Workshop

MICROSERVICE

WITH

SPRING BOOT

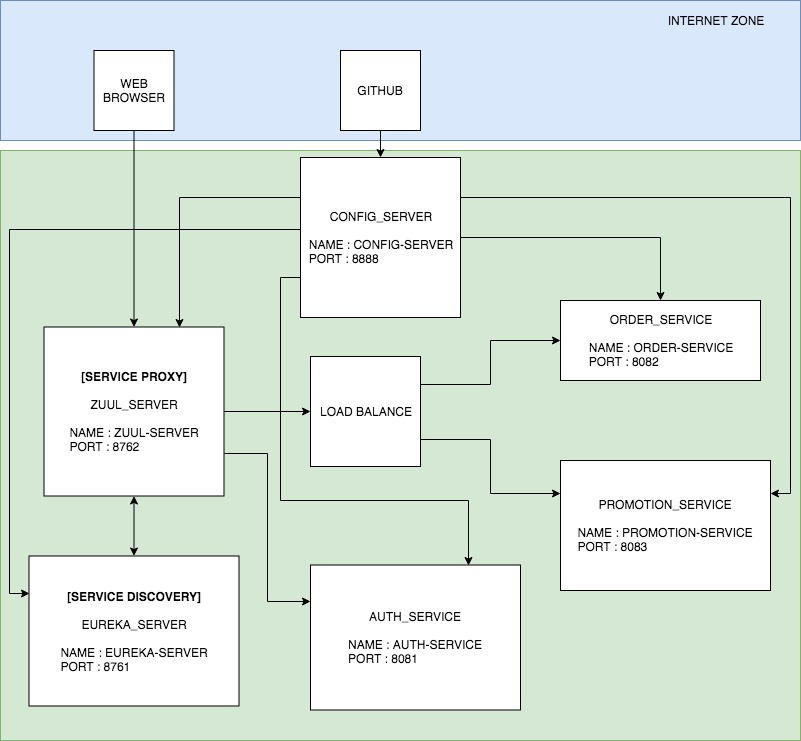
by

PnP Solution

[www.pnpsw.com](http://www.pnpsw.com)

facebook.com/pnpsolution

System Overview



โครงสร้าง Folder ของ Project

* work-shop
  + micro\_service\_config
  + config\_server
  + eureka\_server
  + zuul\_server
  + auth\_service
  + order\_service
  + promotion\_service

DOCKER

**WORKSHOP#1** สร้าง docker container ชื่อ my\_db สำหรับทำเป็น mysql server

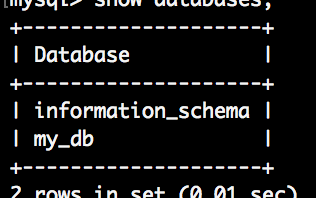
**วิธีการ**

* สร้าง mysql docker container ด้วยคำสั่งดังนี้ \*\*comand ชุดเดียวกัน\*\*

docker run -d -p 3306:3306 --name my\_db -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=my\_db -e MYSQL\_USER=my\_db -e MYSQL\_PASSWORD=my\_db mysql:5.5.60



* เมื่อ run เสร็จแล้วให้ทดสอบหลังการติดตั้งดังนี้
  + docker exec -it my\_db bash
* เมื่อเข้าไปใน prompt ของ my\_db container แล้วให้ทดลองเข้าไปดูใน mysql db ด้วยคำสั่งดังนี้
  + mysql my\_db -u my\_db -p
  + ใส่ password เป็น my\_db
* หลังจากนั้นให้ใช้คำสั่ง ดังนี้เพื่อดูว่ามี database ที่เราระบุสร้างไว้หรือไม่ ด้วยคำสั่งดังนี้
  + show databases;
  + ผลลัพธ์จะได้ดังนี้



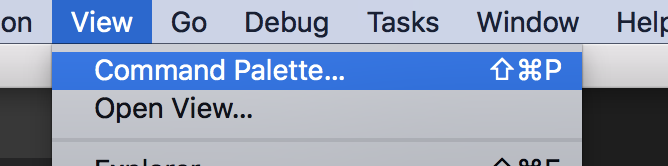
AUTH SERVICE

**WORKSHOP#1** สร้าง project ชื่อ auth\_service โดยมีคุณสมบัติ ดังนี้

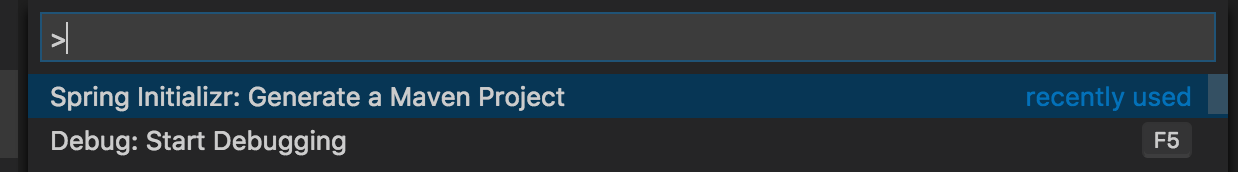
* group id = com.business
* artifect id = auth
* dependencies ดังนี้
  + web
  + actuator
  + jpa (sql)
  + mysql หรือ h2
  + devtools

**วิธีการ**

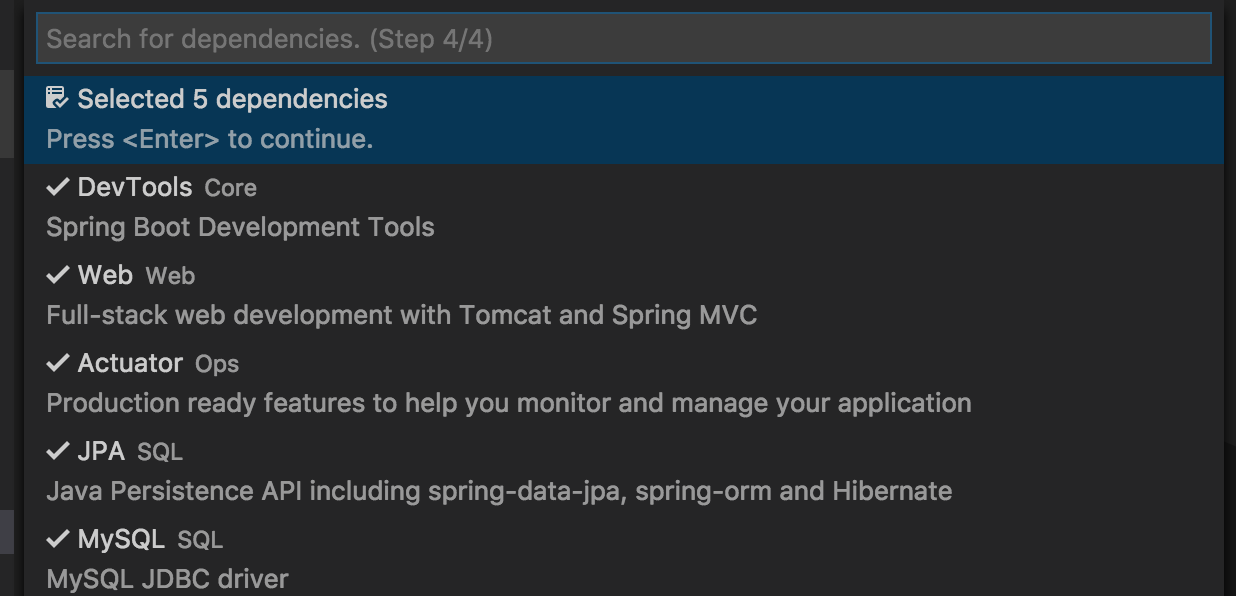
* เข้า Program Visual Studio Code แล้ว ไปที่ เมนู View > Command Palette… ดังภาพ



* เลือก Spring Initializr: Generage a Maven Project ดังภาพ



* ในขึ้นตอนการเลือก Dependencies ให้เลือกดังภาพ



**WORKSHOP#2** ตั้งค่าการเชื่อมต่อ database ไปยัง mysql หรือ h2

**วิธีการ**

* เปิด file application.properties แล้วใส่ข้อมูลดังนี้

## สำหรับ mysql

spring.datasource.url=jdbc:mysql://localhost:3306/my\_db

spring.datasource.username=my\_db

spring.datasource.password=my\_db

spring.datasource.driver=com.mysql.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

## สำหรับ H2

spring.h2.console.enabled=true

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

\*\*หมายเหตุ\*\*

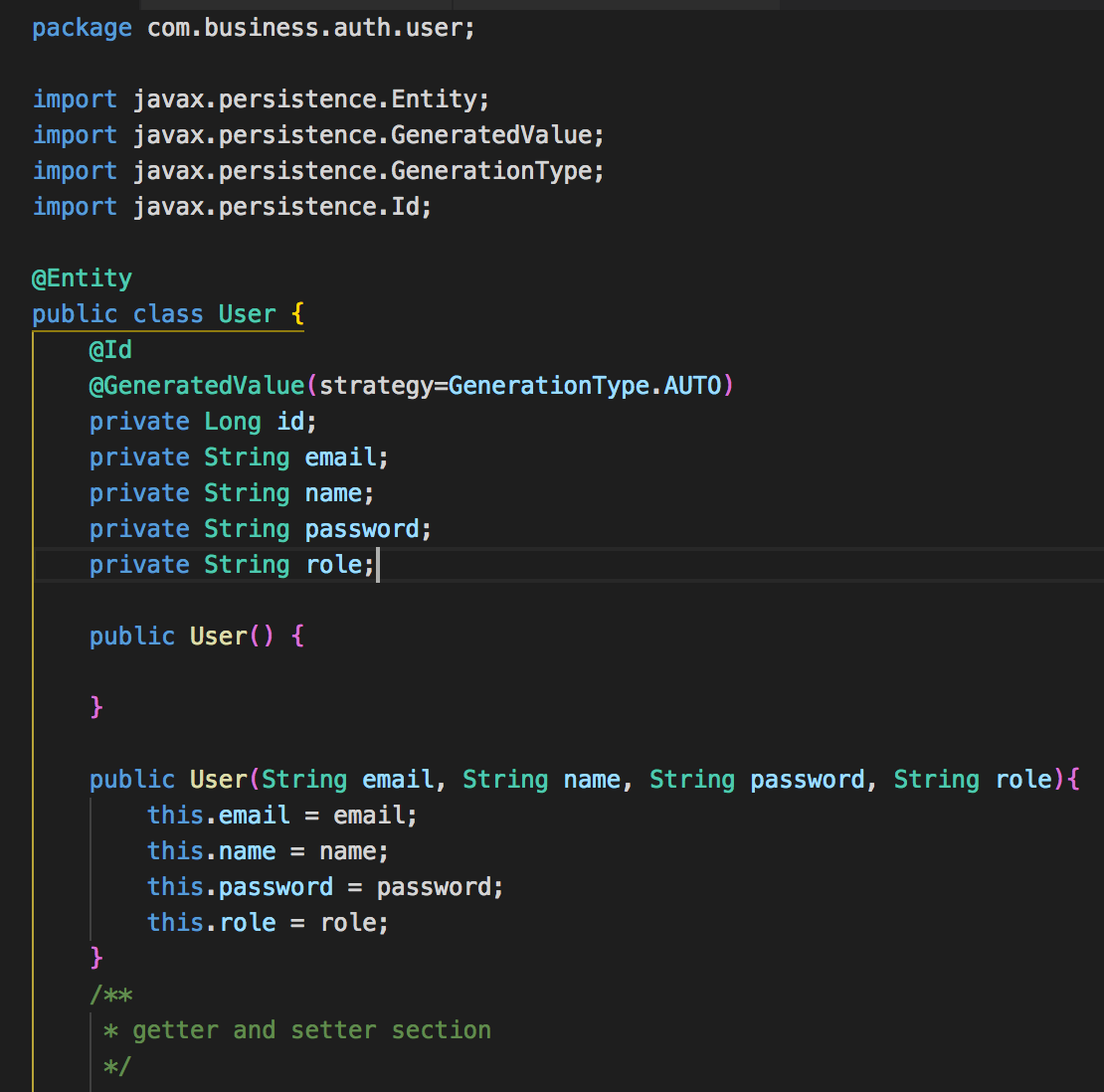
* spring.jpa.database-platform คือ driver สำหรับ hibernate ใช้ในการ generate sql statement
* spring.jpa.hibernate.ddl-auto คือ mode ของการสร้าง table มี 3 mode คือ
  + create สร้าง table ใหม่เสมอเมื่อ start project (ข้อมูลเก่าๆ หาย)
  + update ปรับปรุงโครงสร้าง table ถ้ามีเปลี่ยนแปลง (ข้อมูลไม่หาย)
  + none ไม่สร้าง table
  + validate ตรวจสอบความเปลี่ยนแปลง ไม่สร้าง table
  + create-drop เหมือน create
* spring.jpa.show-sql ให้แสดง sql statement ใน console

**WORKSHOP#3** สร้าง Entity ชื่อ User โดยมี field ดังนี้

* Long id เป็น Auto generate key
* String email
* String password
* String name
* String role

**วิธีการ**

* สร้าง Class ชื่อ User ภายใต้ package com.business.auth.user



\*\* หมายเหตุ\*\* ต้องสร้าง method set, get ให้ครบทุก field ด้วยนะครับ

**WORKSHOP##** สร้าง Repository ของ Entity User

**วิธีการ**

* สร้าง interface ชื่อ UserRepository ภายใต้ package com.business.auth.user และ extends interface CrudRepository ดังนี้

package com.business.auth.user;

import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User, Long> {

}

\*\*หมายเหตุ\*\*

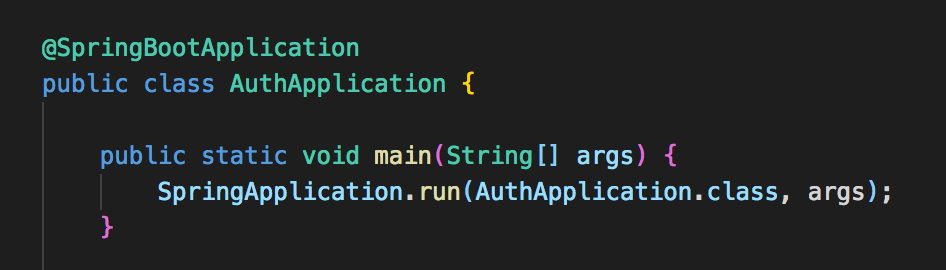
User คือชื่อ Entity Class

Long คือ Type ของ Id ของ Entity Class (แนะนำว่าให้ใช้ Long)

**WORKSHOP##** เปลี่ยนชื่อ Class DemoApplication เป็น AuthApplication

**วิธีการ**

* เปลี่ยนชื่อ File จาก DemoApplication.java เป็น AuthApplication.java
* แก้ไขชื่อ Class จาก DemoApplication เป็น AuthApplication ดังภาพ



* แก้ไข name ใน file pom.xml ดังภาพ



**WORKSHOP##** สร้าง Bean เพื่อทำการ initial data

**วิธีการ**

* เพิ่ม code Bean เข้าไปใน AuthApplication ดังนี้

@Bean

public CommandLineRunner initialUserData(UserRepository repository) {

return (args) -> {

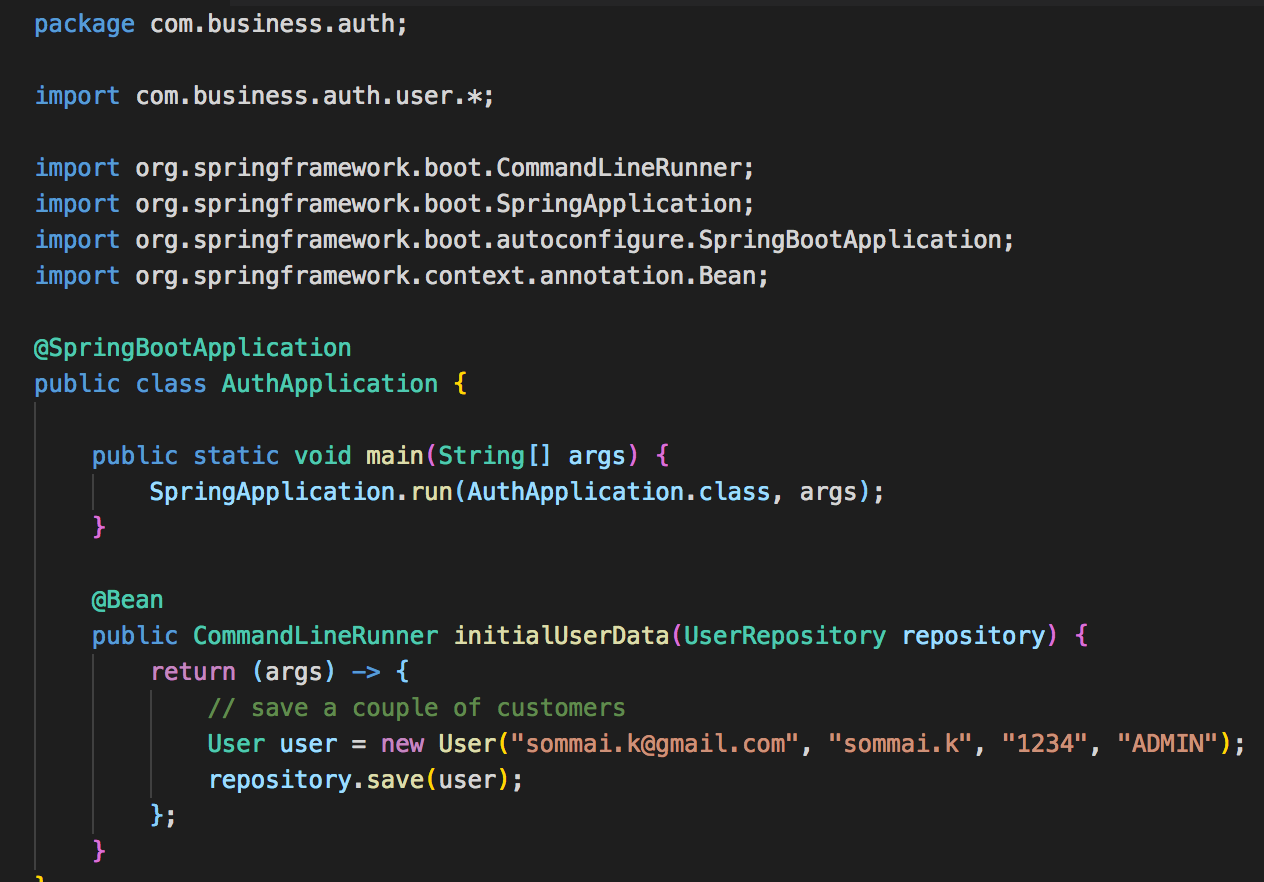
User user = new User("sommai.k@gmail.com", "sommai.k", "1234", "ADMIN");

repository.save(user);

