

Workshop

MICROSERVICE
WITH
SPRING BOOT

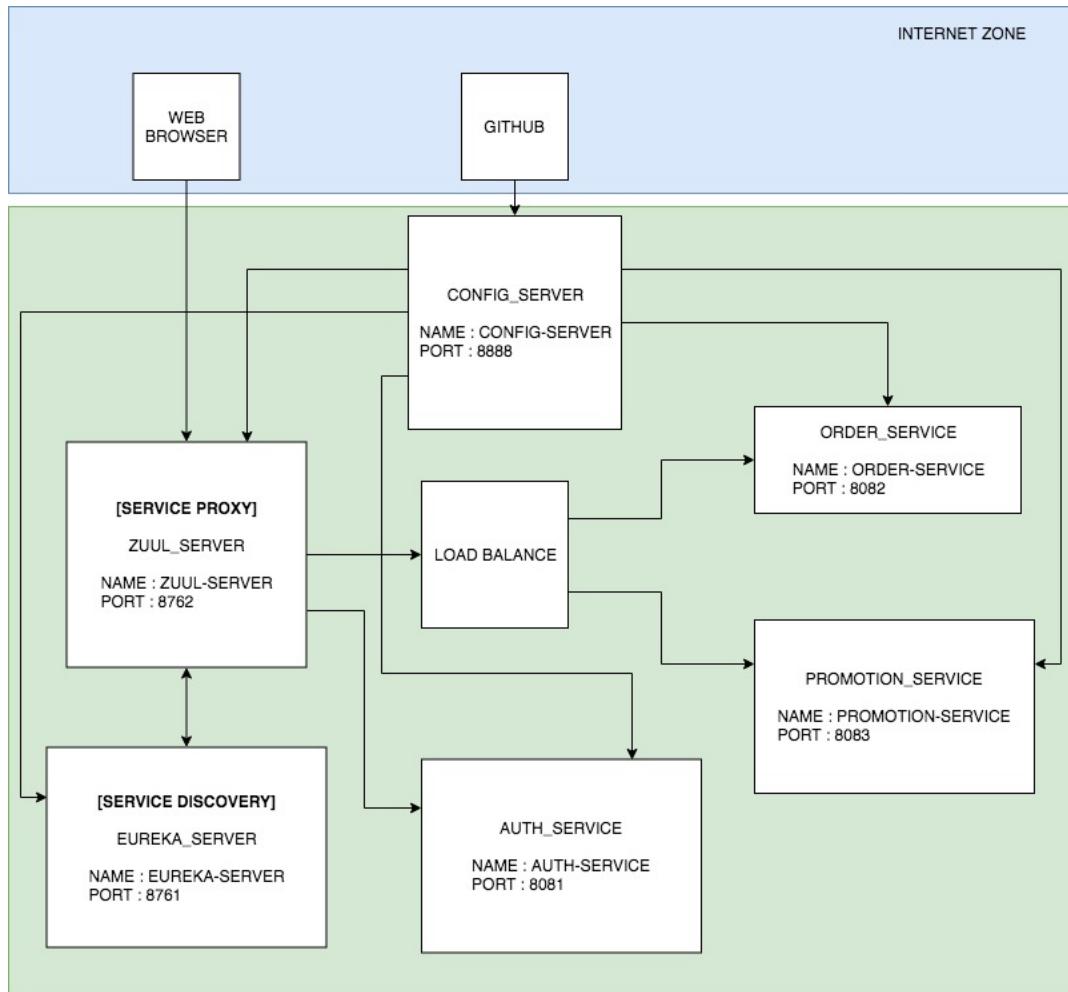
by
PnP Solution

www.pnpsw.com

facebook.com/pnpsolution

MICROSERVICE WITH SPRING BOOT

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
```

```
$ docker run -d -p 3306:3306 --name my_db -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=my_db -e MYSQL_PASSWORD=my_db mysql:5.5.60
```

- เมื่อ run เสร็จแล้วให้กดสอบบหลังการติดตั้งดังนี้
 - docker exec -it my_db bash
- เมื่อเข้าไปใน prompt ของ my_db container แล้วให้กดลงเข้าไปดูใน mysql db ด้วยคำสั่งดังนี้
 - mysql my_db -u my_db -p
 - ใส่ password เป็น my_db
- หลังจากนั้นให้ใช้คำสั่ง ดังนี้เพื่อดูว่ามี database ที่เราสร้างไว้หรือไม่ ด้วยคำสั่งดังนี้
 - show databases;
 - ผลลัพธ์จะได้ดังนี้

```
mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| my_db          |
+-----+
2 rows in set (0.01 sec)
```

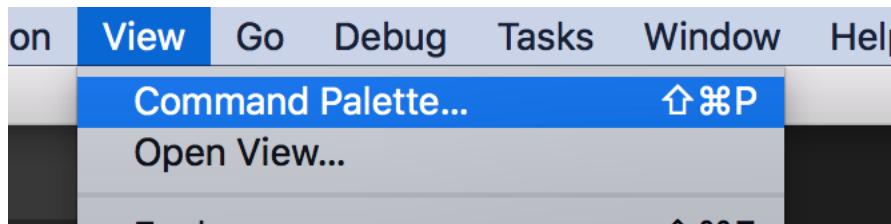
AUTH SERVICE

WORKSHOP#1 สร้าง project ชื่อ auth_service โดยมีคุณสมบัติ ดังนี้

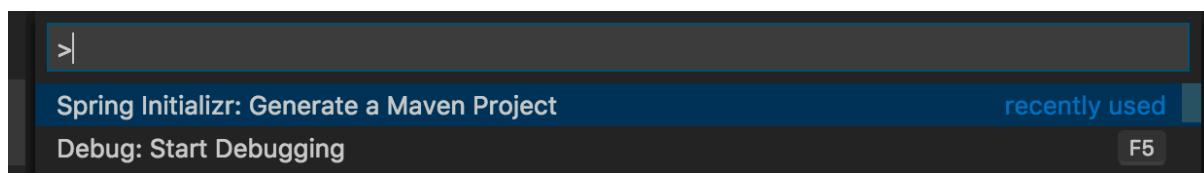
- group id = com.business
- artifact id = auth
- dependencies ดังนี้
 - web
 - actuator
 - jpa (sql)
 - mysql หรือ h2
 - devtools

วิธีการ

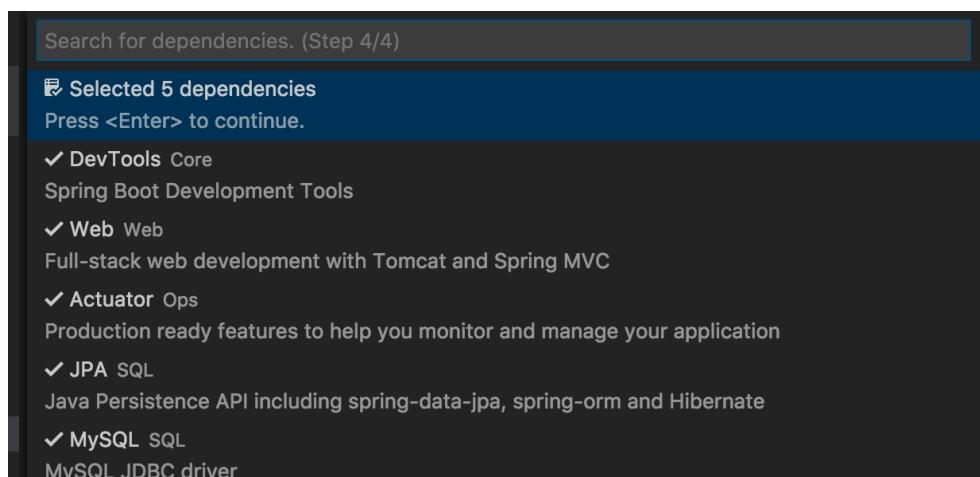
- เข้า Program Visual Studio Code แล้ว ไปที่ เมนู View > Command Palette... ดังภาพ



- เลือก Spring Initializr: Generate a Maven Project ดังภาพ



- ในขั้นตอนการเลือก Dependencies ให้เลือกดังภาพ



WORKSHOP#2 ตั้งค่าการเชื่อมต่อ database ไปยัง mysql หรือ h2

วิธีการ

- เปิด file application.properties และใส่ข้อมูลดังนี้

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

