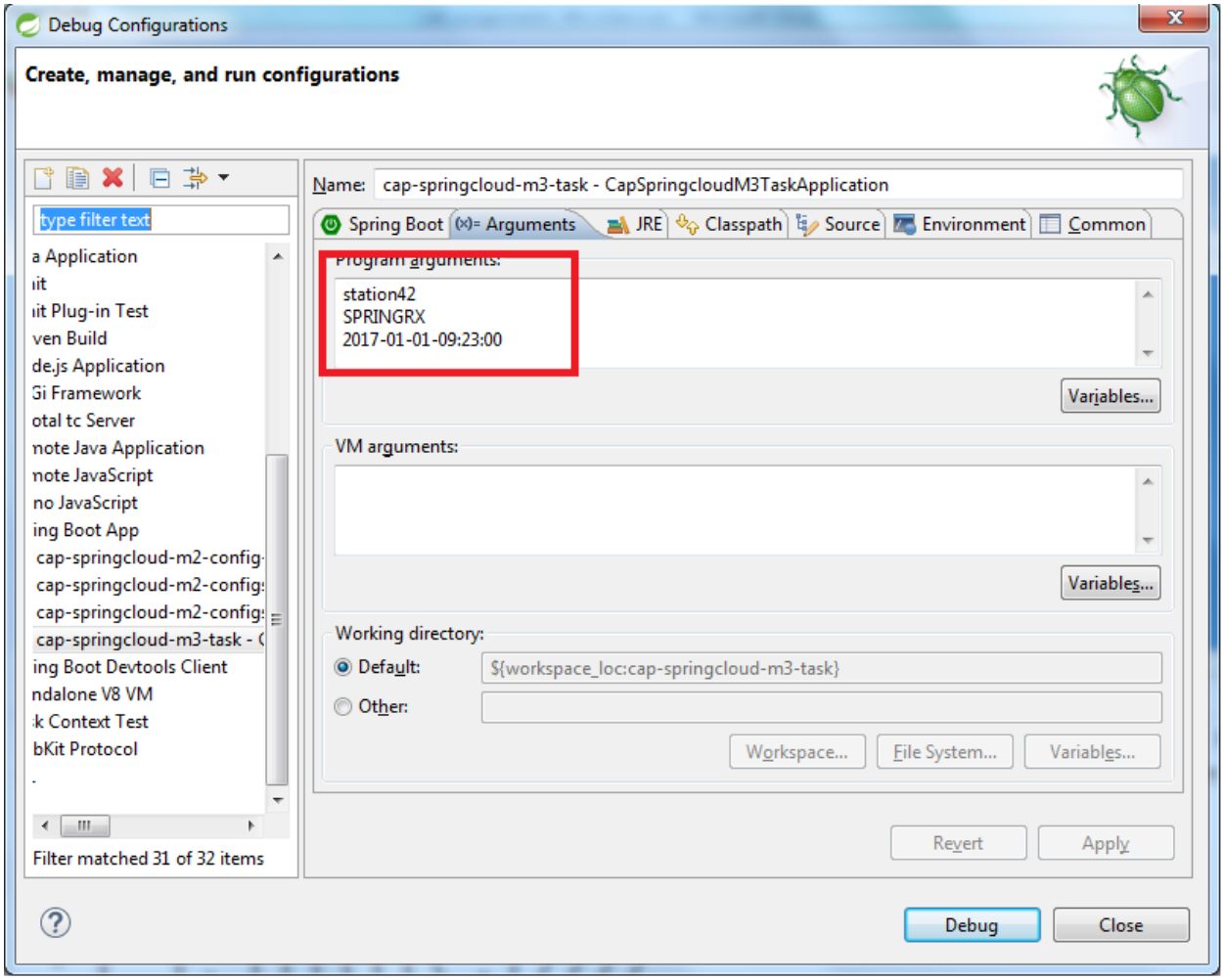
```
spring.application.name=Toll Processor
logging.level.org.springframework.cloud.task=DEBUG
```

**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>

	<groupId>cap.demo</groupId>
	<artifactId>cap-springcloud-m3-task</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>cap-springcloud-m3-task</name>
	<description>Spring Boot task Demo</description>

	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>1.4.5.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
```

```
	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-task</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-task-dependencies</artifactId>
				<version>1.1.2.RELEASE</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>
```

**Run this with the below parameters:**




**Output:**

```
 .   ____          _            __ _ _
/\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.4.5.RELEASE)