};

}

* code ที่ได้หลังใส่ bean เข้าไป



**WORKSHOP##** สร้าง REST API ชื่อ UserController เก็บอยู่ใน package com.business.auth.user ของ Entity User โดยมี url ดังนี้

* GET /user
  + ดึงข้อมูลทั้งหมดของ user
* POST /user
  + รับข้อมูลเข้ามาในรูปแบบของ json object
  + เพิ่มรายการเข้าไปใน table user
* PUT /user/{id}
  + รับข้อมูลเข้ามาในรูปแบบของ json object
  + รับข้อมูล id เข้ามาผ่าน url path
  + เพิ่มรายการเข้าไปใน table user
* DELETE /user/{id}
  + รับข้อมูล id เข้ามาผ่าน url path
  + เพิ่มรายการเข้าไปใน table user
* ทั้ง 4 method เมื่อทำสำเร็จให้ return ออกมาในรูปแบบ json ดังนี้

{

success : true

data : <json>

}

* ทั้ง 4 method เมื่อมีข้อผิดพลาดให้ return ออกมาในรูปแบบ json ดังนี้

{

success : false

message : <string>

}

**วิธีการ**

* สร้าง Class ชื่อ UserController เก็บอยู่ใน package com.business.auth.user ดังนี้

package com.business.auth.user;

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

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

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

@RestController()

@RequestMapping("/user")

public class UserController {

@Autowired

UserRepository userRepository;

}

* เพิ่ม method เพื่อดึงข้อมูล User ชื่อ getUser โดยมี code ดังนี้

@GetMapping("")

public Iterable<User> getUser() {

Iterable<User> list = userRepository.findAll();

return list;

}

* เพิ่ม method เพื่อบันทึกข้อมูล User ชื่อ createUser โดยมี code ดังนี้

@PostMapping("")

public User createUser(@RequestBody Map<String, String> body) {

User user = new User(

body.get("email"),

body.get("user"),

body.get("password"),

body.get("role")

);

User res = userRepository.save(user);

return res;

}

* เพิ่ม method เพื่อ update ข้อมูล User ชื่อ updateUser โดยมี code ดังนี้

@PutMapping("/{id}")

public Map<String, Object> updateUser(

@RequestBody Map<String, String> body,

@PathVariable Long id

) {

Map<String, Object> res = new HashMap<>();

if(userRepository.existsById(id)){

User other = new User(

body.get("email"),

body.get("user"),

body.get("password"),

body.get("role")

);

other.setId(id);

userRepository.save(other);

res.put("success", true);

res.put("data", other);

}else{

res.put("success", false);

res.put("message", "no data found");

}

return res;

}

* เพิ่ม method เพื่อ delete ข้อมูล User ชื่อ deleteUser โดยมี code ดังนี้

@DeleteMapping("/{id}")

public Map<String, Object> deleteUser(

@PathVariable Long id

) {

Map<String, Object> res = new HashMap<>();

if(userRepository.existsById(id)){

userRepository.deleteById(id);

res.put("success", true);

res.put("data", id);

}else{

res.put("success", false);

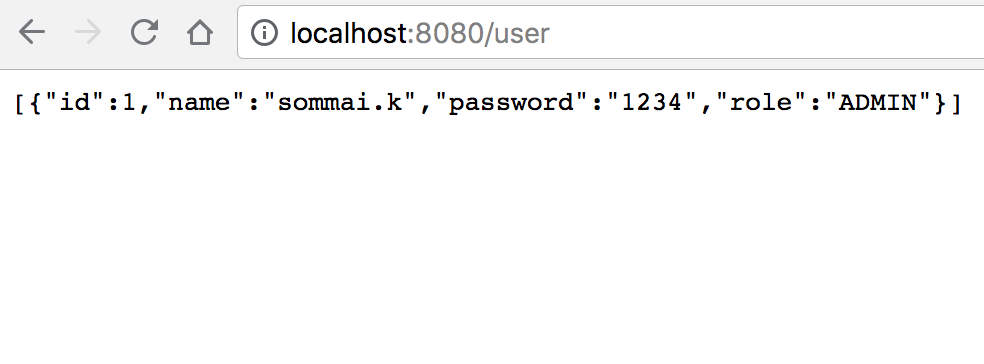
res.put("message", "no data found");

}

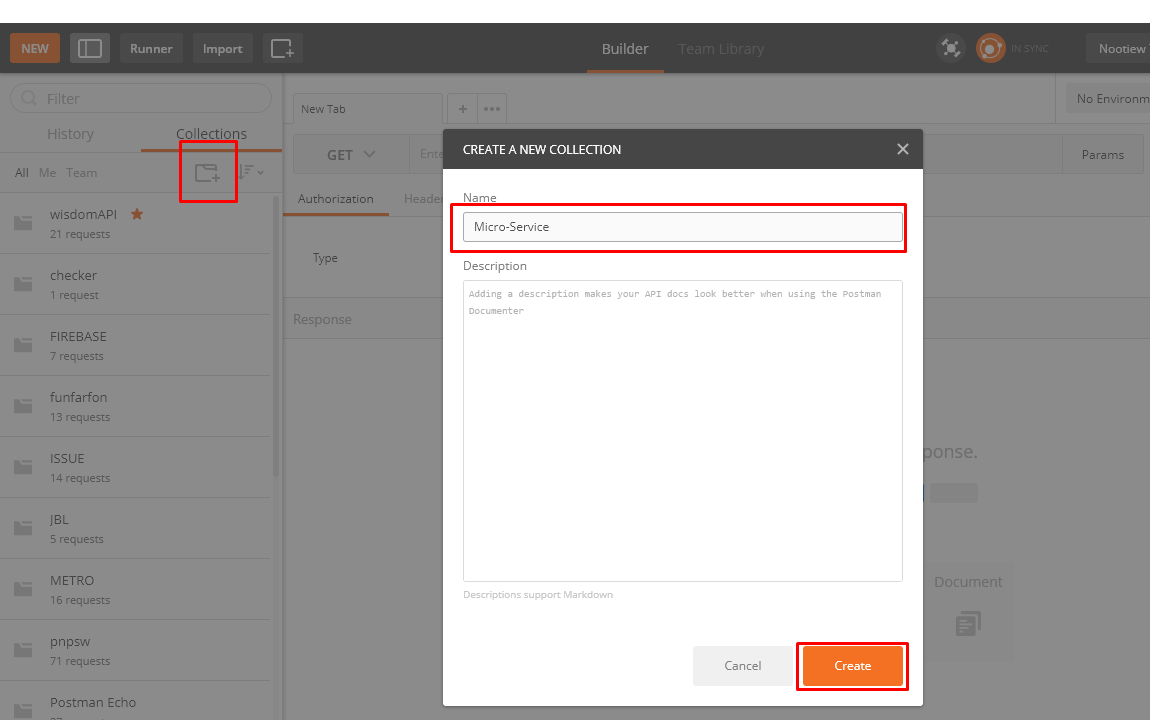
return res;

}

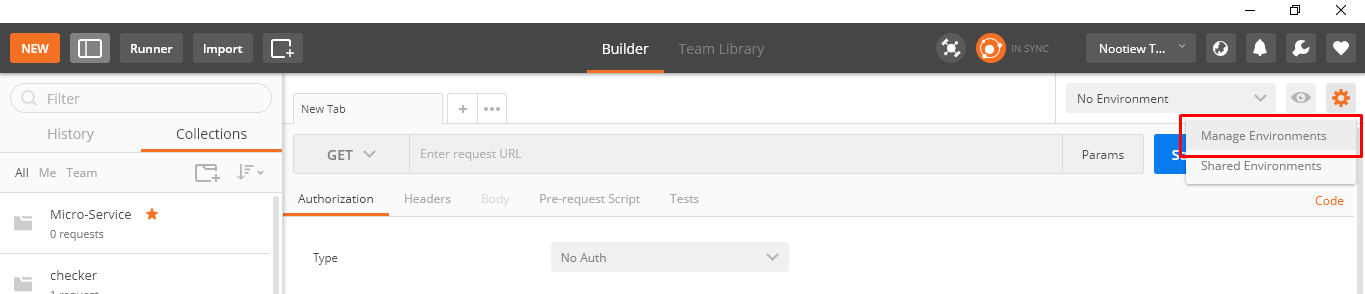
* ทดลอง run program ด้วยคำสั่งดังนี้
  + mvn spring-boot:run
* ทดลองเรียกดูข้อมูลโดยการเข้า url ดังนี้
  + <http://localhost:8080/user>
  + จะได้ผลลัพธ์ดังนี้

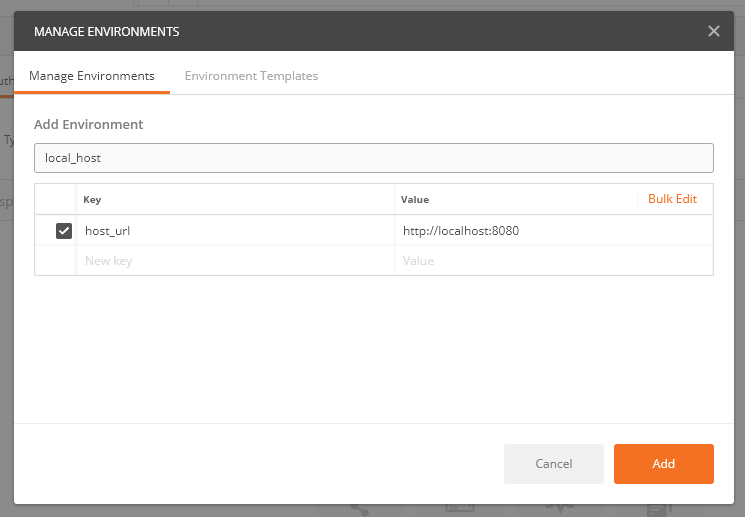


* ทดลองทดสอบ api ทั้งหมด ผ่าน postman โดยมีการตั้งค่าดังนี้
  + เข้าหน้าจอ postman สร้าง collection ชื่อ MicroService

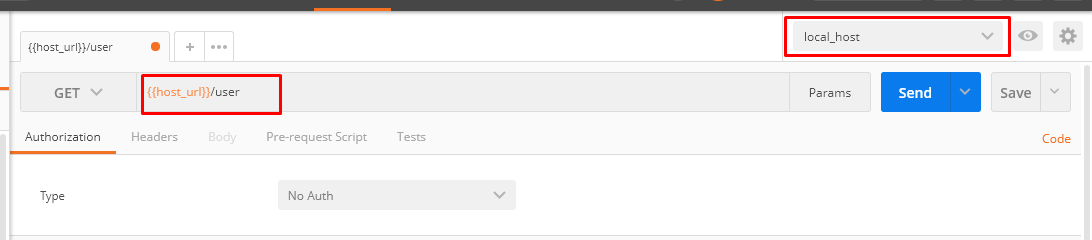


* + สร้าง environment ชื่อว่า local\_host มีค่าดังนี้
    - host\_url = <http://localhost:8080>

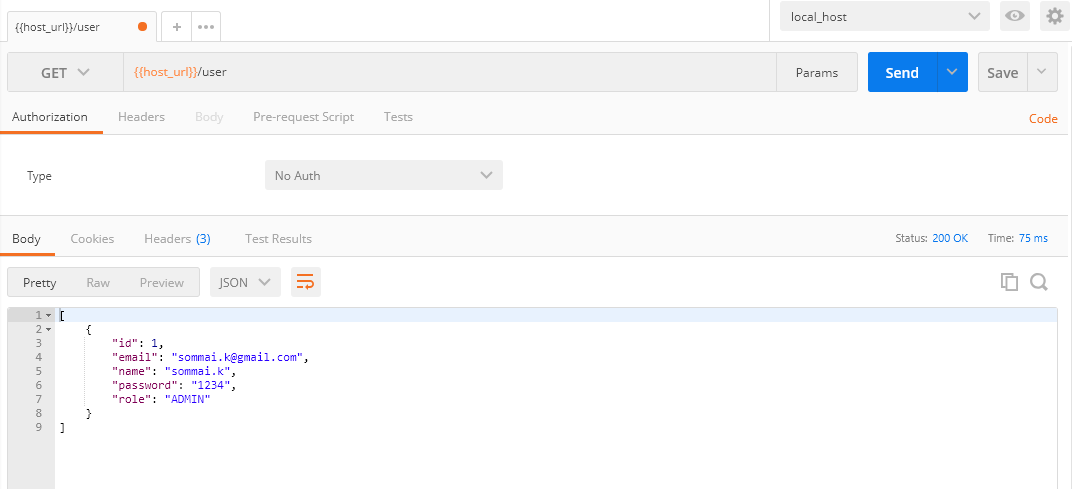




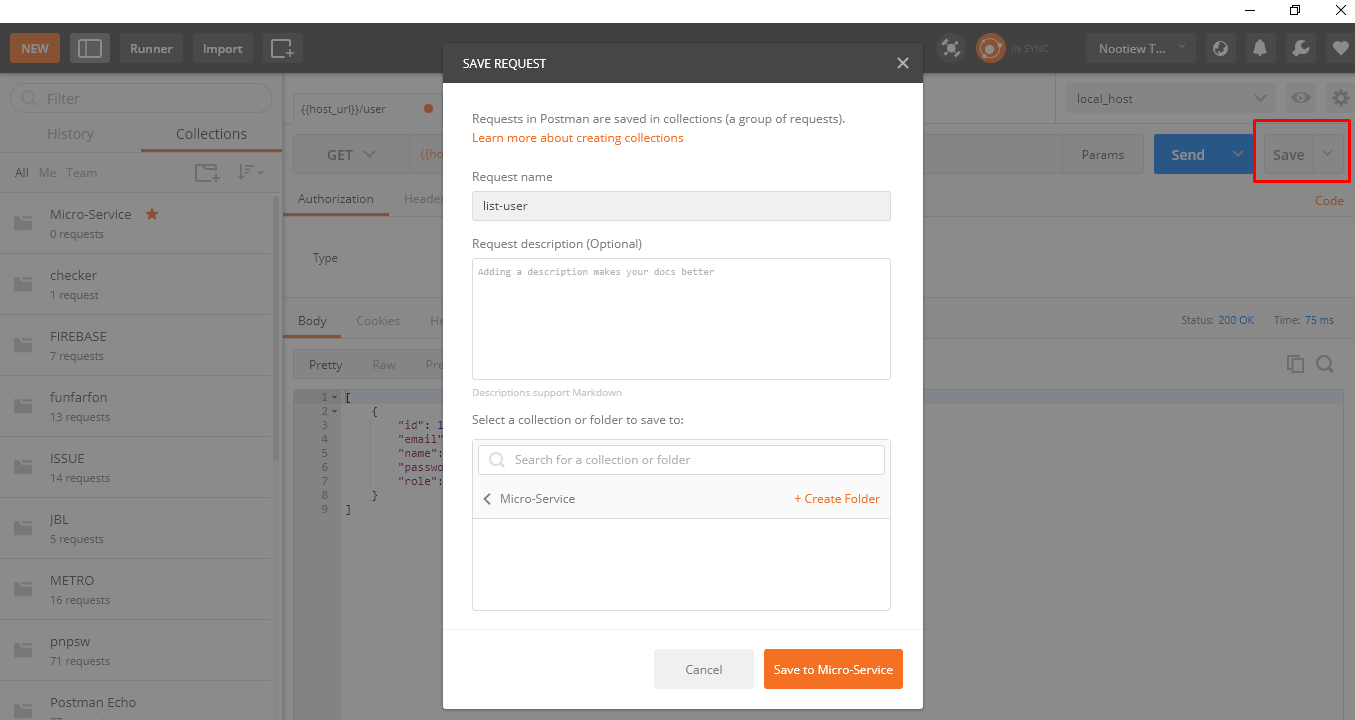
* + สร้าง tab เพื่อทดสอง method get โดยใส่ค่า url ดังนี้
    - {{host\_url}}/user



* + - กดรันจะได้ผลลัพธ์ดังนี้



* + - บันทึก link สำหรับการเทสไว้ใน collection โดยการกด Save แล้วเลือก collection ที่ต้องการ



* + สร้าง tab เพื่อทดสอง method post โดยใส่ค่า url ดังนี้
    - {{host\_url}}/user
    - body มีค่าดังนี้

{

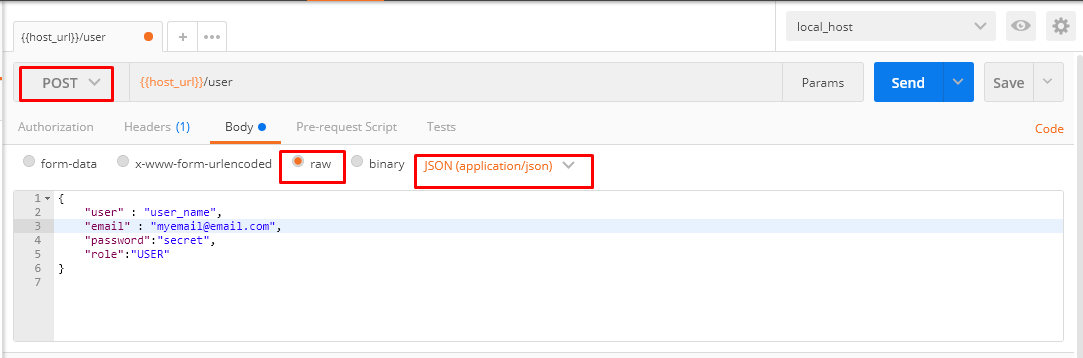
"user" : "user\_name",

"email" : "[myemail@email.com](mailto:myemail@email.com)",

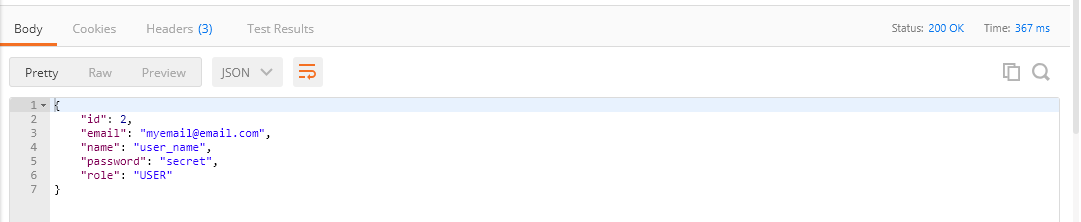
"password":"secret",

"role":"USER"