```
package com.business.auth.user;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class User {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String email;
    private String name;
    private String password;
    private String role;

    public User() {
    }

    public User(String email, String name, String password, String role){
        this.email = email;
        this.name = name;
        this.password = password;
        this.role = role;
    }
    /**
     * getter and setter section
     */
}
```

**** หมายเหตุ** ต้องสร้าง method set, get ให้ครบถ้วนทุก field ด้วยนะครับ**

WORKSHOP#4 สร้าง Repository ของ Entity User

วิธีการ

- สร้าง interface ชื่อ UserRepository ภายใน 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#5 เปลี่ยนชื่อ Class DemoApplication เป็น AuthApplication

วิธีการ

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

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

- แก้ไข name ใน file pom.xml ดังภาพ

```
<packaging>jar</packaging>

<name>auth</name>
<description>Demo project for Spring Boot</description>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.3.RELEASE</version>
```

WORKSHOP#6 สร้าง Bean เพื่อกำกั้น 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

MICROSERVICE WITH SPRING BOOT

```
package com.business.auth;

import com.business.auth.user.*;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class AuthApplication {

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

    @Bean
    public CommandLineRunner initialUserData(UserRepository repository) {
        return (args) -> {
            // save a couple of customers
            User user = new User("sommai.k@gmail.com", "sommai.k", "1234", "ADMIN");
            repository.save(user);
        };
    }
}
```

MICROSERVICE WITH SPRING BOOT

WORKSHOP#7 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
- กัน 4 method เมื่อทำสำเร็จให้ return ออกมาในรูปแบบ json ดังนี้

```
{  
    success : true  
    data : <json>  
}
```
- กัน 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;
}
```

MICROSERVICE WITH SPRING BOOT

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

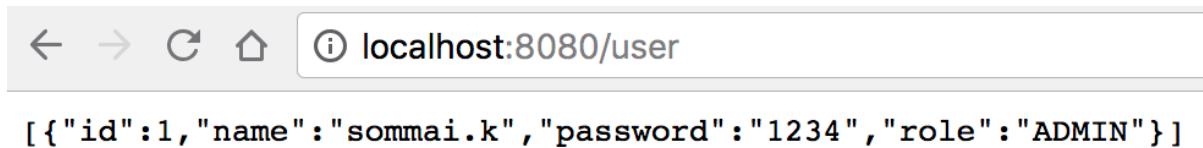
```
@PutMapping("/{id}")
public Map<String, Object> updateUser(
    @RequestBody Map<String, String> body,
    @PathVariable Long id
) {
    Map<String, Object> res = new HashMap<>();
    if(userRepository.existsById(id)){
        User other = new User(
            body.get("email"),
            body.get("user"),
            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;
}
```

MICROSERVICE WITH SPRING BOOT

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

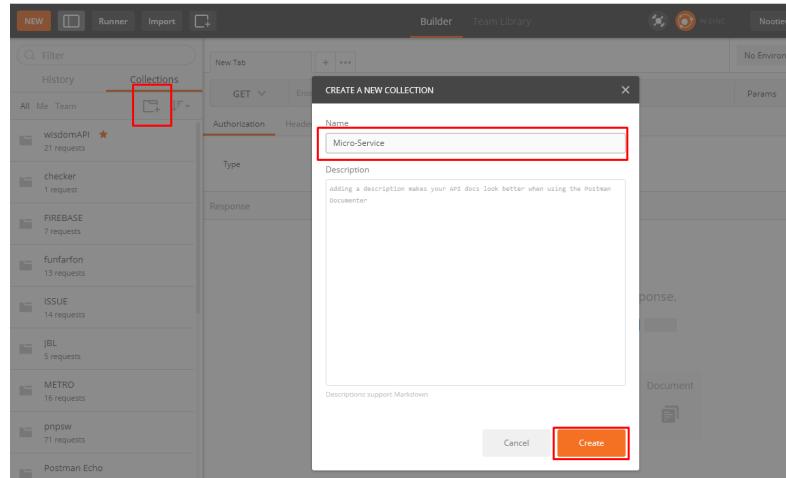
```
@DeleteMapping("/{id}")
public Map<String, Object> deleteUser(
    @PathVariable Long id
) {
    Map<String, Object> res = new HashMap<>();
    if(userRepository.existsById(id)){
        userRepository.deleteById(id);
        res.put("success", true);
        res.put("data", id);
    }else{
        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

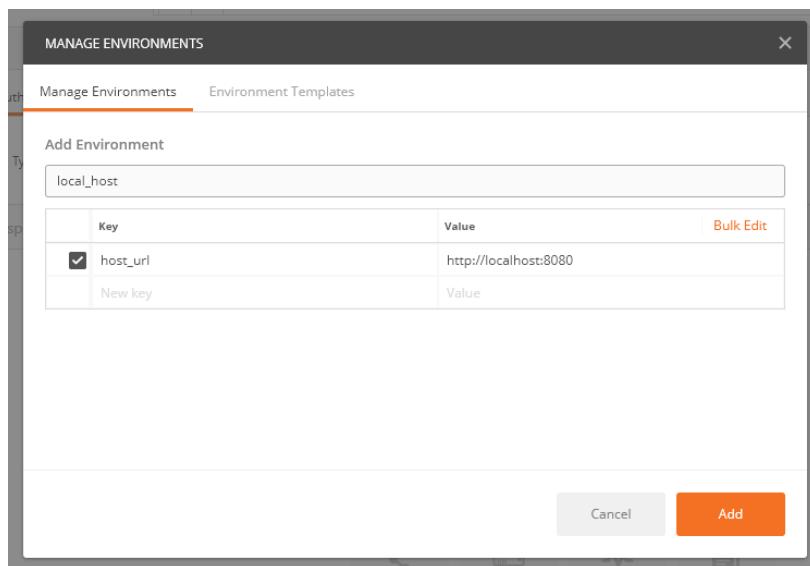
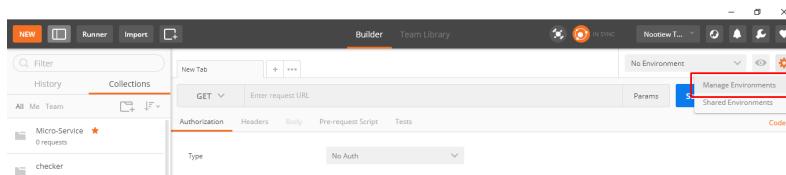


MICROSERVICE WITH SPRING BOOT

- ทดลองทดสอบ api ทั้งหมด ผ่าน postman โดยมีการตั้งค่าดังนี้
 - เข้าหน้าจอ postman สร้าง collection ชื่อ MicroService



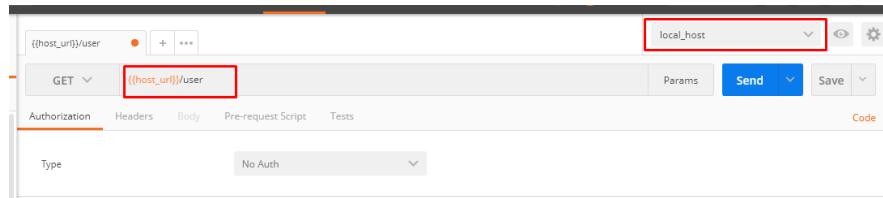
- สร้าง environment ชื่อว่า local_host มีค่าดังนี้
 - host_url = <http://localhost:8080>



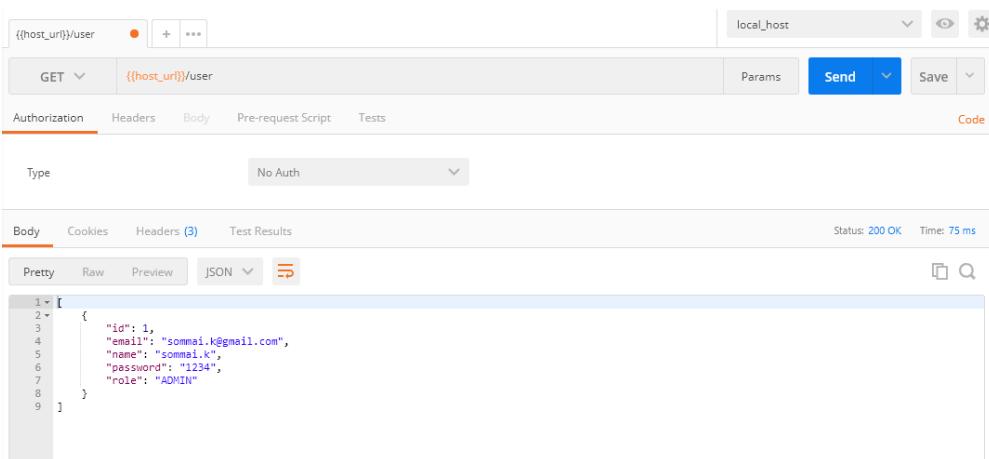
MICROSERVICE WITH SPRING BOOT

- สร้าง tab เพื่อทดสอบ method get โดยใช้ค่า url ดังนี้

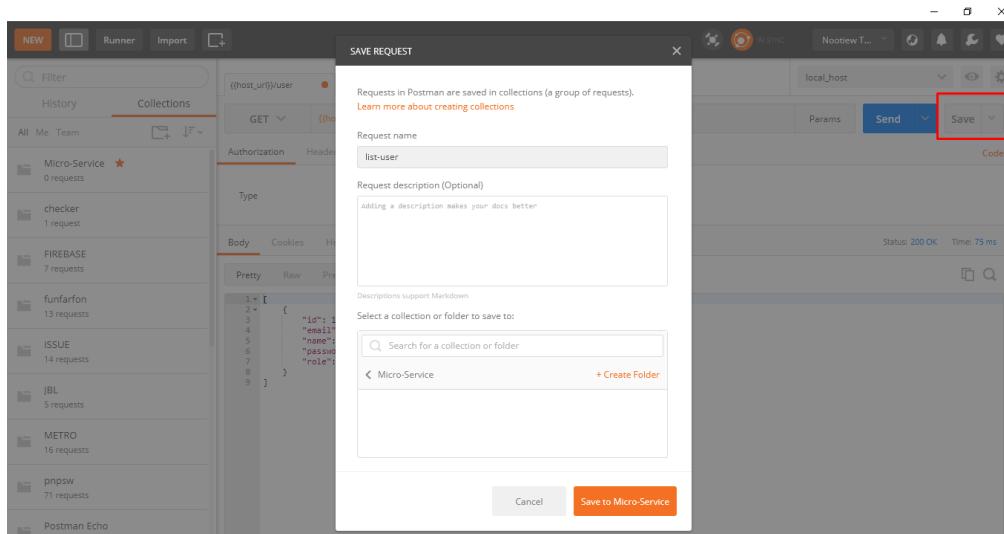
- {{host_url}}/user



- กดรับจะได้ผลลัพธ์ดังนี้



- บันทึก link สำหรับการเทสไว้ใน collection โดยการกด Save แล้วเลือก collection ที่ต้องการ

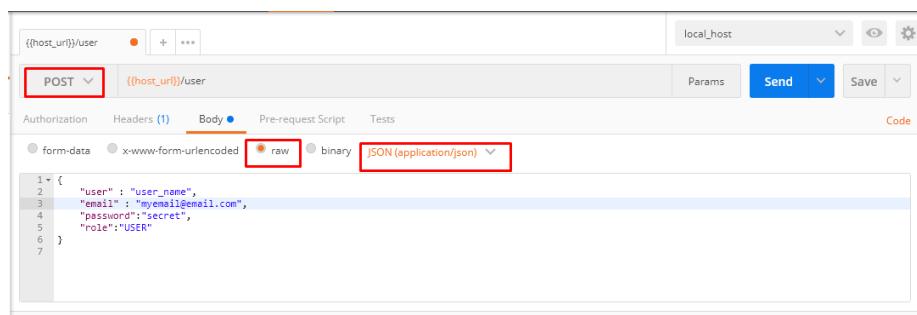


MICROSERVICE WITH SPRING BOOT

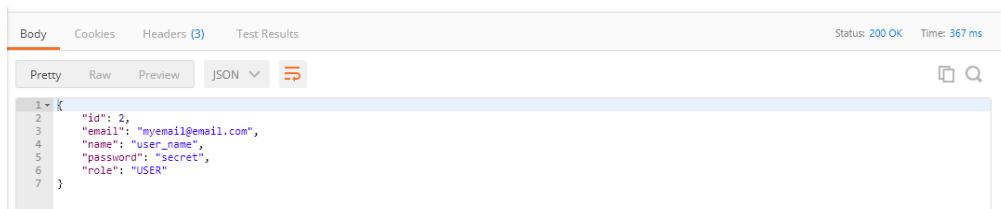
- สร้าง tab เพื่อทดสอบ method post โดยใส่ค่า url ดังนี้

- {{host_url}}/user
- body มีค่าดังนี้