2017-03-11 22:11:40.388  INFO 7596 --- [           main] c.demo.CapSpringcloudM3TaskApplication   : Starting CapSpringcloudM3TaskApplication on LIN73000499 with PID 7596 (D:\vidavid\MyWork\2017\plural_MS_Demo\cap-springcloud-m3-task\target\classes started by vidavid in D:\vidavid\MyWork\2017\plural_MS_Demo\cap-springcloud-m3-task)
2017-03-11 22:11:40.392  INFO 7596 --- [           main] c.demo.CapSpringcloudM3TaskApplication   : No active profile set, falling back to default profiles: default
2017-03-11 22:11:40.446  INFO 7596 --- [           main] s.c.a.AnnotationConfigApplicationContext : Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@5038d0b5: startup date [Sat Mar 11 22:11:40 IST 2017]; root of context hierarchy
```

```
2017-03-11 22:11:41.550 DEBUG 7596 --- [          main] o.s.c.t.c.SimpleTaskConfiguration       : Using
org.springframework.cloud.task.configuration.DefaultTaskConfigurer TaskConfigurer
2017-03-11 22:11:41.926  INFO 7596 --- [          main] o.s.j.e.a.AnnotationMBeanExporter       : Registering beans for JMX exposure on startup
2017-03-11 22:11:41.932  INFO 7596 --- [          main] o.s.c.support.DefaultLifecycleProcessor  : Starting beans in phase 0
2017-03-11 22:11:41.933 DEBUG 7596 --- [          main] o.s.c.t.r.support.SimpleTaskRepository   : Creating: TaskExecution{executionId=0,
exitCode=null, taskName='Toll Processor', startTime=Sat Mar 11 22:11:41 IST 2017, endTime=null, exitMessage='null', errorMessage='null',
arguments=[--spring.output.ansi.enabled=always, station42, SPRINGRX, 2017-01-01-09:23:00]}
Parameter Length is:4
Station Id is :station42, plate is: SPRINGRX, Timestamp: 2017-01-01-09:23:00
2017-03-11 22:11:41.944 DEBUG 7596 --- [          main] o.s.c.t.r.support.SimpleTaskRepository   : Updating: TaskExecution with executionId=0
with the following {exitCode=0, endTime=Sat Mar 11 22:11:41 IST 2017, exitMessage='null', errorMessage='null'}
2017-03-11 22:11:41.944  INFO 7596 --- [          main] s.c.a.AnnotationConfigApplicationContext : Closing
org.springframework.context.annotation.AnnotationConfigApplicationContext@5038d0b5: startup date [Sat Mar 11 22:11:40 IST 2017];
root of context hierarchy
2017-03-11 22:11:41.945  INFO 7596 --- [          main] o.s.c.support.DefaultLifecycleProcessor  : Stopping beans in phase 0
2017-03-11 22:11:41.946  INFO 7596 --- [          main] o.s.j.e.a.AnnotationMBeanExporter       : Unregistering JMX-exposed beans on
shutdown
2017-03-11 22:11:41.947  INFO 7596 --- [          main] c.demo.CapSpringcloudM3TaskApplication   : Started
CapSpringcloudM3TaskApplication in 2.003 seconds (JVM running for 2.746)
```

**Learning:**

- From the example we learnt how to auto refresh the client page when the configuration changes.

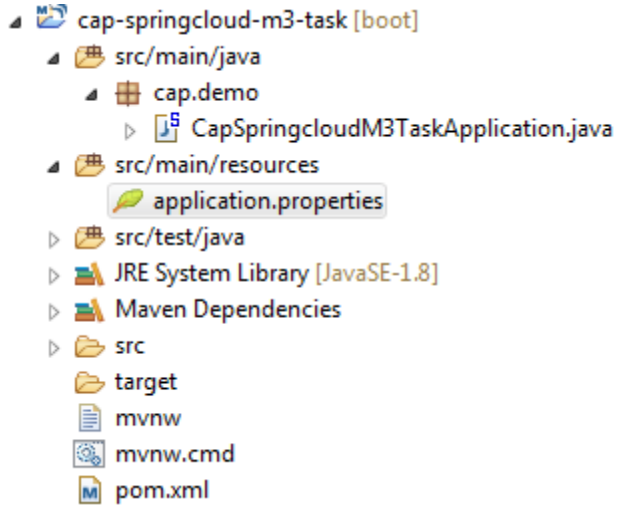
You have successfully completed this lab!

**LAB – 9**

***create MySQL database***
***Add MYSQL dependencies in PM***
***Update applciation properties***
***Call task and observe stored results***
***Execute task and observe results***

**Steps:**

- Create new Spring boot project in the name of **cap-springcloud-m3-task**.
- Add cloud task under cloud core while creating the project.




- Create Database called "**tasklogs**" in mysql.
- Add the below code:

**CapSpringcloudM3TaskApplication.java**