}



* + - กดรันจะได้ผลลัพธ์ดังนี้



* + สร้าง tab เพื่อทดสอง method put โดยใส่ค่า url ดังนี้
    - {{host\_url}}/user/2
    - body มีค่าดังนี้

{

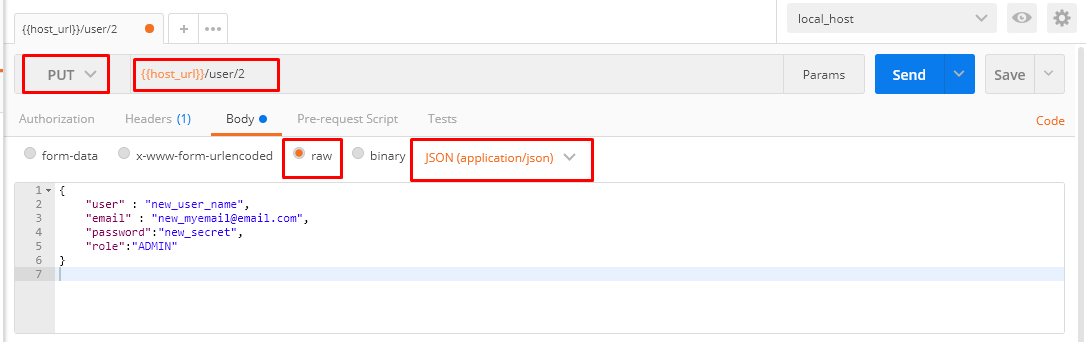
"user" : "new\_user\_name",

"email" : "new\_[myemail@email.com](mailto:myemail@email.com)",

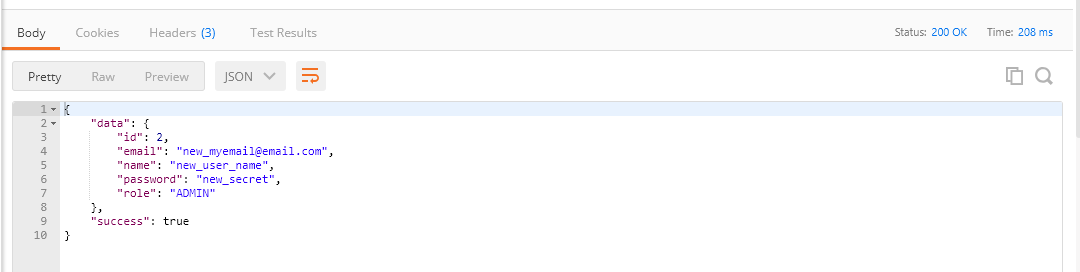
"password":"new\_secret",

"role":"ADMIN"

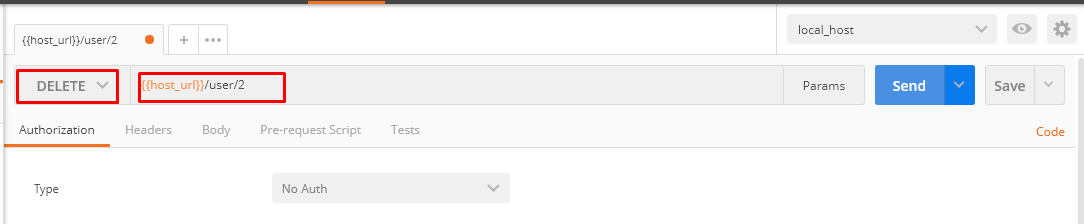
}



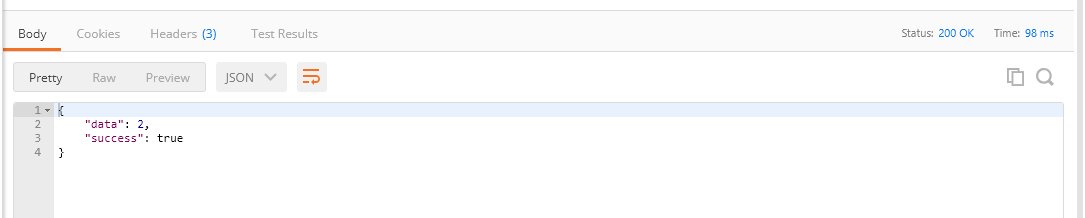
* + - กดรันจะได้ผลลัพธ์ดังนี้



* + สร้าง tab เพื่อทดสอง method delete โดยใส่ค่า url ดังนี้
    - {{host\_url}}/user/1



* + - กดรันจะได้ผลลัพธ์ดังนี้



* + เมื่อทดสอบครบทุก Method และทำการบันทึกไว้ใน collection จะได้ผลลัพธ์ดังนี้

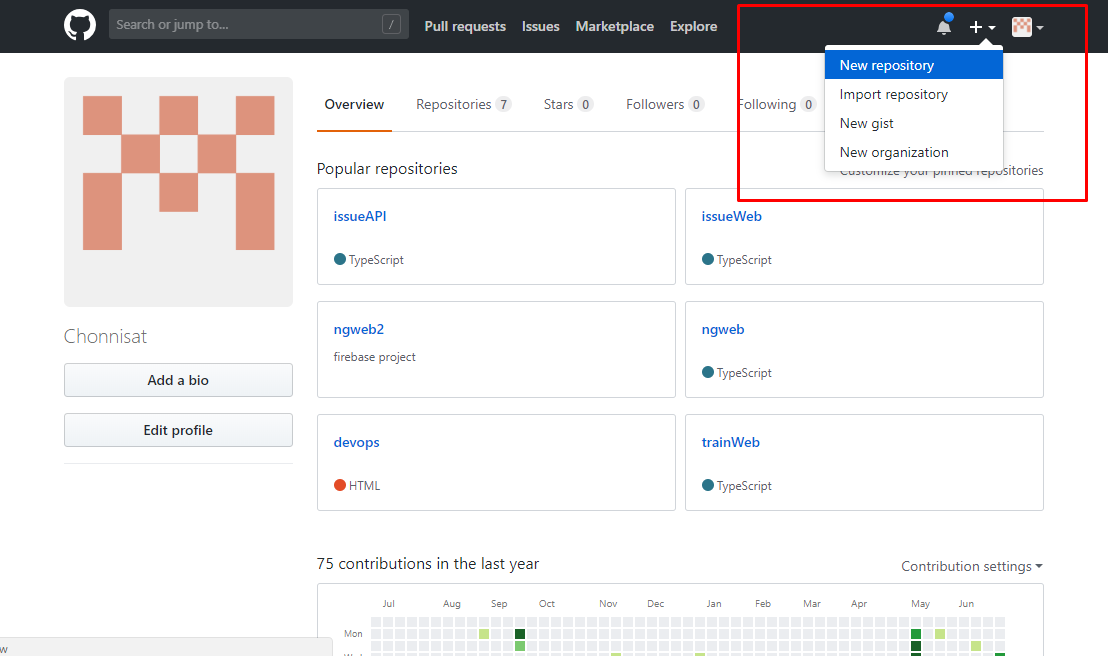


GIT

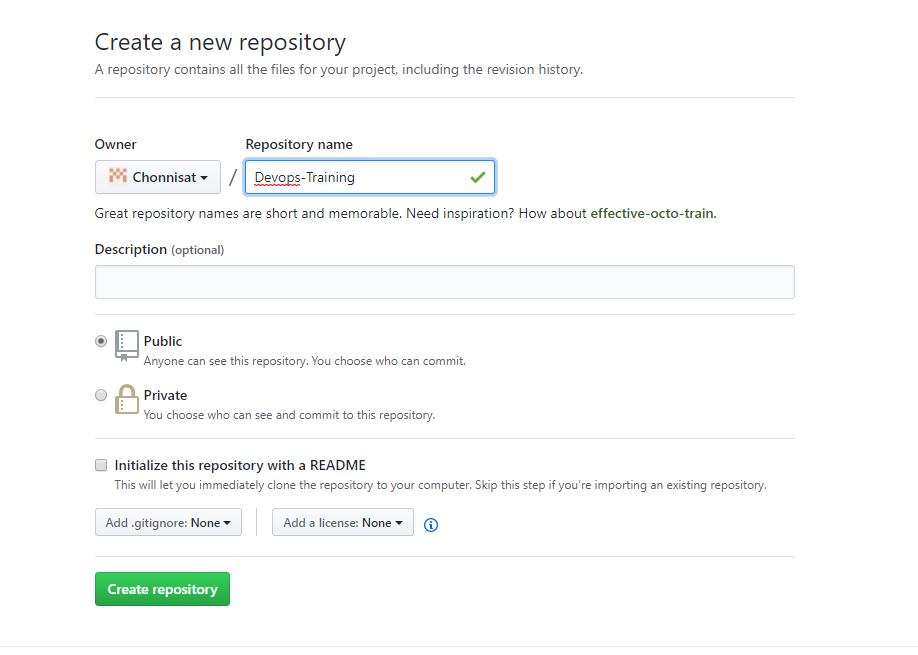
**git#1** สร้าง project ใหม่ชื่อ micro\_service\_config โดยจะทำการ sync code ขึ้นไปบน github.com เพื่อเก็บข้อมูล config file ของทั้งระบบ

**วิธีการ**

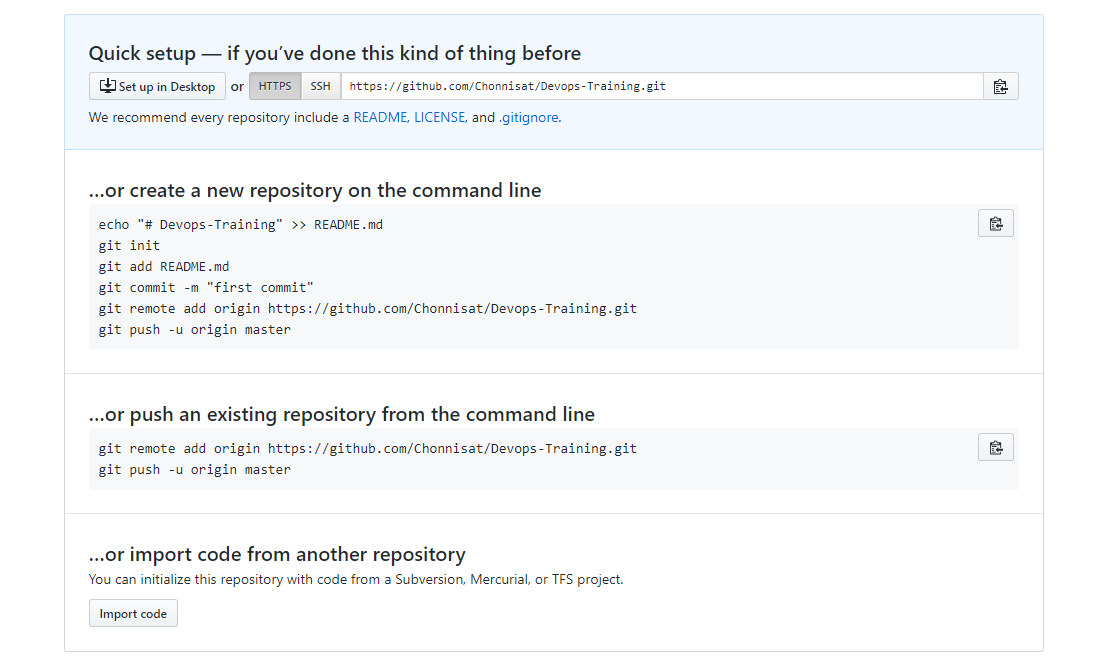
* สร้าง project ใหม่บน github ชื่อ micro\_service\_config ตามขั้นตอนดังนี้
  + เข้า website github.com login ด้วยชื่อตัวเอง
  + กดปุ่ม New repository



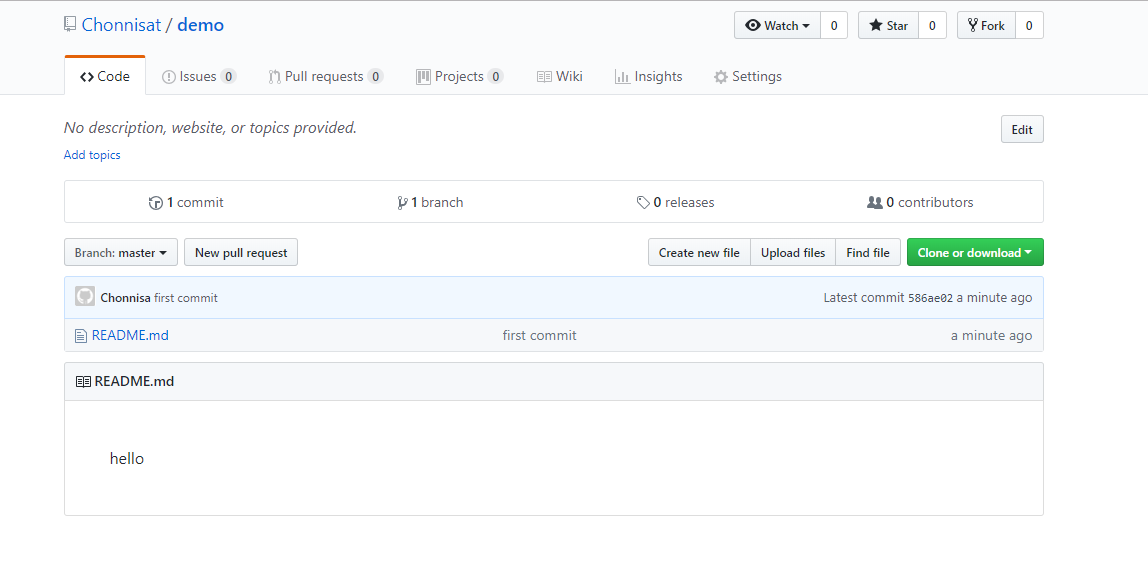
* + ใส่ชื่อ repository เป็น micro\_service\_config แล้วเลือกเป็น option เป็น public กดปุ่ม create repository



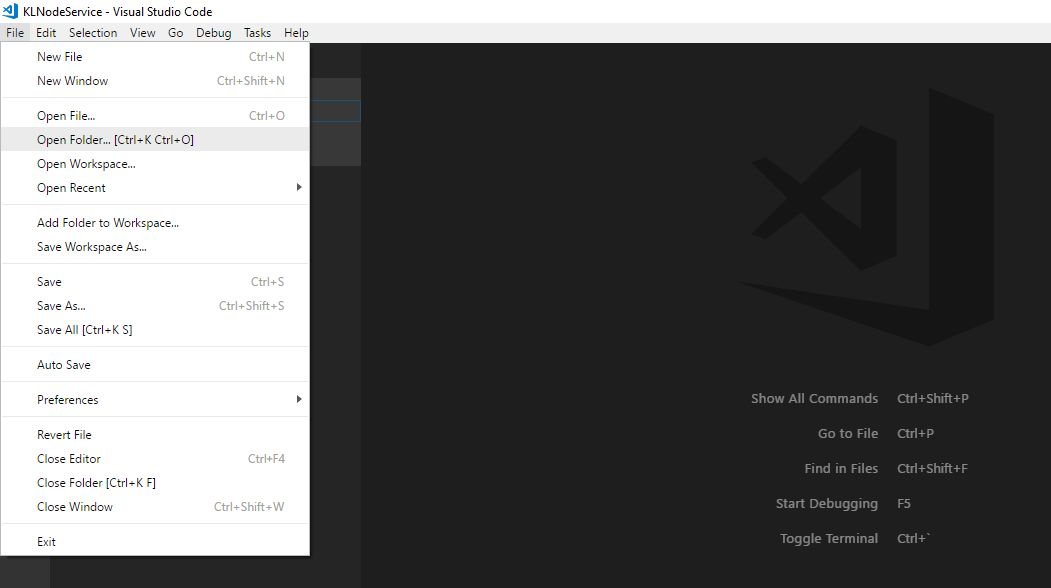
* + จะได้ผลลัพธ์ดังนี้



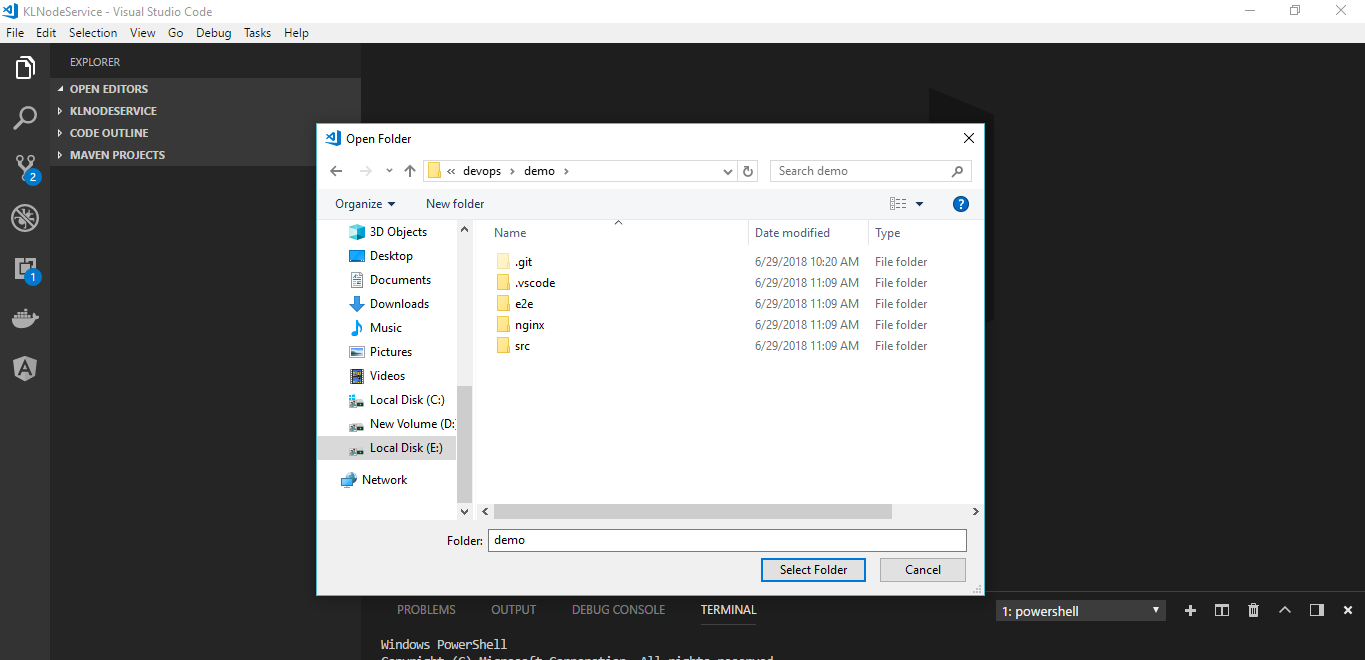
* สร้าง project ใหม่บนเครื่อง ชื่อ micro\_service\_config
  + เปิด terminal ขึ้นมาแล้วรันคำสั่ง ดังนี้
    - mkdir micro\_service\_config
    - cd micro\_service\_config
    - echo hello > README.md
  + ทำให้ project sync กับ github ด้วยการนำคำสั่งในส่วนที่ 1 มา run ดังตัวอย่าง
    - git init
    - git add README.md
    - git commit -m “first commit”
    - git remote add origin [https://github.com/<<your\_id>>/micro\_service\_config.git](https://github.com/%3c%3cyour_id%3e%3e/micro_service_config.git)
    - git push -u origin master
* ผลลัพธ์ที่ได้เมื่อเราเข้าไปที่หน้า project เราอีกครั้งบน github