```
{  
  "user" : "user_name",  
  "email" : "myemail@email.com",  
  "password": "secret",  
  "role": "USER"  
}
```



- กดรันจะได้ผลลัพธ์ดังนี้

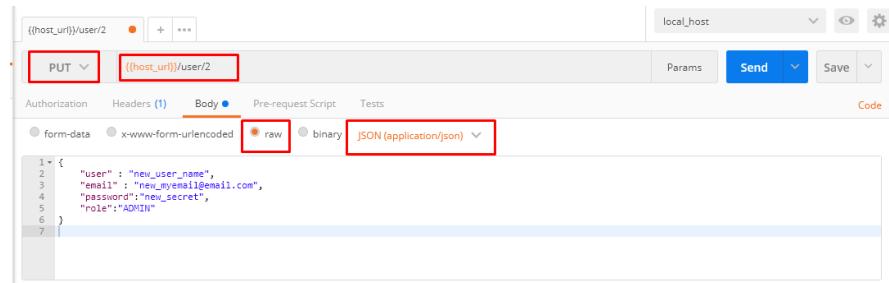


- สร้าง tab เพื่อทดสอบ method put โดยใส่ค่า url ดังนี้

- {{host_url}}/user/2
- body มีค่าดังนี้

```
{  
  "user" : "new_user_name",  
  "email" : "new_myemail@email.com",  
  "password": "new_secret",  
  "role": "ADMIN"  
}
```

MICROSERVICE WITH SPRING BOOT



■ กดรันจะได้ผลลัพธ์ดังนี้

The screenshot shows the Postman interface after sending the PUT request. The status is 200 OK and the time taken is 208 ms. The response body is:

```
1+ {  
2   "data": {  
3     "id": 2,  
4     "email": "new_myemail@email.com",  
5     "name": "new_user_name",  
6     "password": "new_secret",  
7     "role": "ADMIN"  
8   },  
9   "success": true  
10 }
```

○ สร้าง tab เพื่อทดสอบ method delete โดยใส่ค่า url ดังนี้

■ {{host_url}}/user/1

The screenshot shows the Postman interface with a DELETE request to the URL {{host_url}}/user/1. The method is highlighted in red.

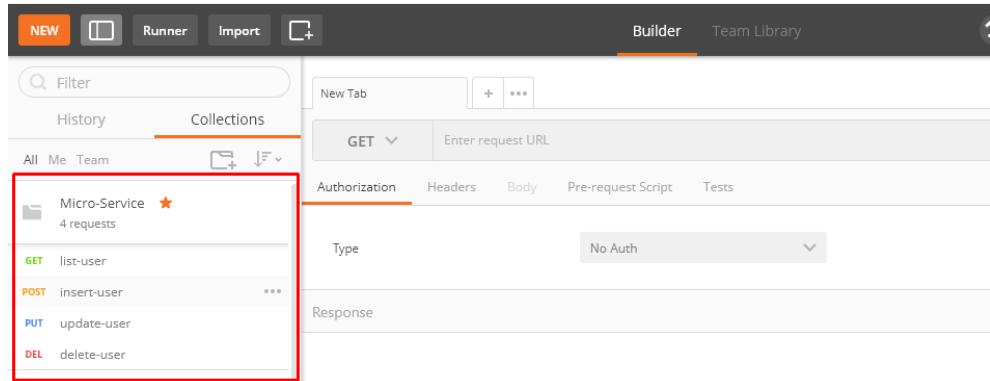
■ กดรันจะได้ผลลัพธ์ดังนี้

The screenshot shows the Postman interface after sending the DELETE request. The status is 200 OK and the time taken is 98 ms. The response body is:

```
1+ {  
2   "data": 2,  
3   "success": true  
4 }
```

MICROSERVICE WITH SPRING BOOT

- เมื่อทดสอบครบถ้วน Method และทำการบันทึกไว้ใน collection จะได้ผลลัพธ์ดังนี้



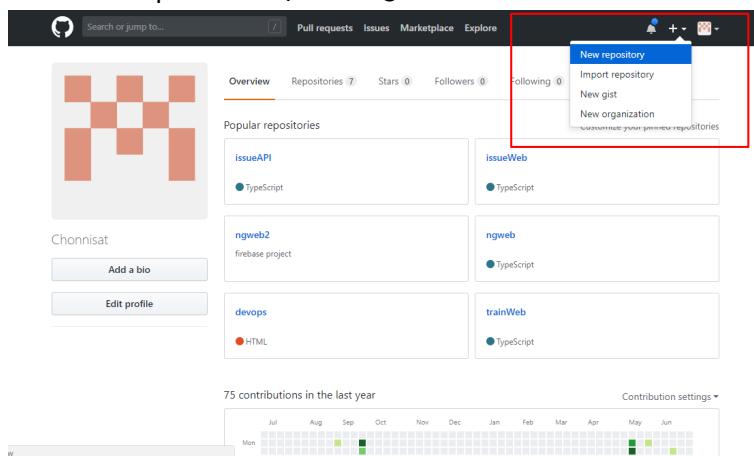
MICROSERVICE WITH SPRING BOOT

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 และเลือกเป็น option เป็น public กดปุ่ม create repository

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner Repository name

Great repository names are short and memorable. Need inspiration? How about `effective-octo-train`.

Description (optional)

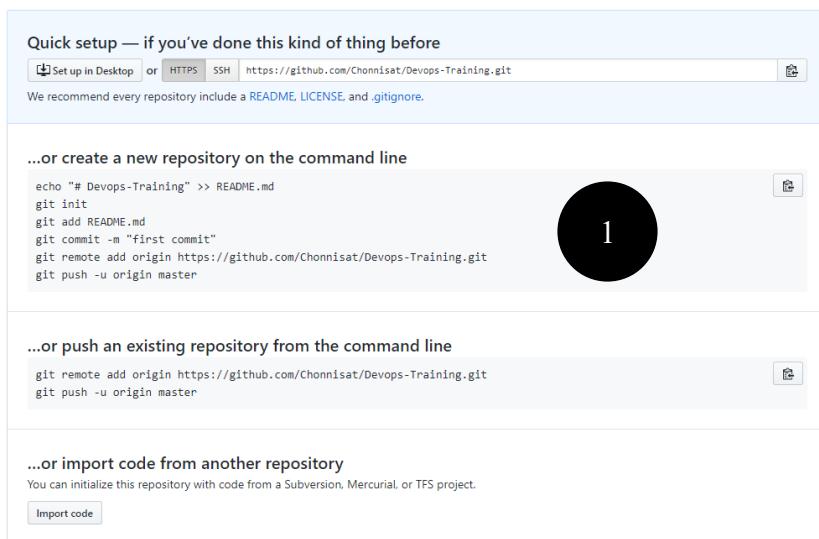
Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

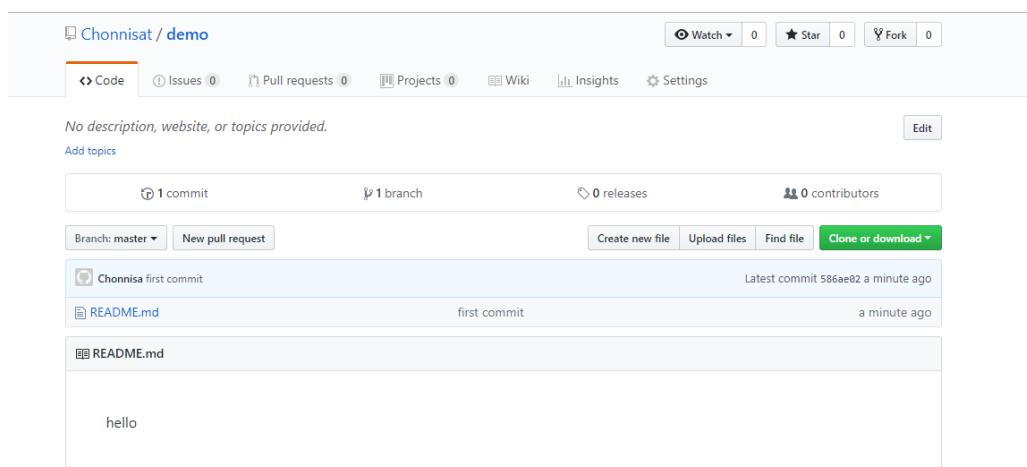
MICROSERVICE WITH SPRING BOOT

- จะได้ผลลัพธ์ดังนี้



The screenshot shows the GitHub 'Quick setup' interface. It includes three sections: 1. 'Set up in Desktop' or 'HTTPS' (selected) or 'SSH' with URL <https://github.com/Chonnisat/Devops-Training.git>. 2. '...or create a new repository on the command line' with terminal commands: echo "# Devops-Training" >> README.md, git init, git add README.md, git commit -m "first commit", git remote add origin https://github.com/Chonnisat/Devops-Training.git, git push -u origin master. 3. '...or push an existing repository from the command line' with terminal commands: git remote add origin https://github.com/Chonnisat/Devops-Training.git, git push -u origin master. A large black circle with the number '1' is overlaid on the first section.

- 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
 - git push -u origin master
- ผลลัพธ์ที่ได้เมื่อเราเข้าไปที่หน้า project เราอีกครั้งบน github

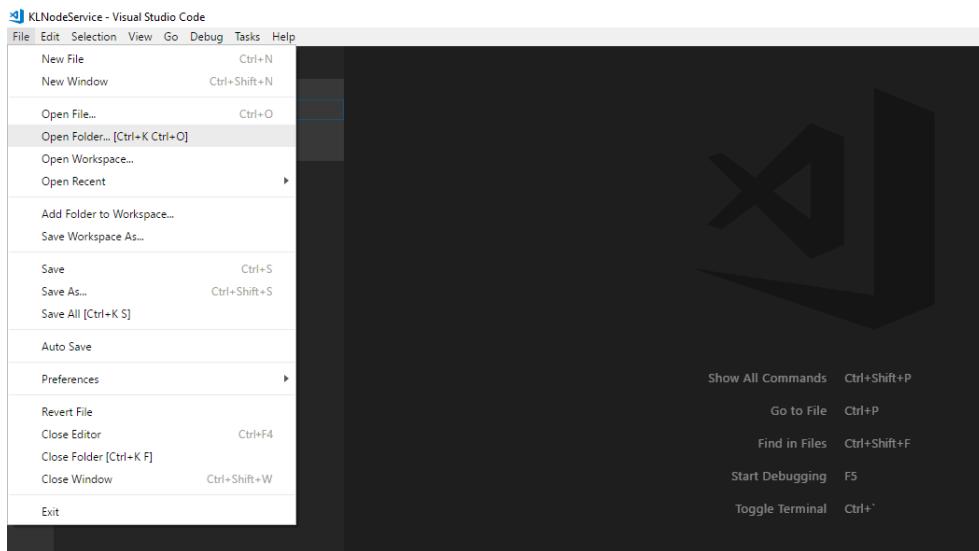


The screenshot shows the GitHub repository details for 'Chonnisat / demo'. It includes:

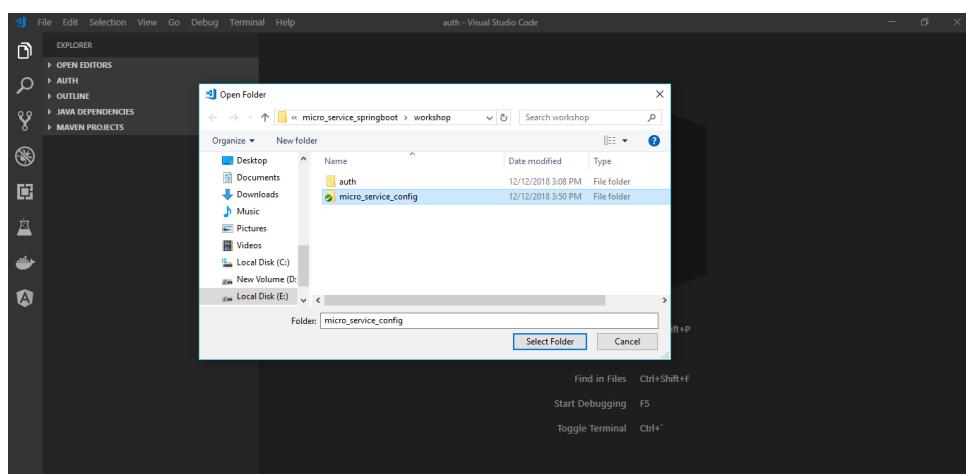
- Header: Watch 0, Star 0, Fork 0
- Code tab selected.
- No description, website, or topics provided.
- 1 commit, 1 branch, 0 releases, 0 contributors.
- Branch: master, New pull request, Create new file, Upload files, Find file, Clone or download.
- Commit history:
 - Chonnisat first commit (a minute ago)
 - README.md (first commit, a minute ago)
 - Content: hello

MICROSERVICE WITH SPRING BOOT

- เปิด program visual studio code เพื่อกำกับ project
 - เปิด project โดยการกดปุ่ม open



- แล้วเลือก folder ที่เราทำการสร้าง project ไว้



MICROSERVICE WITH SPRING BOOT

- 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
```