```
package cap.demo;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.task.configuration.EnableTask;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
@EnableTask
public class CapSpringcloudM3TaskApplication {
```

```
	public static void main(String[] args) {
		SpringApplication.run(CapSpringcloudM3TaskApplication.class, args);
	}

	@Bean
	public TollProcessingTask tollProcessingTask(){
		return new TollProcessingTask();
	}

	public class TollProcessingTask implements CommandLineRunner{

		@Override
		public void run(String... strings) throws Exception {
			//parameters stationid, license plate ,timestamp
			if(null!=strings){
				System.out.println("Parameter Length is:" + strings.length);
				if(strings.length > 0){
				String stationId=strings[1];
				String licensePlate=strings[2];
				String timestamp=strings[3];

				System.out.println("Station Id is :" + stationId +", plate is: "+
licensePlate +
					", Timestamp: " + timestamp);
				}
			}

			System.out.println("Task completed.");
		}

	}
}
```

**application.properties**

```
spring.application.name=Toll Processor
logging.level.org.springframework.cloud.task=DEBUG

spring.datasource.url=jdbc:mysql://localhost:3306/tasklogs
spring.datasource.username=root
spring.datasource.password=admin
```

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

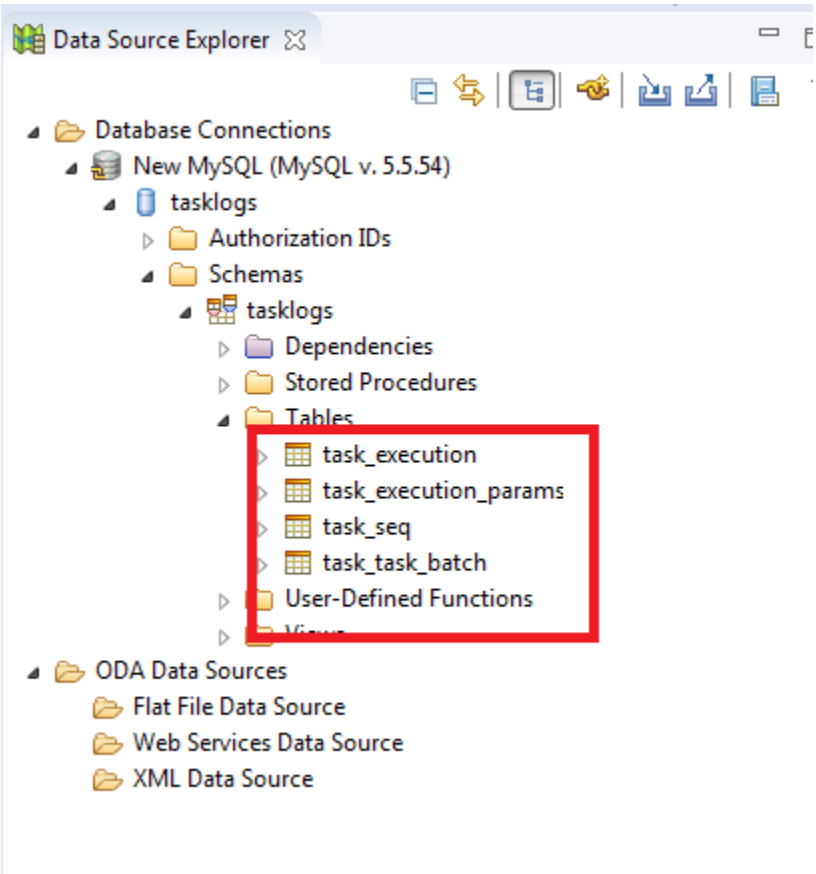
**pom.xml**

```
<?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>

	<groupId>cap.demo</groupId>
	<artifactId>cap-springcloud-m3-task</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>cap-springcloud-m3-task</name>
	<description>Spring Boot task Demo</description>

	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>1.4.3.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-task-starter</artifactId>
			<version>1.0.1.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
```

```
                <scope>runtime</scope>
            </dependency>

            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-jdbc</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>Brixton.SR6</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>
```

**Output:**

- Run your application. It will create all supporting schema as mentioned below:




**Learning:**

- From the example we learnt how to store task in database.

You have successfully completed this lab!

**LAB – 7**

***Add RefreshScope to controller***

**Steps:**

**Output:**

**Learning:**

- From the example we learnt how to auto refresh the client page when the configuration changes.

You have successfully completed this lab!