* เปิด program visual studio code เพื่อทำการแก้ไข project
  + เปิด project โดยการกดปุ่ม open



* + แล้วเลือก folder ที่เราทำการสร้าง project ไว้



* สร้าง file ชื่อ auth-service.properties โดยมีข้อมูลดังนี้

## สำหรับ Mysql

spring.datasource.url=jdbc:mysql://localhost:3306/my\_db

spring.datasource.username=my\_db

spring.datasource.password=my\_db

spring.datasource.driver=com.mysql.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

## สำหรับ H2

spring.h2.console.enabled=true

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

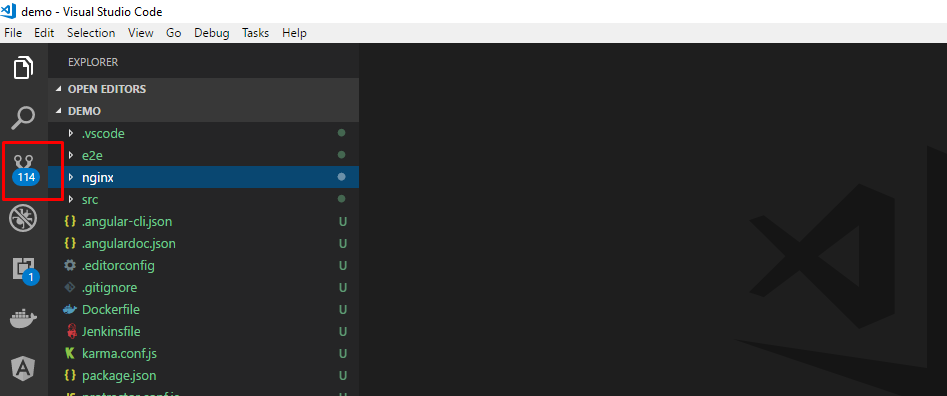
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

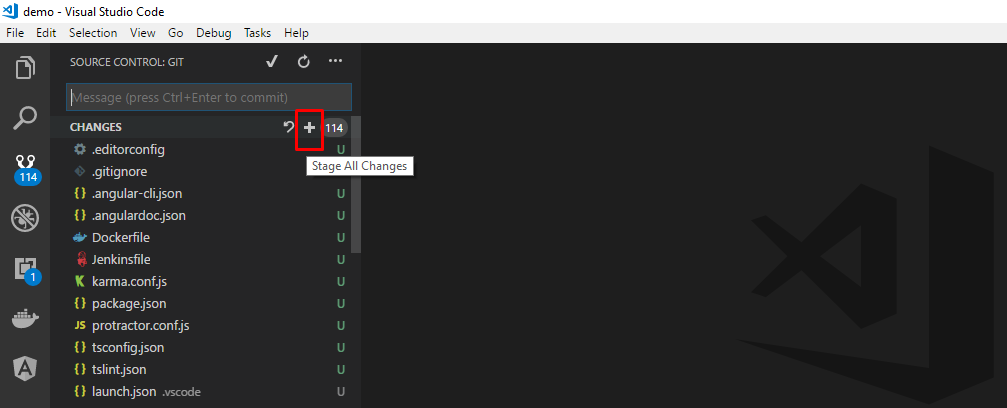
info.app.name=Auth Service

info.app.description=This is auth service on spring boot application

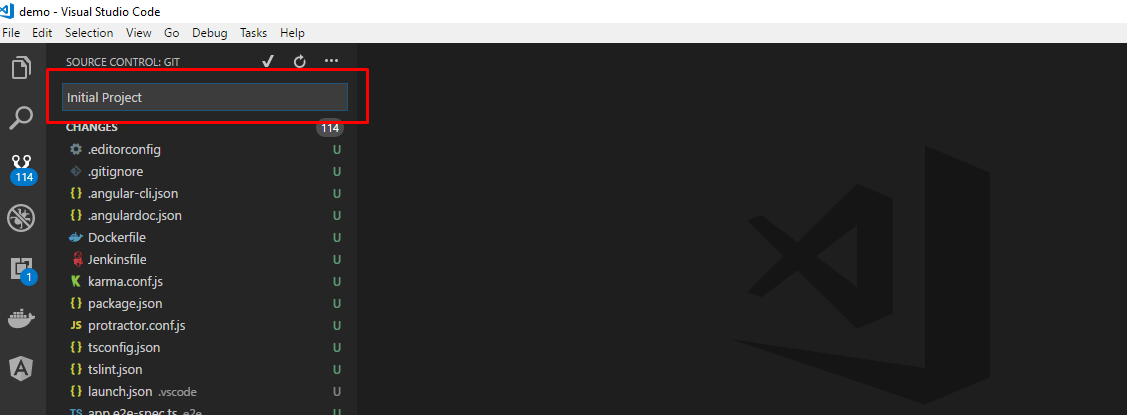
info.app.version=1.0.0

* sync code ขึ้น github ด้วย gui มีวิธีการดังนี้
  + เข้าไปในส่วนของการ sync code โดยการกดปุ่มดังนี้

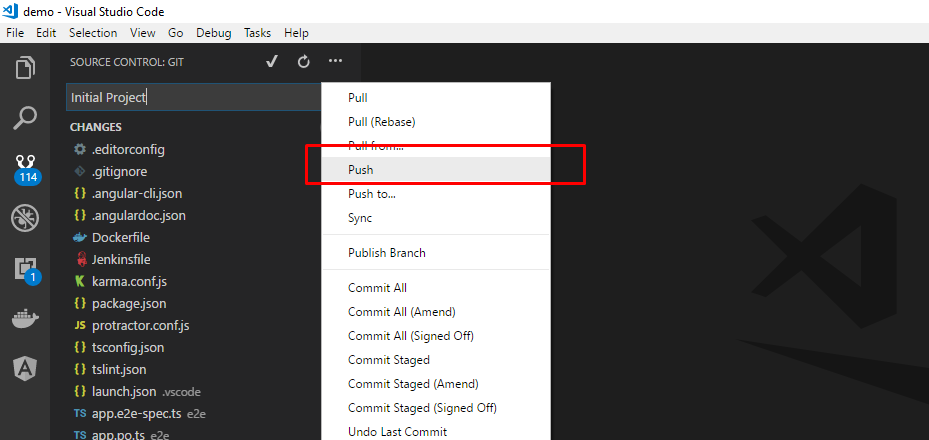




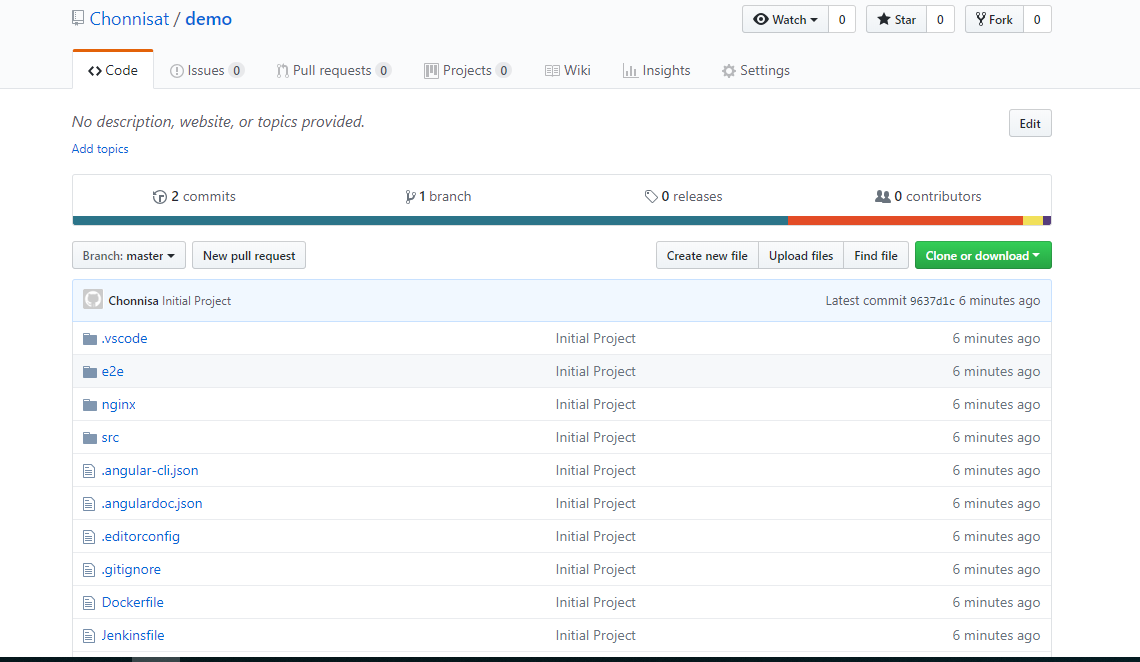
* + พิมพ์ message ที่ต้องการจะ commit



* + กดปุ่ม เพื่อทำการ push code ขึ้นไปบน github



* ตรวจสอบผลลัพธ์ที่ได้จากการ sync code



\*\*หมายเหตุ\*\*

spring-boot สามารถแยก profile ได้ด้วยคำสั่งดังนี้

mvn spring-boot:run -Dspring-boot.run.profiles=prod

หรือ

java -jar XXX.jar --spring.profiles.active=prod

ระบบจะทำการค้นหา file ดังนี้

สมมุติว่าชื่อ project เป็น auth-service แล้วรัน profile prod ก็จะไปหา file ชื่อ

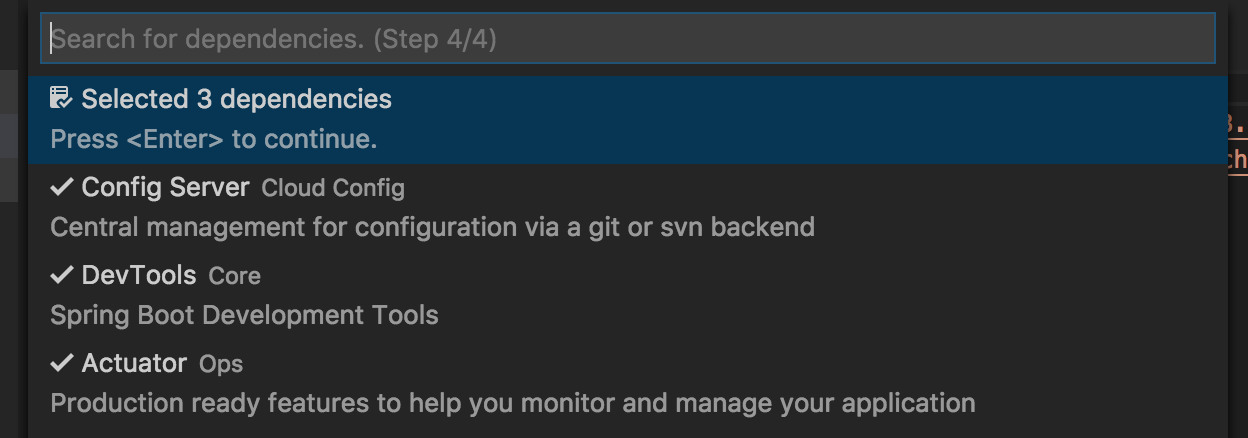
auth-service-prod.properties

CONFIG SERVER

**WORKSHOP#1** สร้าง config server เพื่อใช้ในการเก็บรวบรวม config ของทุกๆ microservice โดยเก็บ config file ใว้ใน github.com

**วิธีการ**

* สร้าง project ใหม่ชื่อ config\_server โดยมีคุณสมบัติดังนี้
  + group id = com.business
  + artifect id = config
  + dependencies ดังนี้
    - actuator
    - config server
    - devtools

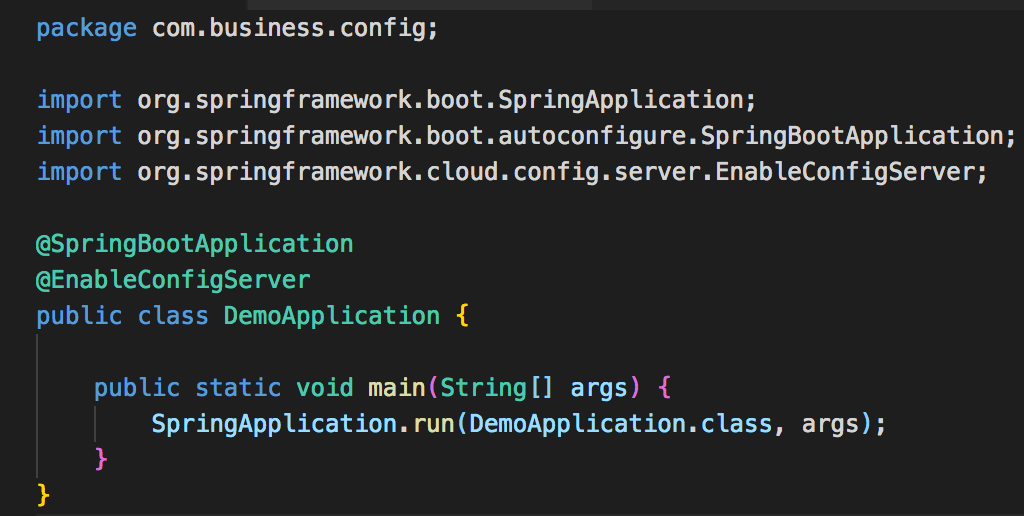


* แก้ไข file application.properties ของ project config\_server ดังนี้

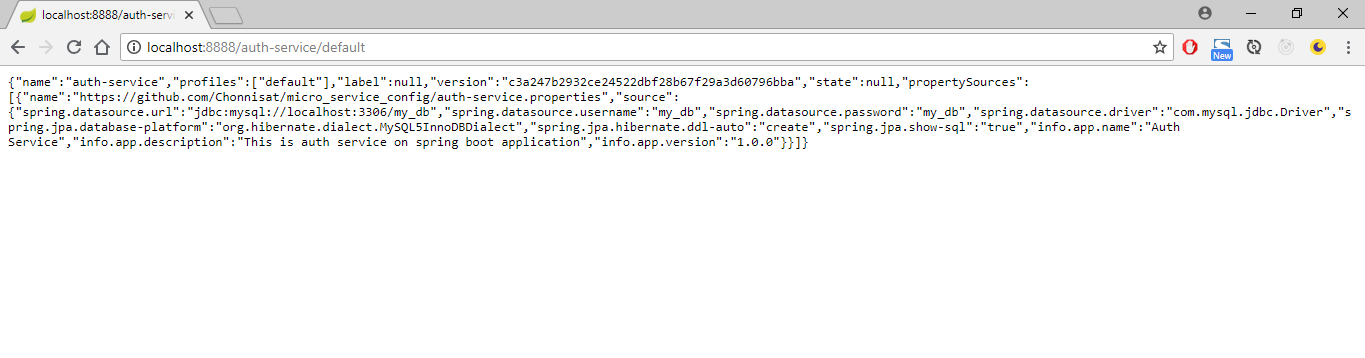
server.port=8888

spring.cloud.config.server.git.uri=https://github.com/<<your\_id>>/ micro\_service\_config

* เพิ่ม @EnableConfigServer เพื่อเปิดใช้ง่าน Configserver โดยเพิ่มไปใน code ดังนี้



* start config server ด้วยคำสั่งดังนี้
  + mvn spring-boot:run
* ทดสอบการทำงานของ config server ด้วยการทดลองเปิด url ดังนี้
  + <http://localhost:8888/auth-service/default>
  + จะได้ผลลัพธ์ดังนี้