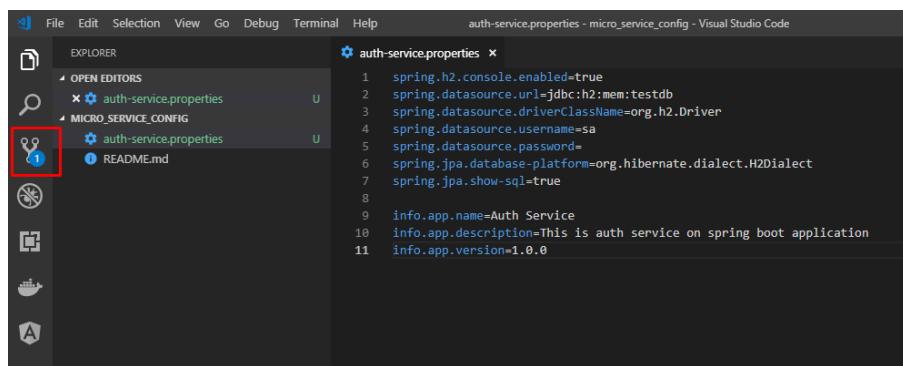
```
spring.h2.console.enabled=true
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

```
info.app.name=Auth Service
```

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

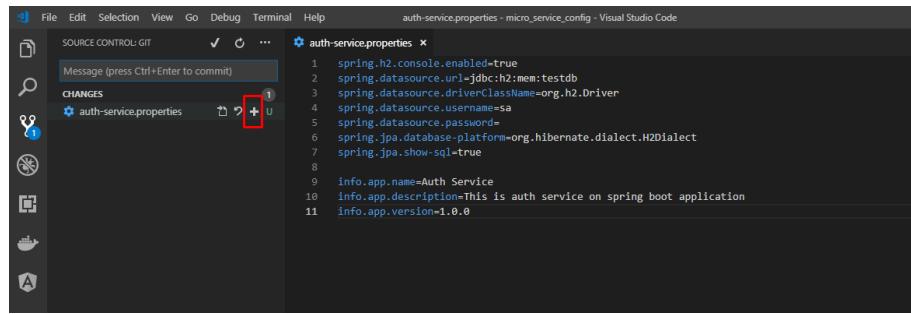
```
info.app.version=1.0.0
```

- sync code ขึ้น github ด้วย gui มีวิธีการดังนี้
 - เข้าไปในส่วนของการ sync code โดยการกดปุ่มดังนี้



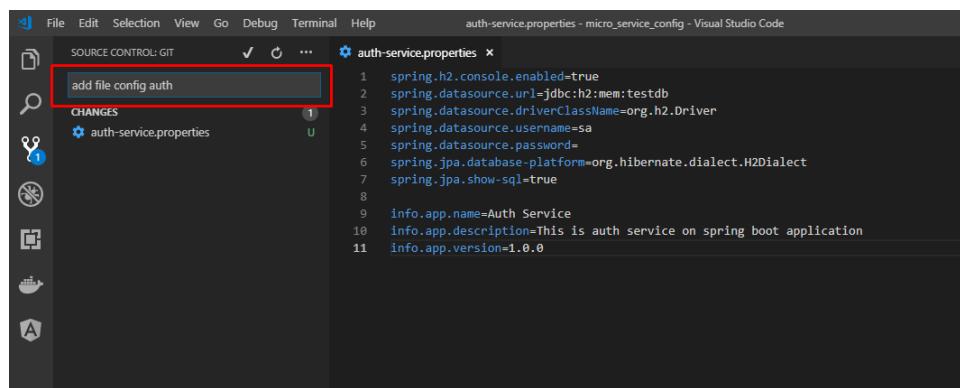
```
auth-service.properties
1  spring.h2.console.enabled=true
2  spring.datasource.url=jdbc:h2:mem:testdb
3  spring.datasource.driverClassName=org.h2.Driver
4  spring.datasource.username=sa
5  spring.datasource.password=
6  spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
7  spring.jpa.show-sql=true
8
9  info.app.name=Auth Service
10 info.app.description=This is auth service on spring boot application
11 info.app.version=1.0.0
```

MICROSERVICE WITH SPRING BOOT



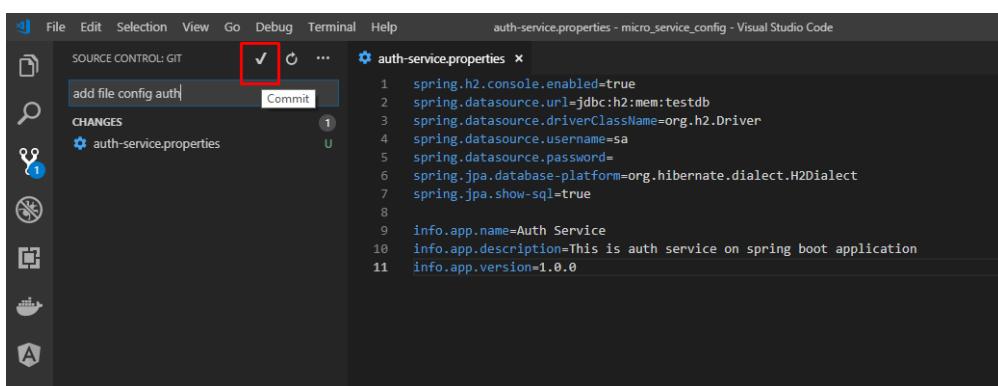
```
auth-service.properties - micro_service.config - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
auth-service.properties x
SOURCE CONTROL: GIT ✓ ⚡ ...
Message (press Ctrl+Enter to commit)
CHANGES
auth-service.properties + 0
1 spring.h2.console.enabled=true
2 spring.datasource.url=jdbc:h2:mem:testdb
3 spring.datasource.driverClassName=org.h2.Driver
4 spring.datasource.username=sa
5 spring.datasource.password=
6 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
7 spring.jpa.show-sql=true
8
9 info.app.name=Auth Service
10 info.app.description=This is auth service on spring boot application
11 info.app.version=1.0.0
```

- พิมพ์ message ที่ต้องการจะ commit



```
auth-service.properties - micro_service.config - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
auth-service.properties x
SOURCE CONTROL: GIT ✓ ⚡ ...
add file config auth
CHANGES
auth-service.properties 1
1 spring.h2.console.enabled=true
2 spring.datasource.url=jdbc:h2:mem:testdb
3 spring.datasource.driverClassName=org.h2.Driver
4 spring.datasource.username=sa
5 spring.datasource.password=
6 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
7 spring.jpa.show-sql=true
8
9 info.app.name=Auth Service
10 info.app.description=This is auth service on spring boot application
11 info.app.version=1.0.0
```

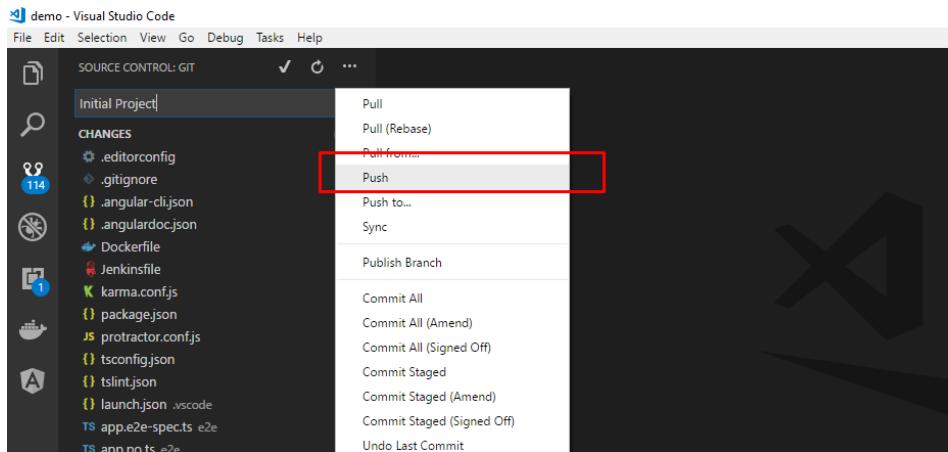
- กดเครื่องหมายถูกเพื่อกำกัร commit code