\*\*หมายเหตุ\*\*

default หมายถึง ค่า profile ที่ต้องการดู

CONFIG CLIENT

**WORKSHOP#1** เรียกใช้ configuration file จาก config server

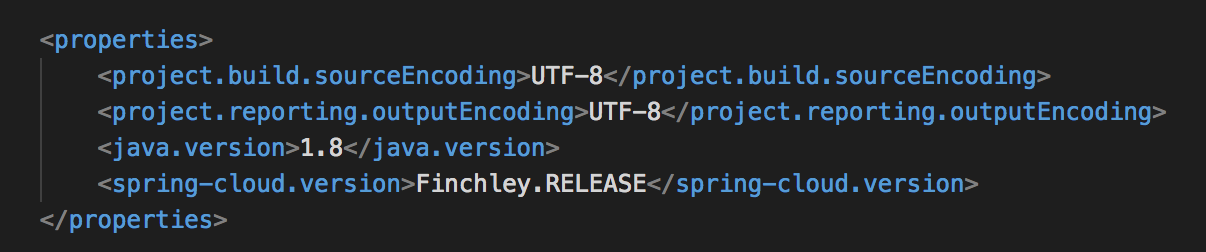
**วิธีการ**

* เปิด project auth\_service
  + เพิ่ม file bootstrap.properties โดยมีข้อมูลดังนี้

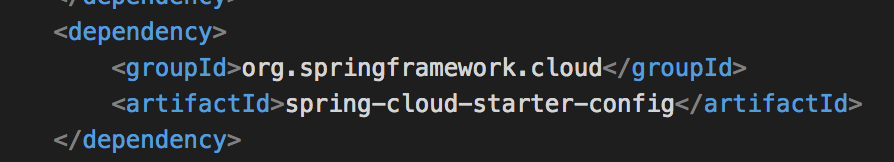
spring.application.name=auth-service

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

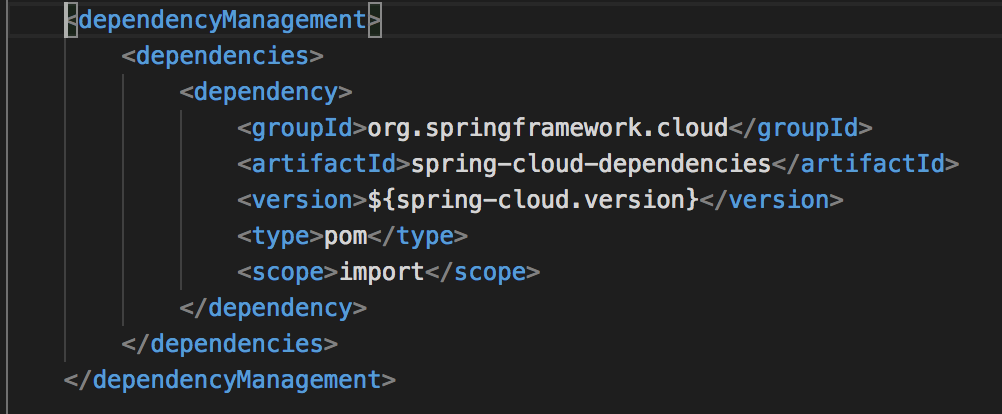
* ลบข้อมูลใน file application.properties ให้ว่าง
* เพิ่ม properties เข้าไปใน file pom.xml ดังนี้



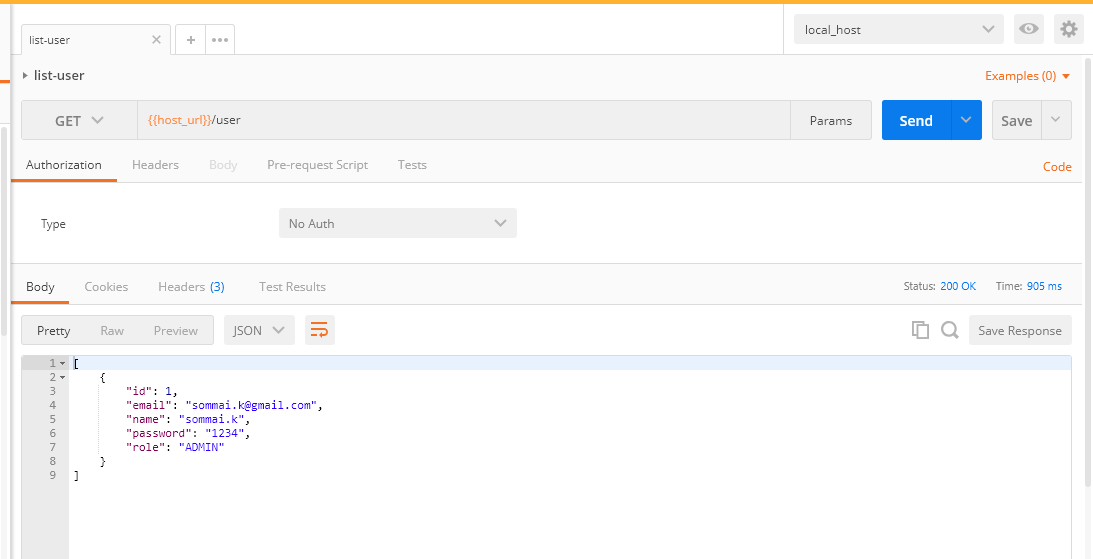
* เพิ่ม dependencies เข้าไป ดังนี้



* เพิ่ม dependencyManagement ต่อจาก dependencies เข้าไปดังนี้



* start auth-service ด้วยคำสั่งดังนี้
  + mvn spring-boot:run
* ทดสอบการทำงานของ auth-service ด้วย postman



EUREKA SERVER

**WORKSHOP#1** set up eureka server เพื่อใช้สำหรับทำ service discovery

**วิธีการ**

* สร้าง project ใหม่ชื่อ eureka\_server โดยมีคุณสมบัติดังนี้
  + group id = com.business
  + artifect id = eureka
  + dependencies ดังนี้
    - actuator
    - eureka server
    - devtools



* แก้ไข application.properties ให้เป็นดังนี้

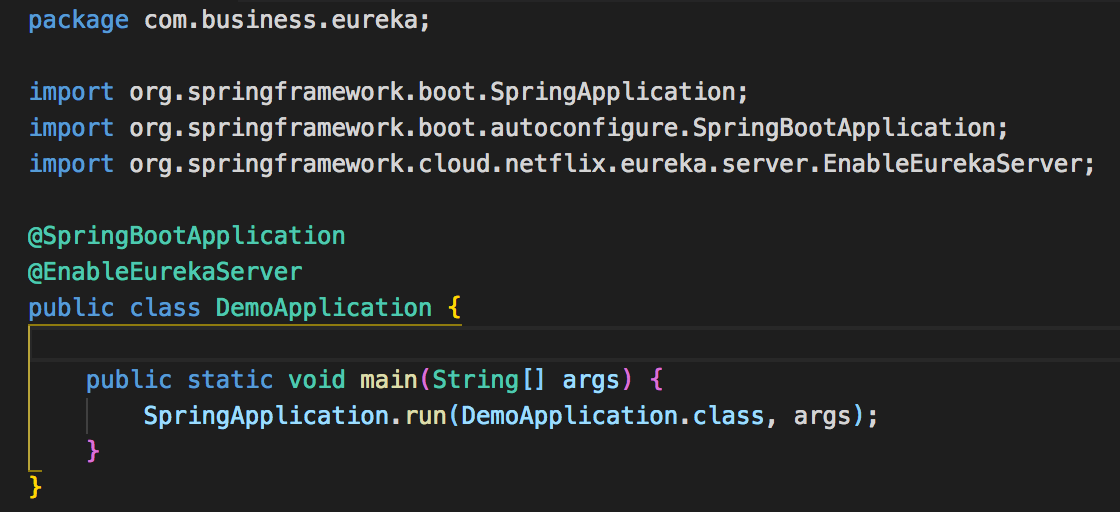
spring.application.name=eureka-server

server.port=8761

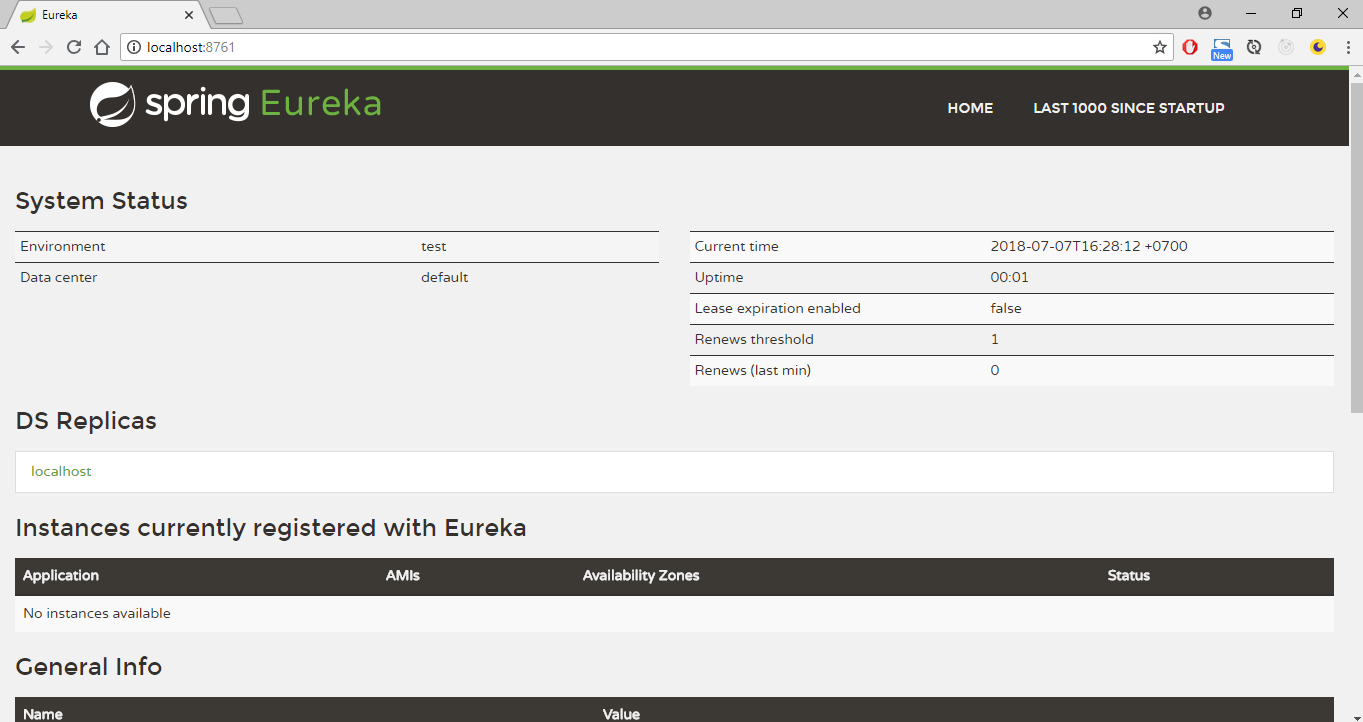
eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

* Add @EnableEurekaServer ที่ file DemoApplication.java ดังนี้



* ทดสอบการทำงานของ Euraka Server โดยการเริ่ม start server ด้วยคำสั่งดังนี้
  + mvn spring-boot:run
* ทดสอบเข้า url ดังนี้
  + <http://localhost:8761>



**Exercise**

**exercise#1** ให้เปลี่ยนการเรียกใช้ config file ของ eureka\_server จาก application.properties ไปใช้บน config server

**\*\*แนวทาง\*\***

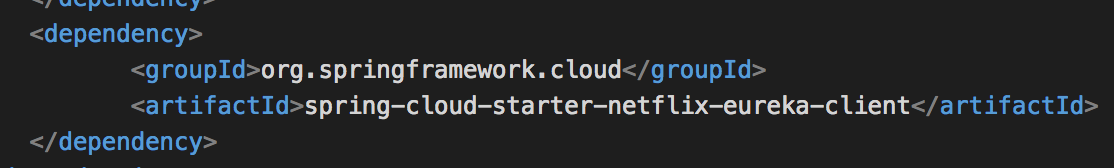
* สร้าง file eureka-server.properties ที่ project micro\_service\_config แล้ว push ขึ้น git server
* สร้าง file bootstrap.properties ที่ project eureka\_server
* ลบข้อมูลใน file application.properties ที่ project eureka\_server
* restart server eureka\_server

EUREKA CLIENT

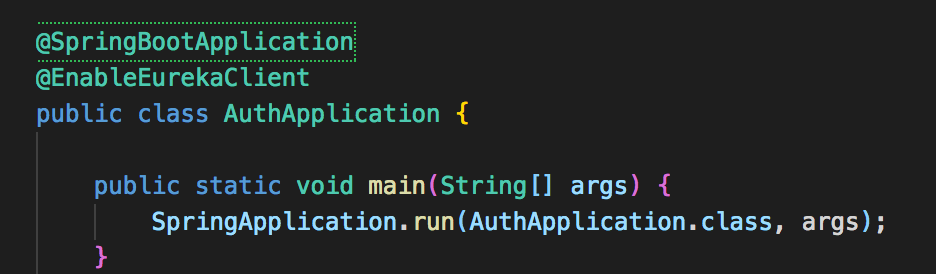
**WORKSHOP#1** set up auth service ให้เป็น eureka client

**วิธีการ**

* เปิด project ชื่อ auth\_service
  + เพิ่ม dependencies ใน file pom.xml ดังนี้



* + เพิ่ม @EnableEurekaClient ที่ File AuthApplication.java ดังนี้

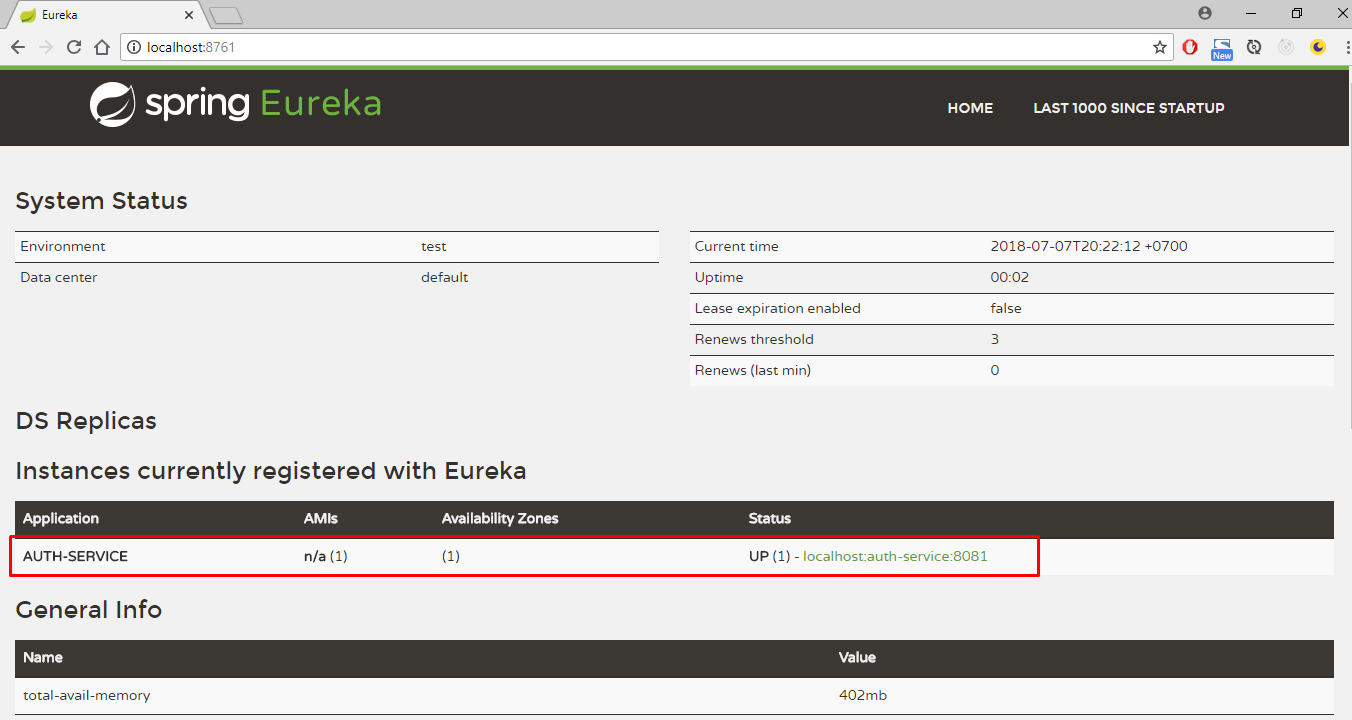


* เปิด project micro\_service\_config แล้วแก้ไข file auth-service.properties โดยเพิ่ม config ดังนี้

server.port=8081

eureka.client.service-url.default-zone=http://localhost:8761/eureka

* restart auth\_service ใหม่
* ทดลองว่า service ได้ตั้งค่าถูกต้องโดยไปตรวจสอบว่ามีรายการ service อยู่บน Eureka Server หรือไม่ โดยเข้า url ดังนี้
  + <http://localhost:8761>

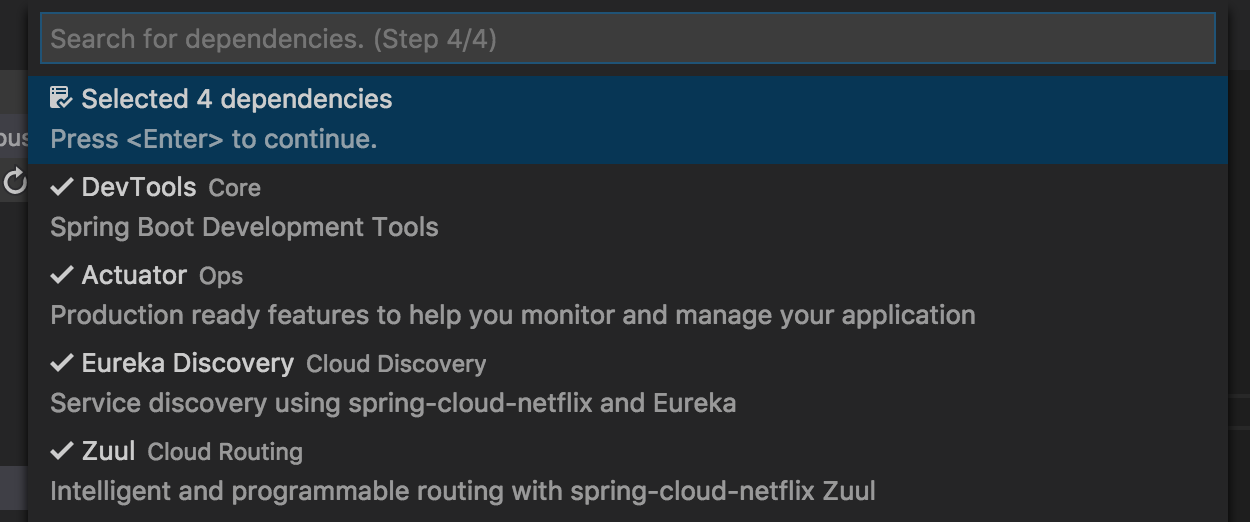


ZUUL PROXY

**WORKSHOP#1** setup zuul proxy server เพื่อรองรับการเรียกจาก web browser

**วิธีการ**

* สร้าง project ใหม่ชื่อ zuul\_server โดยมีคุณสมบัติดังนี้
  + group id = com.business
  + artifect id = zuul
  + dependencies ดังนี้
    - actuator
    - zuul
    - devtools
    - eureka discovery



* แก้ไข file application.properties ของ project zuul\_server ดังนี้

server.port=8762

spring.application.name=zuul-server

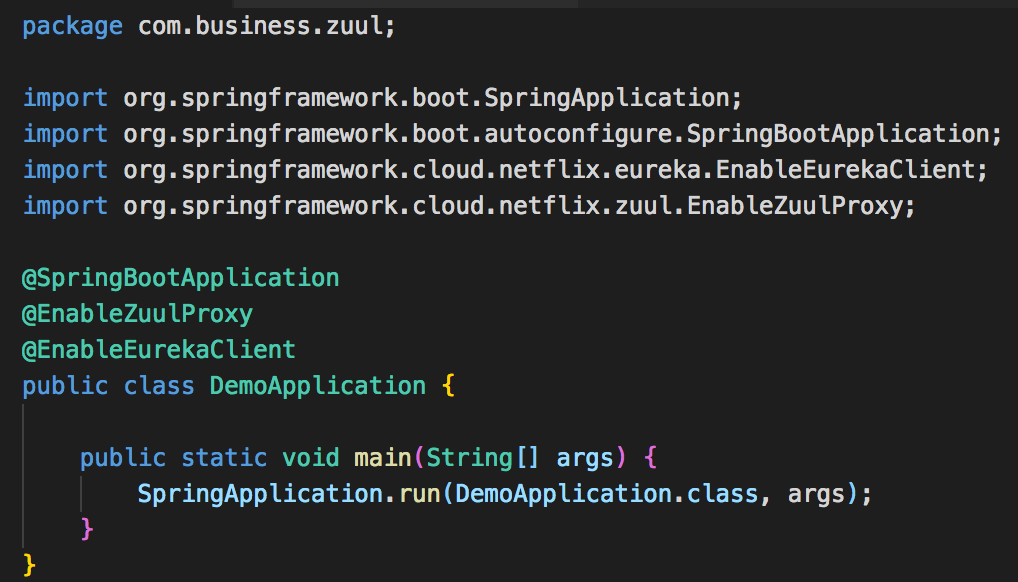
eureka.client.service-url.default-zone=http://localhost:8761/eureka/

zuul.ignored-services=\*

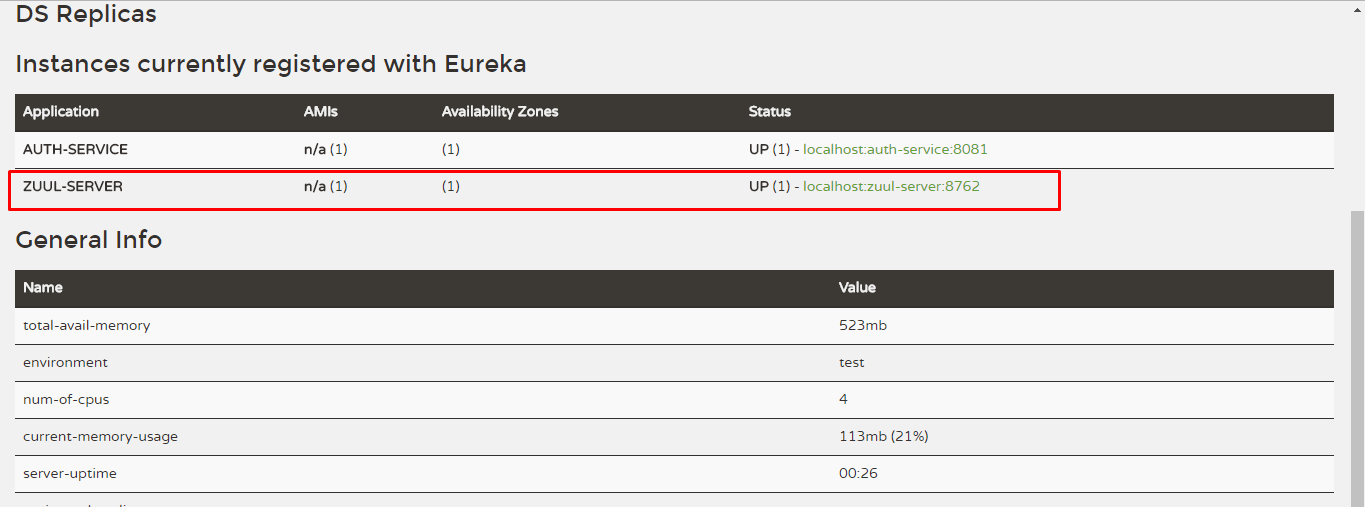
zuul.routes.auth-service.path=/auth/\*\*

zuul.routes.auth-service.service-id=auth-service

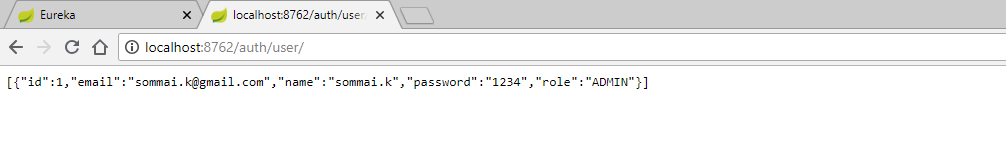
* เพิ่ม @EnableZuulProxy, @EnableEurekaClient เพื่อเปิดใช้ง่าน ZuulProxy โดยเพิ่มไปใน code ดังนี้



* start config server ด้วยคำสั่งดังนี้
  + mvn spring-boot:run
* ทดลองว่า service ได้ตั้งค่าถูกต้องโดยไปตรวจสอบว่ามีรายการ service อยู่บน Eureka Server หรือไม่ โดยเข้า url ดังนี้
  + <http://localhost:8761>



* ทดสอบการทำงานของ zuul server ด้วยการทดลองเปิด url ดังนี้
  + <http://localhost:8762/auth/user/>
  + จะได้ผลลัพธ์ดังนี้



**Exercise**

**exercise#1** ให้เปลี่ยนการเรียกใช้ config file ของ zuul\_server จาก application.properties ไปใช้บน config server

**\*\*แนวทาง\*\***

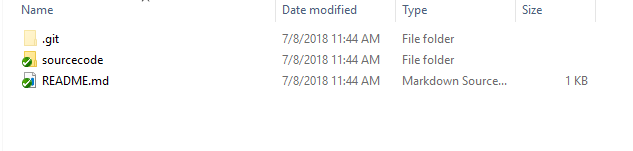
* สร้าง file zuul-server.properties ที่ project micro\_service\_config แล้ว push ขึ้น git server
* สร้าง file bootstrap.properties ที่ project zuul\_server
* ลบข้อมูลใน file application.properties ที่ project zuul\_server
* restart server zuul\_server

SPRING BOOT SECURITY WITH JWT

**WORKSHOP#1** clone project มาจาก github

**วิธีการ**

* clone project จาก github เพื่อเอา source code security มาใช้งาน โดยการรันคำสั่งดังนี้
  + git clone <https://github.com/Sommaik/springboot-workshop.git>
  + จะได้ folder ดังนี้



**WORKSHOP#2** แก้ไข proxy server ให้ทำการ check security ก่อนการเรียกใช้ service

**วิธีการ**

* แก้ไข file pom.xml ของ zuul\_server เพิ่มรายการดังนี้เข้าไป

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

<dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt</artifactId>  
 <version>0.9.0</version>  
</dependency>

* แก้ไข file application.properties ของ zuul\_server เพิ่มรายการดังนี้

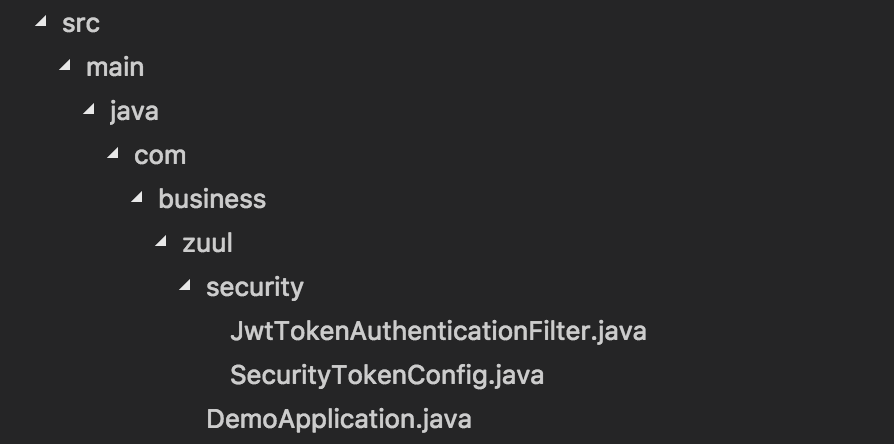
zuul.routes.auth-service.strip-prefix=false

zuul.routes.auth-service.sensitive-headers=Cookie,Set-Cookie

zuul.routes.auth-admin.path=/admin/\*\*

zuul.routes.auth-admin.service-id=auth-service

* สร้าง folder ชื่อ security ภายใน folder zuul แล้วนำ code จาก path /sourcecode/security/zuul\_server มาใส่ไว้ดังภาพ



**WORKSHOP#3** แก้ไข project auth\_service เพื่อให้รองรับการใช้ jwt security

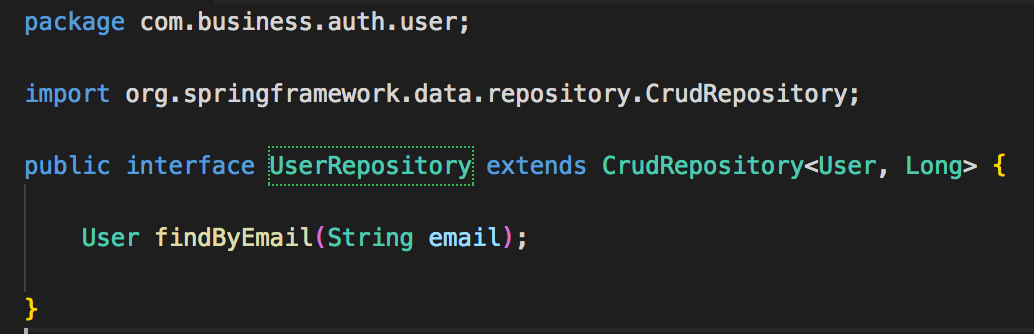
**วิธีการ**

* แก้ไข file pom.xml ของ auth\_server เพิ่มรายการดังนี้เข้าไป

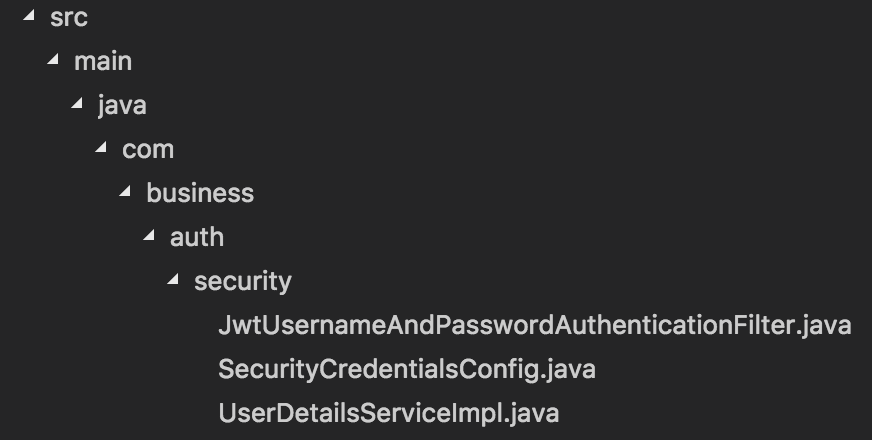
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
</dependency>

<dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt</artifactId>  
 <version>0.9.0</version>  
</dependency>

* เพิ่ม method ใน interface UserRepository เพื่อให้สามารถค้นหาตาม email ได้ ดังภาพ



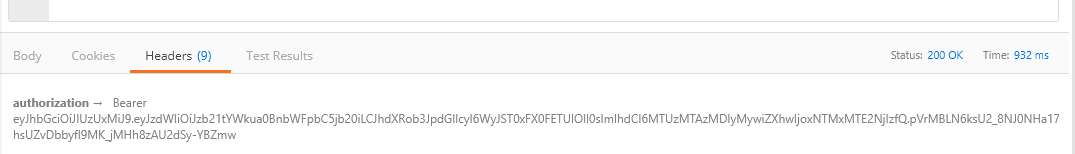
* สร้าง folder ชื่อ security ภายใน folder auth แล้วนำ code จาก path /sourcecode/security/auth\_server มาใส่ไว้ดังภาพ



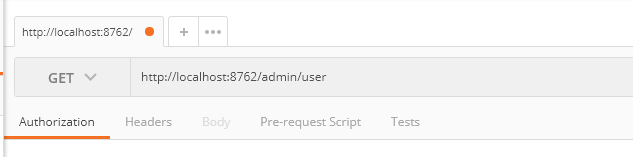
* restart server auth\_service, zuul\_server ตามลำดับ
* เข้าไปที่ post man เพื่อทดสอบการทำงานว่า security ทำงานหรือไม่
  + สร้าง post request ไปที่ <http://localhost>:8762/auth ดังภาพ



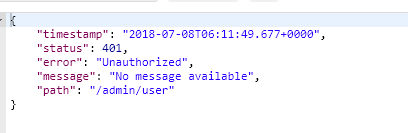
* เมื่อสั่ง run จะได้ token มาดังภาพ



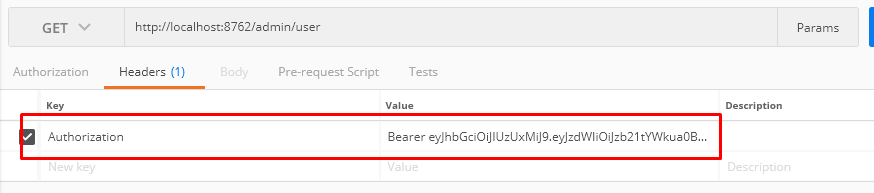
* + สร้าง get request ไปที่ <http://localhost>:8762/admin/user ดังภาพ



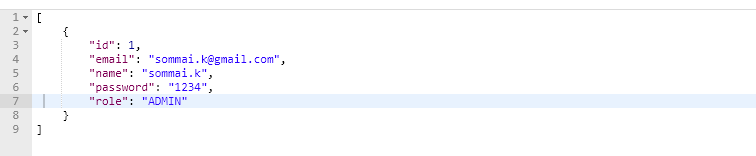
* เมื่อสั่ง run จะได้ ผลลัพธ์เป็น 401 unauthorized มาดังภาพ



* ให้นำ token ที่ได้ส่งไปที่ header ดังภาพ



* เมื่อสั่ง run จะได้ ผลลัพธ์เป็นข้อมูลมาดังภาพ



ORDER SERVICE

**WORKSHOP#1** สร้าง file ชื่อ order-service.properties ใน micro\_service\_config โดยรายการดังนี้

## สำหรับ Mysql

spring.datasource.url=jdbc:mysql://localhost:3306/my\_db

spring.datasource.username=my\_db

spring.datasource.password=my\_db

spring.datasource.driver=com.mysql.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

## สำหรับ H2

spring.h2.console.enabled=true

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

info.app.name=Order Service Application

info.app.description=Order Service on spring boot application

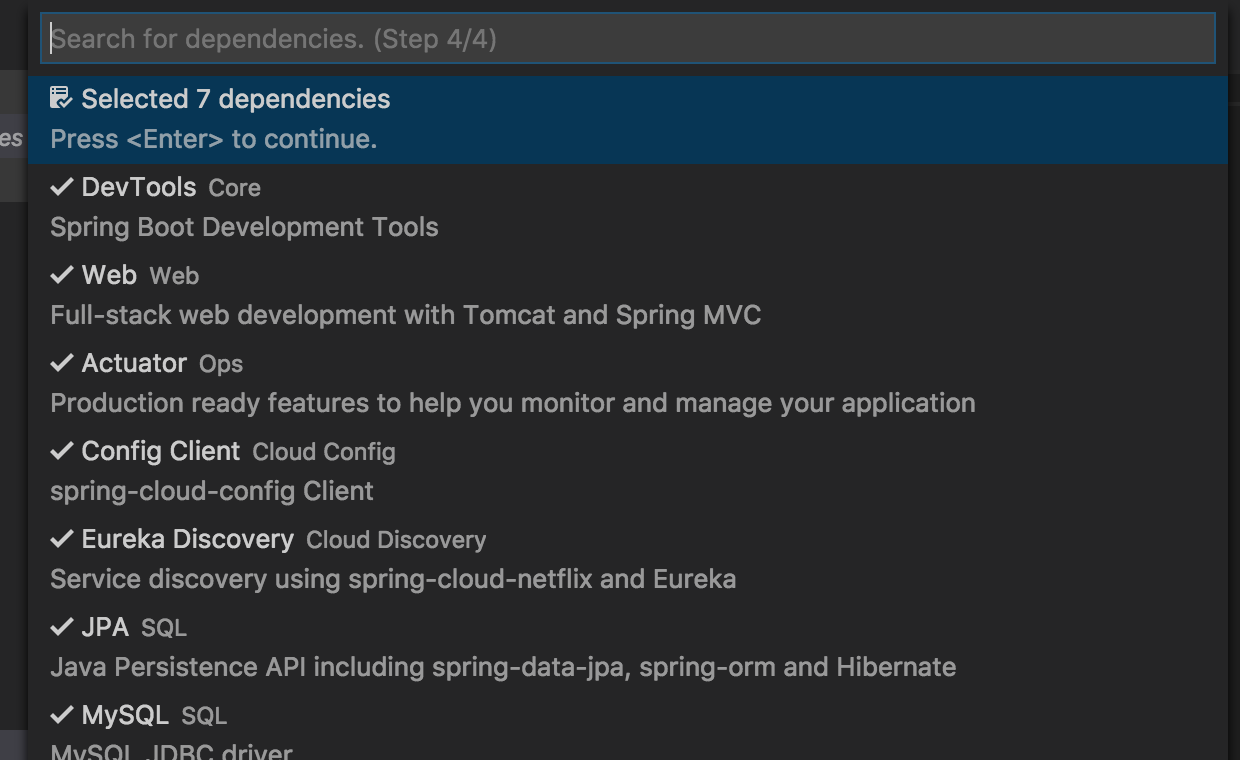
info.app.version=1.0.0

server.port=8082

eureka.client.service-url.default-zone=http://localhost:8761/eureka

**WORKSHOP#2** สร้าง project ชื่อ order\_service โดยมีคุณสมบัติ ดังนี้

* group id = com.business
* artifect id = order
* dependencies ดังนี้
  + web
  + actuator
  + jpa (sql)
  + mysql หรือ h2
  + devtools
  + config client
  + eureka discovery