```
auth-service.properties - micro_service.config - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
auth-service.properties x
SOURCE CONTROL: GIT ✓ ⚡ ...
add file config auth Commit
CHANGES
auth-service.properties 1
1 spring.h2.console.enabled=true
2 spring.datasource.url=jdbc:h2:mem:testdb
3 spring.datasource.driverClassName=org.h2.Driver
4 spring.datasource.username=sa
5 spring.datasource.password=
6 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
7 spring.jpa.show-sql=true
8
9 info.app.name=Auth Service
10 info.app.description=This is auth service on spring boot application
11 info.app.version=1.0.0
```

- กดปุ่ม ... เพื่อกำกัร push code ขึ้นไปบน github

MICROSERVICE WITH SPRING BOOT



- ตรวจสอบผลลัพธ์ที่ได้จากการ sync code

A screenshot of a GitHub repository page. The repository name is "pnpsolution / micro_service_config". The page shows basic statistics: 2 commits, 1 branch, 0 releases, and 0 contributors. There are buttons for "Edit", "Manage topics", "Create new file", "Upload files", "Find file", and "Clone or download". Below these are two commit details:

- Chonnis Thiembundit add file config auth (Latest commit fba6425 a minute ago)
- README.md (first commit 22 minutes ago)
- auth-service.properties (add file config auth a minute ago)

****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

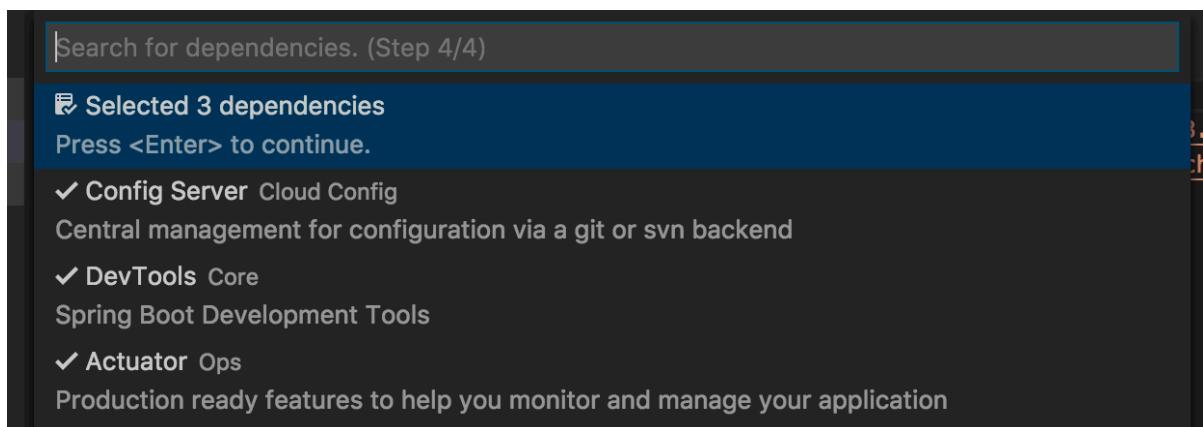
สมมุติว่าชื่อ project เป็น auth-service และรัน profile prod ก็จะไปหา file ชื่อ auth-service-**prod**.properties

CONFIG SERVER

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

วิธีการ

- สร้าง project ใหม่ชื่อ config_server โดยมีคุณสมบัติดังนี้
 - group id = com.business
 - artifact 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/<><>/micro_service_config
```

- เพิ่ม @EnableConfigServer เพื่อเปิดใช้งาน Configserver โดยเพิ่มไปใน code ดังนี้

MICROSERVICE WITH SPRING BOOT

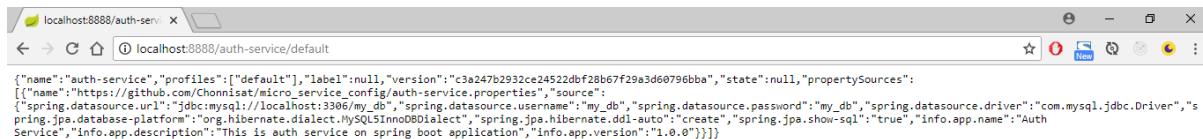
```
package com.business.config;

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

@SpringBootApplication
@EnableConfigServer
public class DemoApplication {

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

- 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 ที่ต้องการ

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

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

- เพิ่ม dependencies เข้าไป ดังนี้
- ```
</dependency>
<dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
```

- เพิ่ม dependencyManagement ต่อจาก dependencies เข้าไปดังนี้

```
<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>
```

- start auth-service ด้วยคำสั่งดังนี้
  - mvn spring-boot:run

## MICROSERVICE WITH SPRING BOOT

- ทดสอบการทำงานของ auth-service ด้วย postman

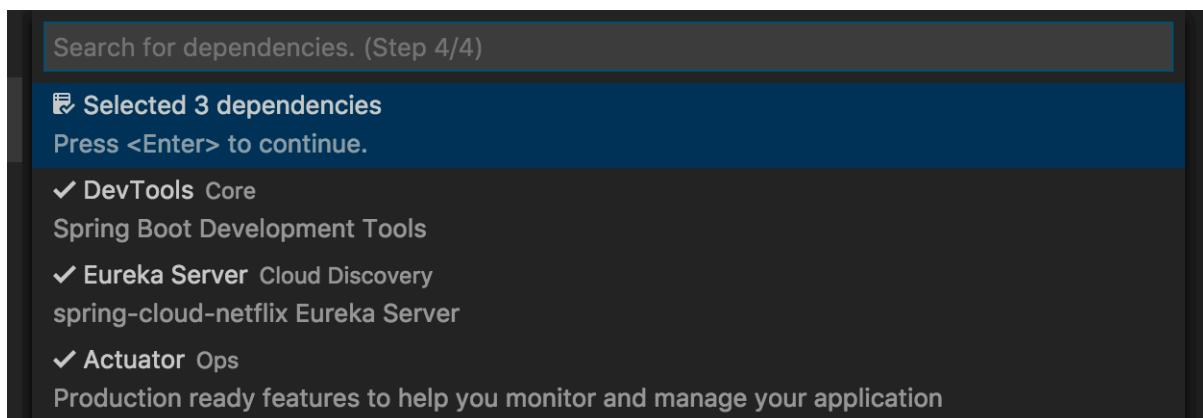
The screenshot shows the Postman application interface. A collection named 'list-user' is selected. A specific request named 'list-user' is displayed, showing a GET request to the URL `{host_url}/user`. The 'Authorization' tab is selected, showing 'No Auth'. The response status is 200 OK with a time of 905 ms. The response body is a JSON object:

```
1. [
2. {
3. "id": 1,
4. "email": "sonnai.k@gmail.com",
5. "name": "sonnai.k",
6. "password": "1234",
7. "role": "ADMIN"
8. }
9.]
```

## EUREKA SERVER

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

- สร้าง project ใหม่ชื่อ eureka\_server โดยมีคุณสมบัติดังนี้
  - group id = com.business
  - artifact 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

## MICROSERVICE WITH SPRING BOOT

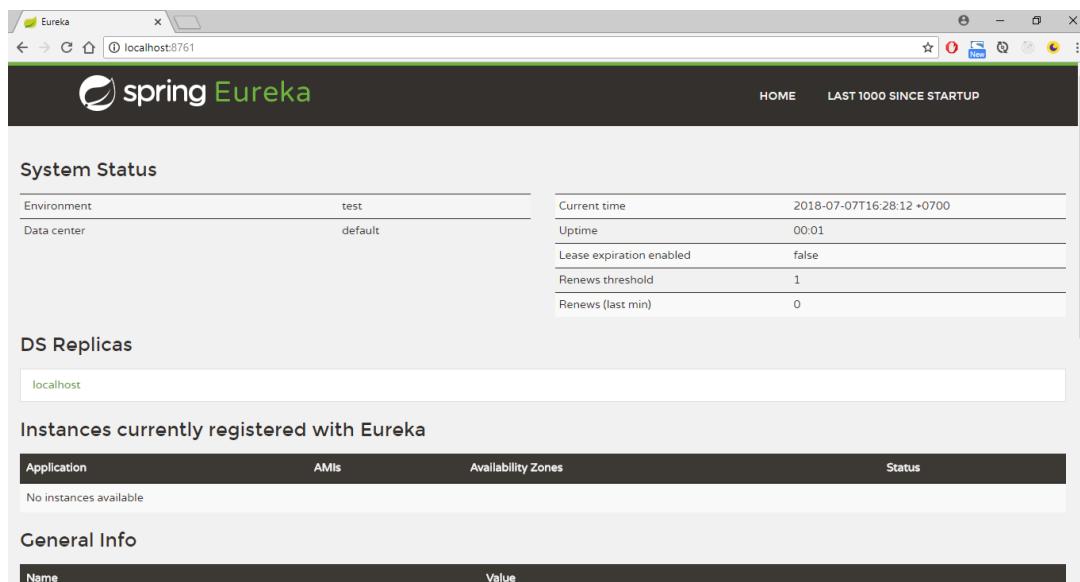
```
package com.business.eureka;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class DemoApplication {

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

- ทดสอบการทำงานของ Eureka Server โดยการรัน 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

```
</dependency>
<dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

- เพิ่ม @EnableEurekaClient ที่ File AuthApplication.java ดังนี้

```
@SpringBootApplication
@EnableEurekaClient
public class AuthApplication {

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

- เปิด project micro\_service\_config และแก้ไข 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>

## MICROSERVICE WITH SPRING BOOT

The screenshot shows the Spring Eureka dashboard running at [localhost:8761](http://localhost:8761). The top navigation bar includes links for HOME and LAST 1000 SINCE STARTUP. The main content area is divided into sections: System Status, DS Replicas, and General Info.

**System Status:**

Environment	test	Current time	2018-07-07T20:22:12 +0700
Data center	default	Uptime	00:02
		Lease expiration enabled	false
		Renews threshold	3
		Renews (last min)	0

**DS Replicas:**

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH-SERVICE	n/a (1)	(1)	UP (1) - localhost:auth-service:8081

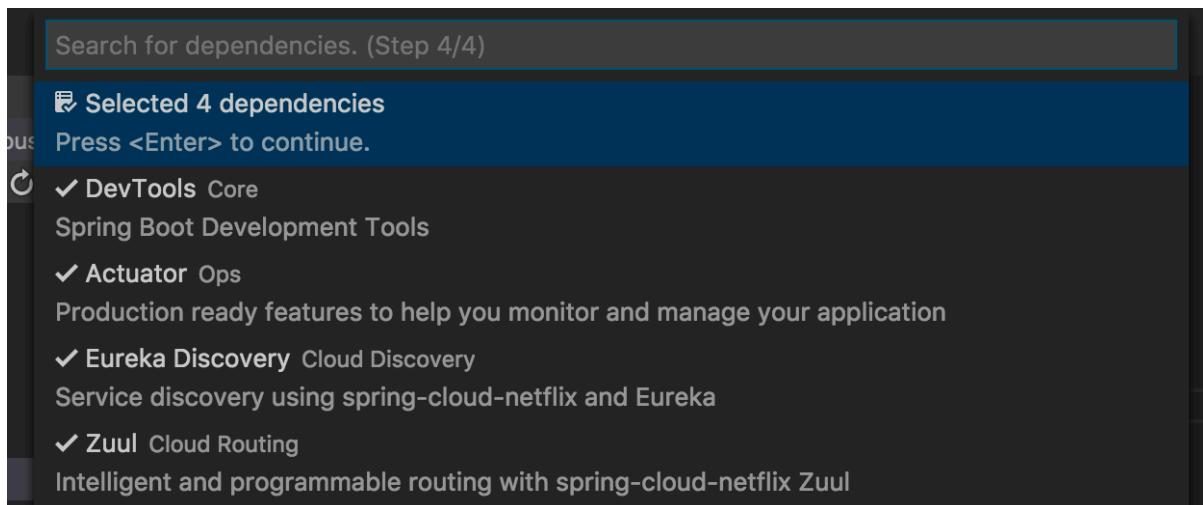
**General Info:**

Name	Value
total-avail-memory	402mb

## ZUUL PROXY

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

- สร้าง project ใหม่ชื่อ zuul\_server โดยมีคุณสมบัติดังนี้
  - group id = com.business
  - artifact 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 เพื่อเปิดใช้งาน ZuulProxy โดยเพิ่มไปใน code ดังนี้

## MICROSERVICE WITH SPRING BOOT

```
package com.business.zuul;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication
@EnableZuulProxy
@EnableEurekaClient
public class DemoApplication {

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

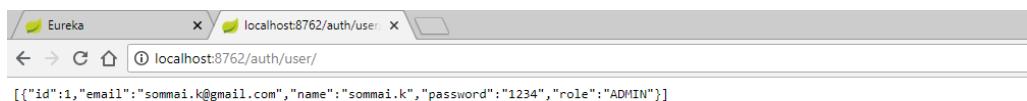
- start config server ด้วยคำสั่งดังนี้
  - mvn spring-boot:run
- กดลองว่า service ได้ตั้งค่าถูกต้องโดยไปตรวจสอบว่ามีรายการ service อยู่บน Eureka Server หรือไม่ โดยเข้า url ดังนี้
  - <http://localhost:8761>

DS Replicas			
Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
AUTH-SERVICE	n/a (1)	(1)	UP (1) - localhost:auth-service:8081
ZUUL-SERVER	n/a (1)	(1)	UP (1) - localhost:zuul-server:8762

General Info	
Name	Value
total-avail-memory	523mb
environment	test
num-of-cpus	4
current-memory-usage	113mb (21%)
server-upptime	00:26

- กดสอบการทำงานของ zuul server ด้วยการกดลองเปิด 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 เพื่อเวลา source code security มาใช้งาน โดยการรันคำสั่งดังนี้
  - git clone <https://github.com/pnpsolution/springboot-workshop.git>
  - จะได้ folder ดังนี้

Name	Date modified	Type	Size
.git	7/8/2018 11:44 AM	File folder	
sourcecode	7/8/2018 11:44 AM	File folder	
README.md	7/8/2018 11:44 AM	Markdown Source...	1 KB

**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 config file 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
```

## MICROSERVICE WITH SPRING BOOT

- สร้าง folder ชื่อ security ภายใน folder zuul และนำ code จาก path /sourcecode/security/zuul\_server มาใส่ไว้ดังภาพ

```
src
└── main
 └── java
 └── com
 └── business
 └── zuul
 └── security
 ├── JwtTokenAuthenticationFilter.java
 ├── SecurityTokenConfig.java
 └── DemoApplication.java
```

### 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

```
package com.business.auth.user;

import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User, Long> {

 User findByEmail(String email);
}
```

## MICROSERVICE WITH SPRING BOOT

- สร้าง folder ชื่อ security ภายใน folder auth และนำ code จาก path /sourcecode/security/auth\_server มาใส่ไว้ดังภาพ

```
src
└── main
 └── java
 └── com
 └── business
 └── auth
 └── security
 ├── JwtUsernameAndPasswordAuthenticationFilter.java
 ├── SecurityCredentialsConfig.java
 └── UserDetailsServiceImpl.java
```

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

POST [localhost:8762/auth](#)

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary **JSON (application/json)**

```
1 {
2 "username": "sommai.k@hotmail.com",
3 "password": "12345"
4 }
```

Body Cookies Headers (9) Test Results

**authorization → Bearer**  
eyJhbGciOiJIUzUxMiJ9.eyJzdWlIiOiJzb21tYWkua0Bob3RtYWlsLmNvbSIsImF1dGhvcmI0aWVzIjpblIJPTI  
0jWEgEvsrfP9uNgvx7Y6XKDoGK9dm-yQzPAuz9PivTOldgQ

## MICROSERVICE WITH SPRING BOOT

- เมื่อสั่ง run จะได้ token มาดังภาพ

Body Cookies Headers (9) Test Results Status: 200 OK Time: 932 ms

authorization → Bearer eyJhbGciOiJIUzIwMjQ.eyJzdWliOiJzb21tYWkua0BnbWFpbC5jb20iLCJhdXRob3JpdGllcyI6WyJST0xFx0FETUIOii0slmhdCl6MTUzMtaMDlyMywiZXhwIjoxNTMxMTE2NjIzfQ.pVrMBLN6ksU2\_8NJ0NHa17hsUzvDbbYf9MK\_jMHh8zAU2dSy-YBZmw

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

http://localhost:8762/ + ...