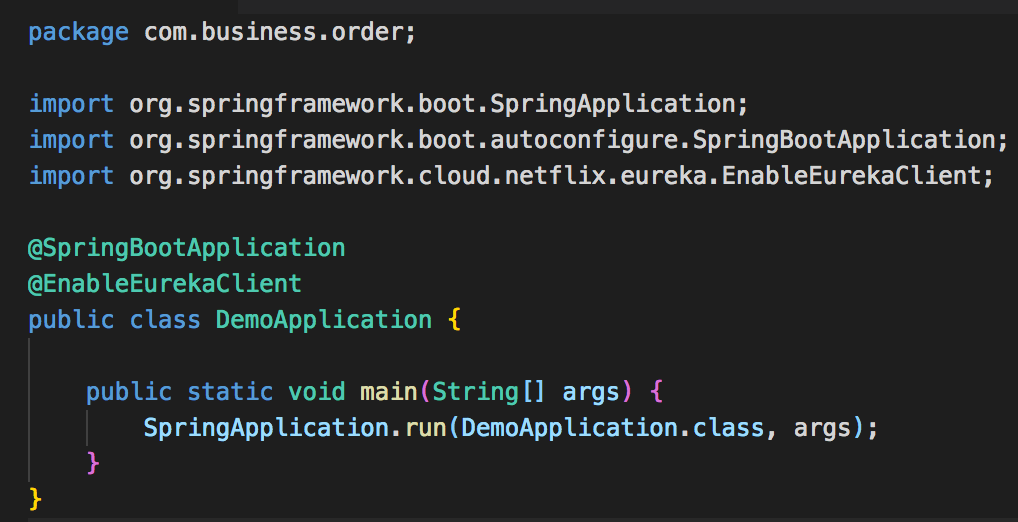


* สร้าง file bootstrap.properties ดังนี้

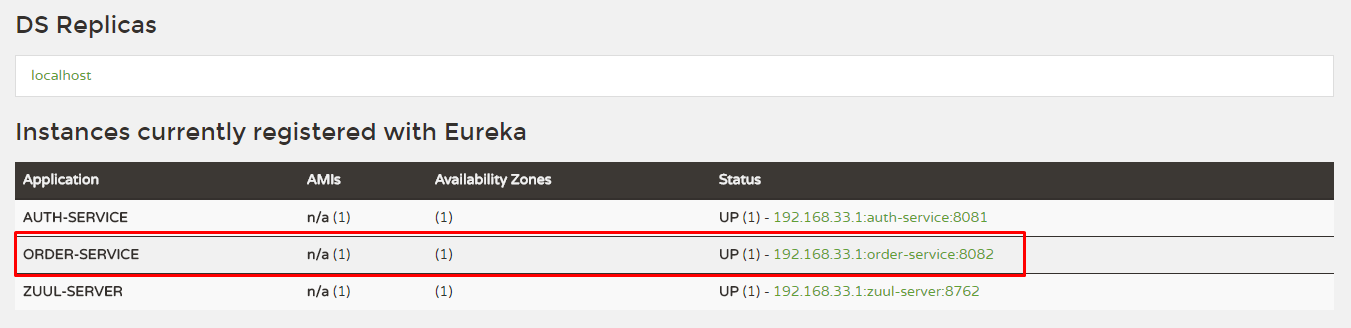
spring.application.name=order-service

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

* เพิ่ม @EnableEurekaClient ดังภาพ



* start service ด้วยคำสั่งดังนี้
  + mvn spring-boot:run
* ทดสอบว่าระบบทำงานถูกต้องโดยการเข้าไปที่ url ดังนี้
  + <http://localhost:8761>
  + จะมีชื่อ order-service ปรากฏขึ้นมาดังภาพ

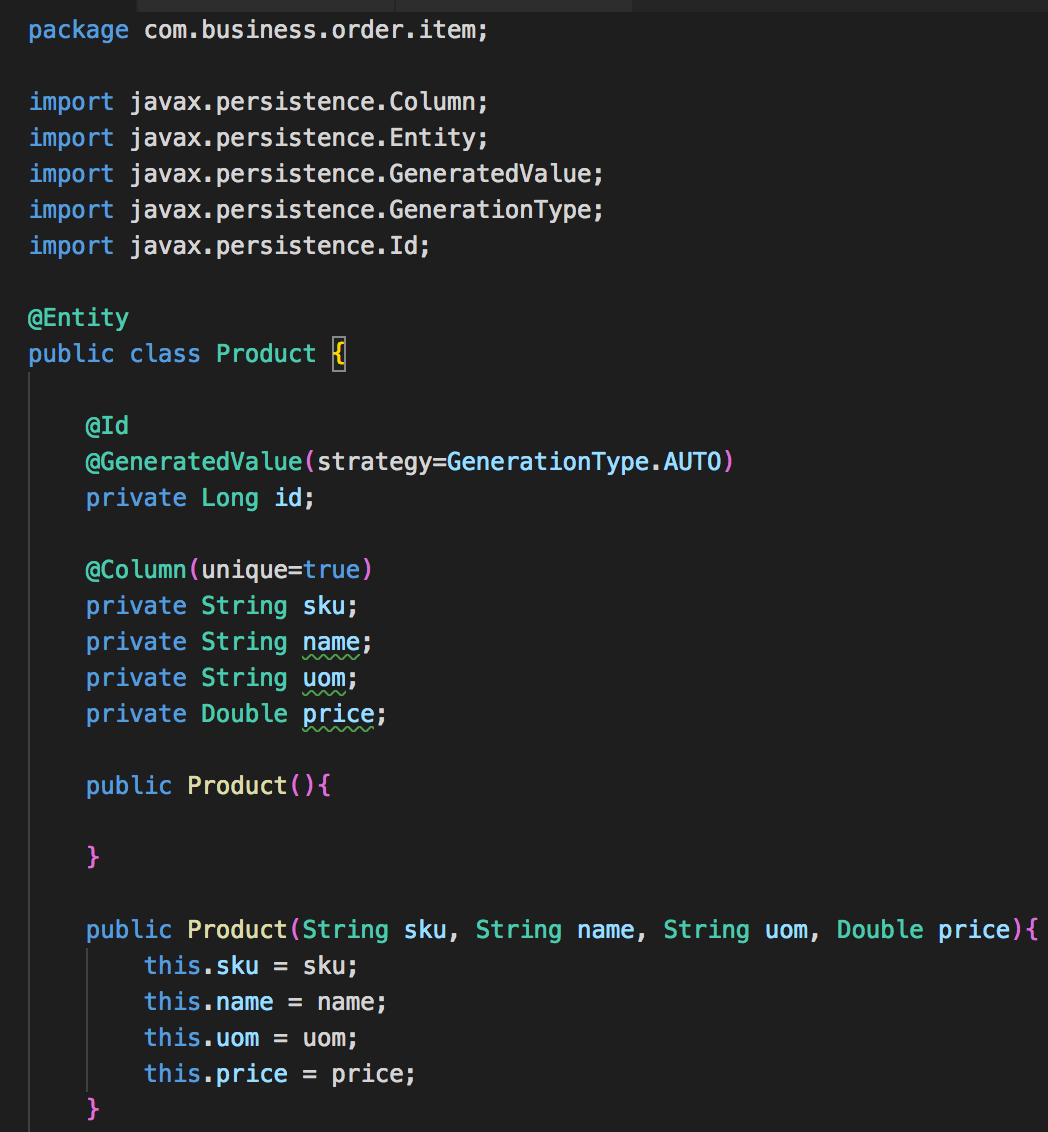


**WORKSHOP#3** สร้าง Entity ชื่อ Product โดยมี field ดังนี้

* Long id เป็น Auto generate key
* String sku
* String name
* String uom
* Double price

**วิธีการ**

* สร้าง Class ชื่อ Product ภายใต้ package com.business.order.item



\*\* หมายเหตุ\*\* ต้องสร้าง method set, get ให้ครบทุก field ด้วยนะครับ

**WORKSHOP#4** สร้าง Repository ของ Entity Product

**วิธีการ**

* สร้าง interface ชื่อ ProductRepository ภายใต้ package com.business.order.item และ extends interface CrudRepository ดังนี้

package com.business.order.item;

import org.springframework.data.repository.CrudRepository;

public interface ProductRepository extends CrudRepository<Product, Long> {

}

\*\*หมายเหตุ\*\*

Product คือชื่อ Entity Class

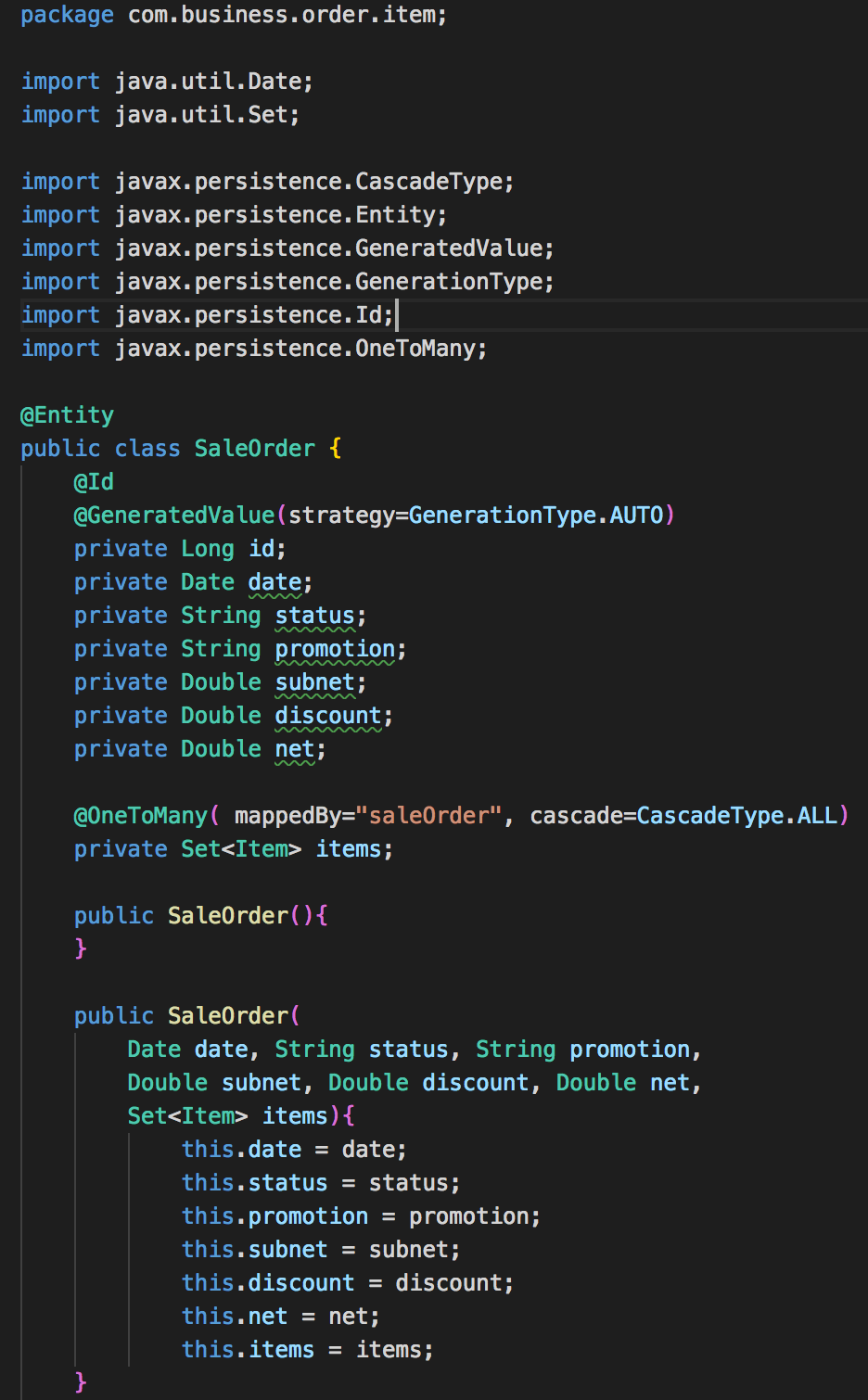
Long คือ Type ของ Id ของ Entity Class (แนะนำว่าให้ใช้ Long)

**WORKSHOP#5** สร้าง Entity ชื่อ SaleOrder โดยมี field ดังนี้

* Long id เป็น Auto generate key
* Date date
* String status
* String promotion
* Double subnet
* Double discount
* Double net
* Set<Item> items \*\*OneToMany กับ Item\*\*

**วิธีการ**

* สร้าง Class ชื่อ SaleOrder ภายใต้ package com.business.order.item



\*\* หมายเหตุ\*\* ต้องสร้าง method set, get ให้ครบทุก field ด้วยนะครับ

**WORKSHOP#6** สร้าง Repository ของ Entity SaleOrder

**วิธีการ**

* สร้าง interface ชื่อ SaleOrderRepository ภายใต้ package com.business.order.item และ extends interface CrudRepository ดังนี้

package com.business.order.item;

import org.springframework.data.repository.CrudRepository;

public interface SaleOrderRepository extends CrudRepository<SaleOrder, Long> {

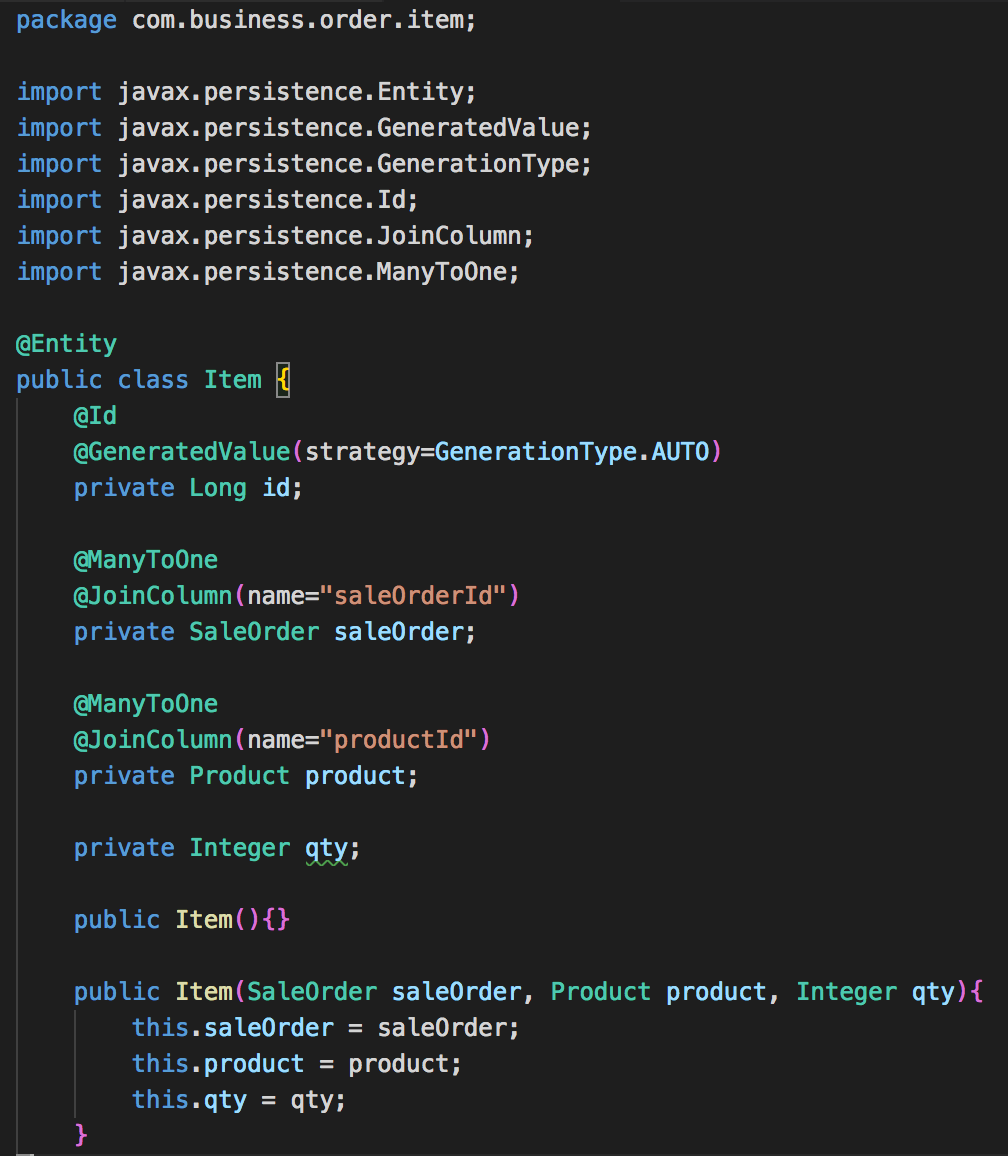
}

**WORKSHOP#7** สร้าง Entity ชื่อ Item โดยมี field ดังนี้

* Long id เป็น Auto generate key
* SaleOrder saleOrder \*\*ManyToOne\*\*
* Product product \*\*ManyToOne\*\*
* Integer qty

**วิธีการ**

* สร้าง Class ชื่อ Item ภายใต้ package com.business.order.item



\*\* หมายเหตุ\*\* ต้องสร้าง method set, get ให้ครบทุก field ด้วยนะครับ

**WORKSHOP#8** สร้าง Repository ของ Entity Item

**วิธีการ**

* สร้าง interface ชื่อ ItemRepository ภายใต้ package com.business.order.item และ extends interface CrudRepository ดังนี้

package com.business.order.item;

import org.springframework.data.repository.CrudRepository;

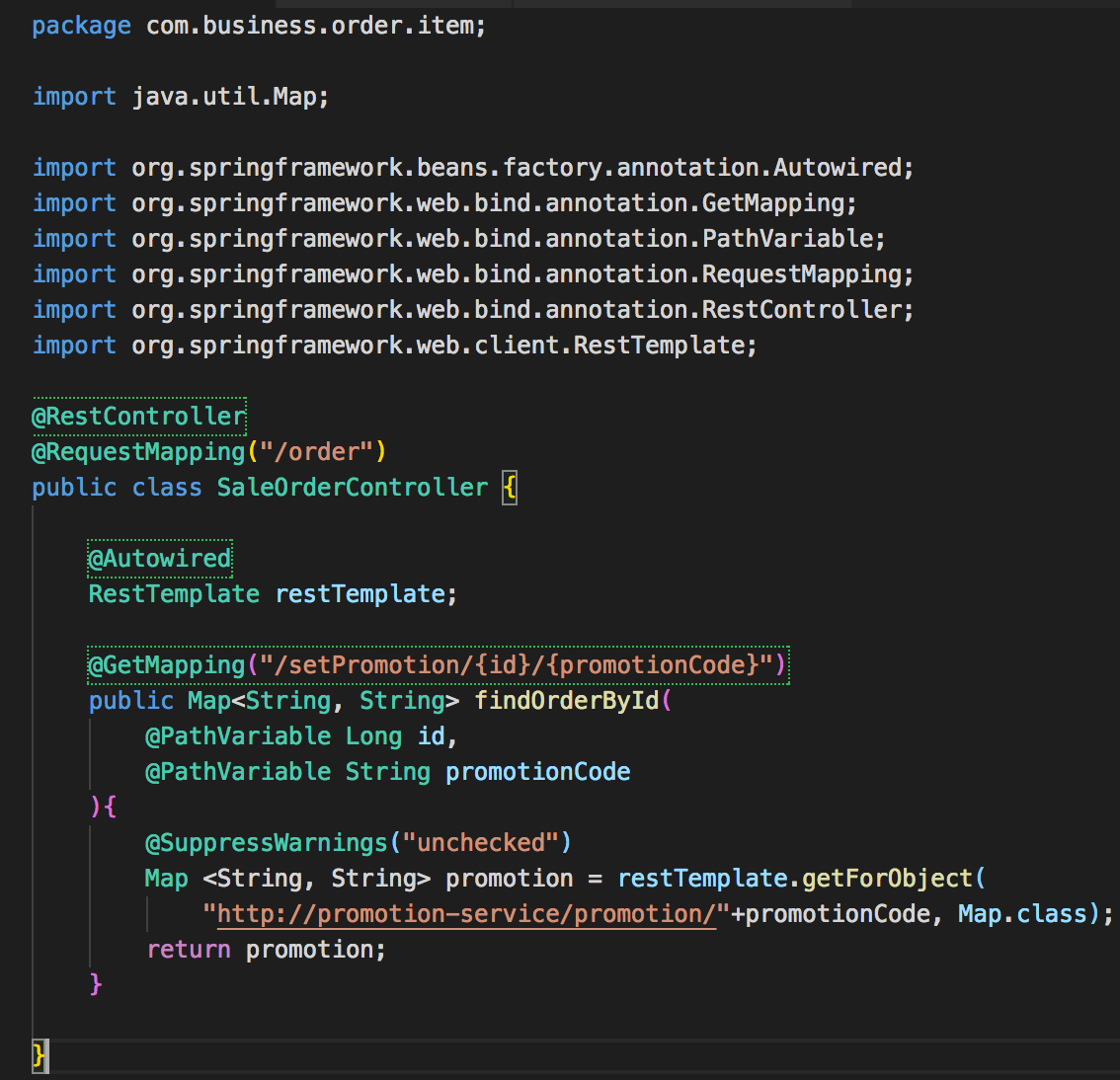
public interface ItemRepository extends CrudRepository<Item, Long> {

}

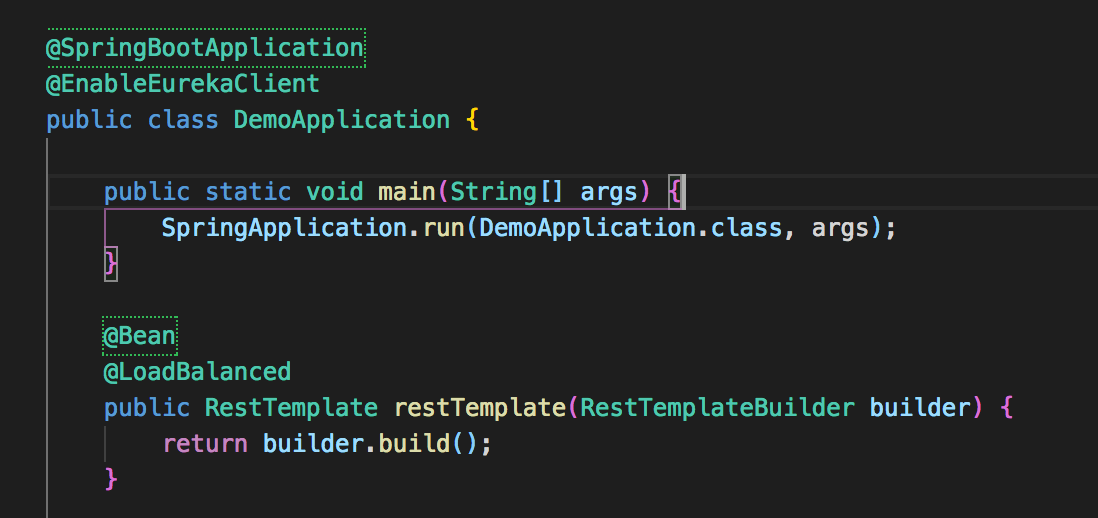
**WORKSHOP#9** สร้าง Controller ชื่อ SaleOrderController เพื่อทำเป็น rest api

**วิธีการ**

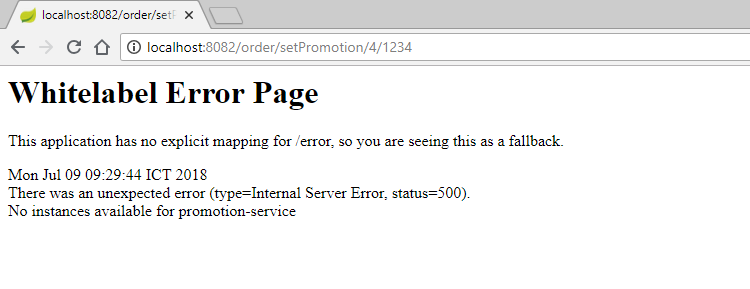
* สร้าง class ชื่อ SaleOrderController ภายใต้ package com.business.order.item โดยมี get method ชื่อ "/order/ setPromotion/{id}/{promotionCode} " โดย ภายใน method จะมีการเรียกไป service promotion-service เพื่อดึงค่า promotion มาใช้ใน controller โดยมี code ดังนี้



* เพิ่ม Bean RestTemplate เพื่อให้สามารถเรียกใช้ method ของ service อื่นๆ ภายในระบบโดยต้องไปเพิ่มใน DemoApplication ดังนี้



* start service ด้วยคำสั่งดังนี้
  + mvn spring-boot:run
* ทดสอบการทำงานด้วยการเข้าไปเรียกใช้ผ่าน url ดังนี้
  + <http://localhost:8082/order/setPromotion/4/1234>
  + จะได้ผลลัพธ์ดังนี้



HYSTRIX

**WORKSHOP#1** config hystrix เข้าไปที่ project order\_service เพื่อให้สามารถ handle error ในกรณีที่มีการเรียกไปที่ service อื่นแล้ว service นั้นๆ มีปัญหา

* แก้ไข file pom.xml ของ project order\_service โดยเพิ่มรายการเข้าไปดังนี้

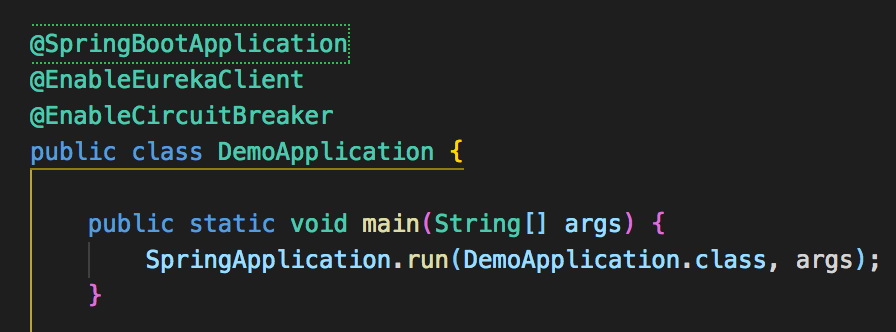
<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-hystrix</artifactId>

</dependency>

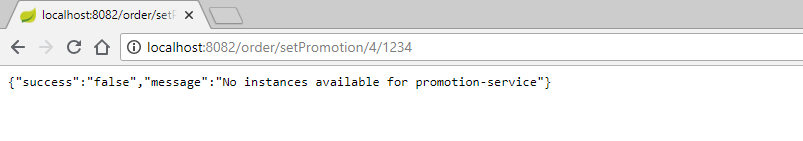
* เพิ่ม @EnableCircuitBreaker เข้าไปที่ DemoApplication ดังนี้



* แก้ไข code ใน class SaleOrderController โดยเพิ่ม @HystrixCommand กับเพิ่ม method getPromotionFallback เข้าไปดังนี้



* start service ด้วยคำสั่งดังนี้
  + mvn spring-boot:run
* ทดสอบการทำงานด้วยการเข้าไปเรียกใช้ผ่าน url ดังนี้
  + <http://localhost:8082/order/setPromotion/4/1234>
  + จะได้ผลลัพธ์ดังนี้



SLEUTH

**WORKSHOP#1** config sleuth เข้าไปที่ project order\_service เพื่อใช้ในการช่วยเก็บ log ในกรณีที่มีการเรียกไปที่ service อื่น

* แก้ไข file pom.xml ของ project order\_service โดยเพิ่มรายการเข้าไปดังนี้

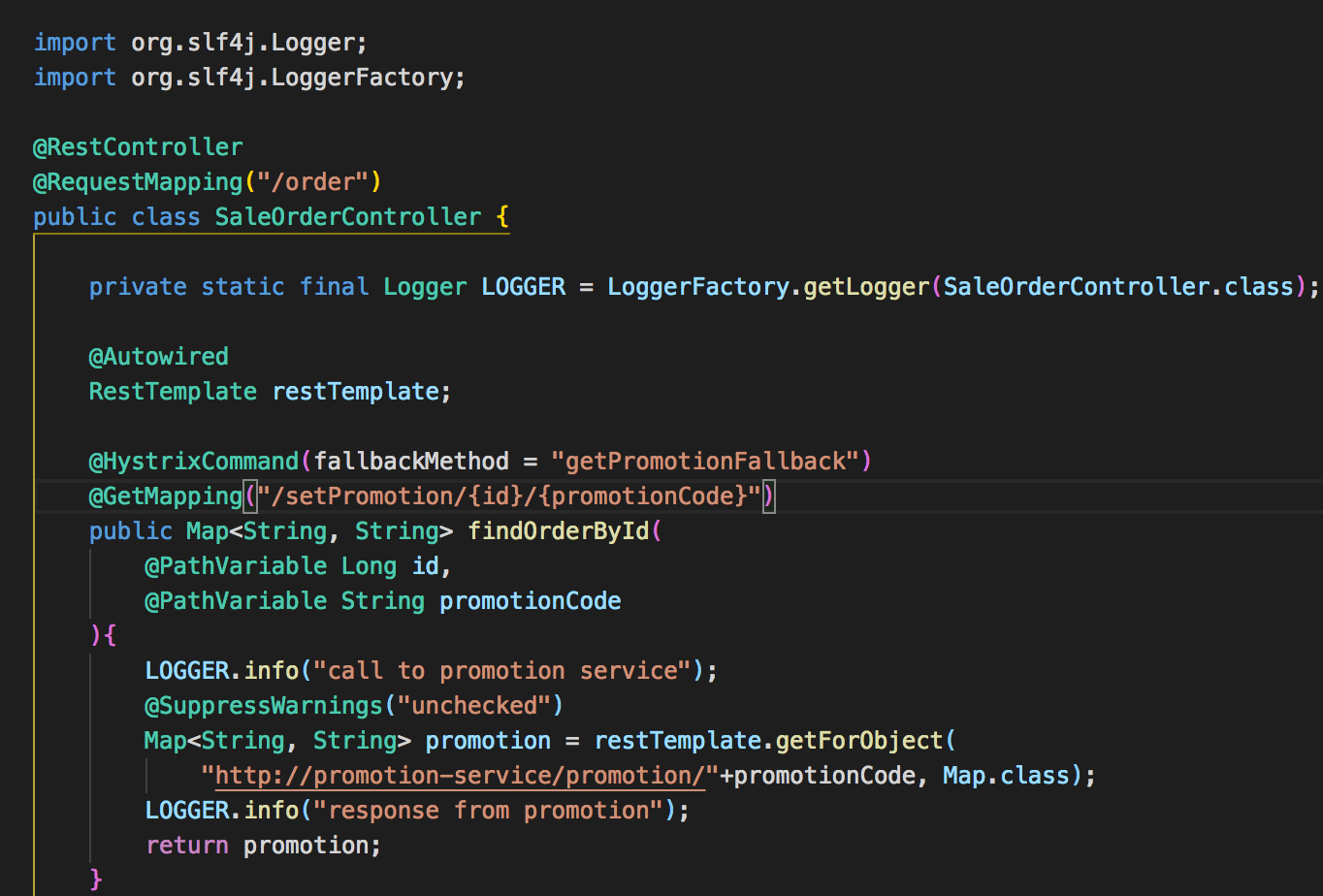
<dependency>

<groupId>org.springframework.cloud</groupId>

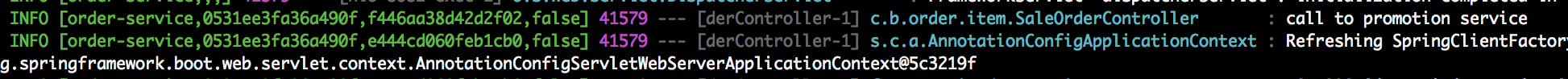
<artifactId>spring-cloud-starter-sleuth</artifactId>

</dependency>

* เพิ่ม code ใน class SaleOrderController ดังนี้



* restart order\_service
* ทดสอบการทำงานด้วยการเข้าไปเรียกใช้ผ่าน url ดังนี้
  + <http://localhost:8082/order/setPromotion/4/1234>
  + จะมี log แสดงใน console ดังภาพ



**Exercise**

**exercise#1** ให้ตั้งค่าที่ zuul server ให้สามารถเรียกใช้งาน order\_service ผ่าน url ดังนี้ได้ [http://localhost:8762/order/<url](http://localhost:8762/order/%3curl) ของ order\_service>

**\*\*แนวทาง\*\***

* แก้ไข file properties ของ zuul server ให้สามารถมองเห็น order-service

**exercise#2** ให้สร้าง microservice โดยมีคุณสมบัติ ดังนี้

* project name = promotion\_service
* application name = promotion-service
* port = 8083
* group id = com.business
* artifect id = promotion
  + dependencies ดังนี้
    - web
    - actuator
    - jpa (sql)
    - mysql หรือ h2
    - devtools
    - config client
    - eureka discovery
    - hystrix
    - sleuth
* Entity ดังนี้
  + Promotion
    - Long id
    - String code
    - Double discount
* Repository ดังนี้
  + PromotionRepository
* สร้างข้อมูลเพื่อทำการทดสอบจำนวน 3 รายการดังนี้
  + code = FREE100, discount = 100
  + code = FREE200, discount = 200
  + code = FREE150, discount = 150
* Rest Api ดังนี้
  + GET /promotion/{promotionCode}
    - รับค่ามาเป็น promotion code
    - ให้หาค่า discount จาก code ที่ได้รับ
    - ส่งค่ากลับเป็น json ในกรณีหาเจอ
      * { valid : true, discount : 100 }
    - ส่งค่ากลับเป็น json ในกรณีหาไม่เจอ
      * { valid : false }

**exercise#3** แก้ไข code ของ service order\_service ให้ส่งค่ามาเป็น ดังนี้

* เมื่อเรียกไป <http://localhost:8082/order/setPromotion/4/FREE100> คาดหวังผลลัพธ์ดังนี้
  + { id : 4, promotion : FREE100, subnet : 330, discount : 100, net : 230}
* เมื่อเรียกไป http://localhost:8762/order/setPromotion/4/FREE100 โดยส่ง Authentication ไปคาดหวังผลลัพธ์ดังนี้
  + { id : 4, promotion : FREE100, subnet : 330, discount : 100, net : 230}