GET http://localhost:8762/admin/user

Authorization Headers Body Pre-request Script Tests

- เมื่อสั่ง run จะได้ ผลลัพธ์เป็น 401 unauthorized มาดังภาพ

```
{ "timestamp": "2018-07-08T06:11:49.677+0000", "status": 401, "error": "Unauthorized", "message": "No message available", "path": "/admin/user" }
```

- ให้นำ token ที่ได้ส่งไปที่ header ดังภาพ

GET http://localhost:8762/admin/user Params

Authorization Headers (1) Body Pre-request Script Tests

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzIwMjQ.eyJzdWliOiJzb21tYWkua0B... New key Value Description	

- เมื่อสั่ง run จะได้ ผลลัพธ์เป็นข้อมูลมาดังภาพ

```
1 [2 { 3 "id": 1, 4 "email": "sommai.k@gmail.com", 5 "name": "sommai.k", 6 "password": "1234", 7 "role": "ADMIN" 8 } 9]
```

## 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
```

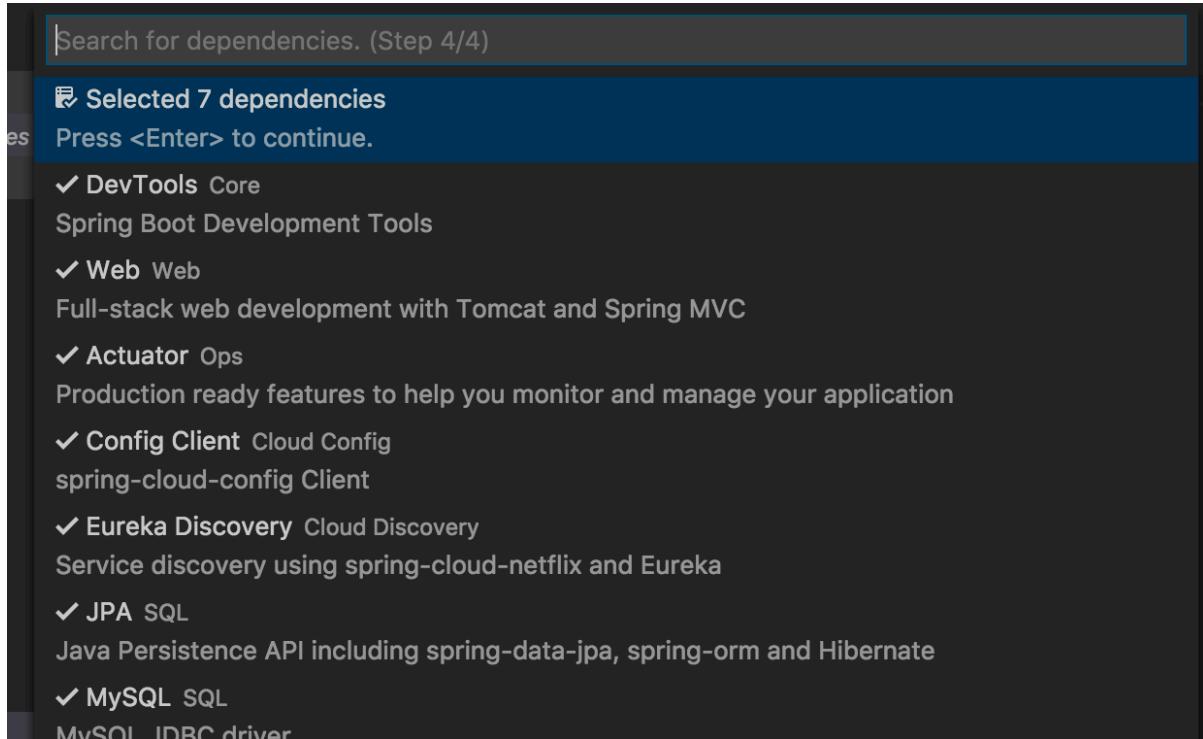
```
info.app.name=Order Service Application
info.app.description=Order Service on spring boot application
info.app.version=1.0.0
server.port=8082
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

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

- group id = com.business
- artifact id = order
- dependencies ดังนี้
  - web
  - actuator
  - jpa (sql)
  - mysql หรือ h2

## MICROSERVICE WITH SPRING BOOT

- devtools
- config client
- eureka discovery



- สร้าง file bootstrap.properties ดังนี้

```
spring.application.name=order-service
spring.cloud.config.uri=http://localhost:8888
```

- เพิ่ม @EnableEurekaClient ลงภาพ

```

package com.business.order;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
@EnableEurekaClient
public class DemoApplication {

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

```

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

DS Replicas			
localhost			
Application	AMIs	Availability Zones	Status
AUTH-SERVICE	n/a (1)	(1)	UP (1) - 192.168.33.1:auth-service:8081
ORDER-SERVICE	n/a (1)	(1)	UP (1) - 192.168.33.1:order-service:8082
ZUUL-SERVER	n/a (1)	(1)	UP (1) - 192.168.33.1:zuul-server:8762

### 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

```
package com.business.order.item;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Product {

 @Id
 @GeneratedValue(strategy=GenerationType.AUTO)
 private Long id;

 @Column(unique=true)
 private String sku;
 private String name;
 private String uom;
 private Double price;

 public Product(){

 }

 public Product(String sku, String name, String uom, Double price){
 this.sku = sku;
 this.name = name;
 this.uom = uom;
 this.price = price;
 }
}
```

\*\* หมายเหตุ\*\* ต้องสร้าง method set, get ให้ครบถ้วนกับ field ทั้งหมด

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

##### วิธีการ

- สร้าง interface ชื่อ ProductRepository ภายใน 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 ภายใน package com.business.order.item

## MICROSERVICE WITH SPRING BOOT

```
package com.business.order.item;

import java.util.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

@Entity
public class SaleOrder {
 @Id
 @GeneratedValue(strategy=GenerationType.AUTO)
 private Long id;
 private Date date;
 private String status;
 private String promotion;
 private Double subnet;
 private Double discount;
 private Double net;

 @OneToMany(mappedBy="saleOrder", cascade=CascadeType.ALL)
 private Set<Item> items;

 public SaleOrder(){
 }

 public SaleOrder(
 Date date, String status, String promotion,
 Double subnet, Double discount, Double net,
 Set<Item> items){
 this.date = date;
 this.status = status;
 this.promotion = promotion;
 this.subnet = subnet;
 this.discount = discount;
 this.net = net;
 this.items = items;
 }
}
```

\*\* ក្នុងកិច្ចការណ៍ ត้องសរាង method set, get ឱ្យកសបុក្ខ field ដោយបន្ថែមក្នុង

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

**วิธีการ**

- สร้าง interface ชื่อ SaleOrderRepository ภายใน 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 ภายใน package com.business.order.item

```
package com.business.order.item;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
public class Item {
 @Id
 @GeneratedValue(strategy=GenerationType.AUTO)
 private Long id;

 @ManyToOne
 @JoinColumn(name="saleOrderId")
 private SaleOrder saleOrder;

 @ManyToOne
 @JoinColumn(name="productId")
 private Product product;

 private Integer qty;

 public Item(){}
 public Item(SaleOrder saleOrder, Product product, Integer qty){
 this.saleOrder = saleOrder;
 this.product = product;
 this.qty = qty;
 }
}
```

\*\* ហ្មាយឱកទុ \*\* ត้องสร้าง method set, get ให้ครบถ้วน field ដ้วยนะครับ

### WORKSHOP#8 สร้าง Repository ของ Entity Item

#### วิธีการ

- สร้าง interface ชื่อ ItemRepository ภายใน 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 Bean เพื่อกำกับ initial data

#### วิธีการ

- เพิ่ม code Bean เข้าไปใน DemoApplication ดังนี้ เพื่อ initial data

```
@SpringBootApplication
@EnableEurekaClient
public class DemoApplication {

 Run | Debug
 public static void main(String[] args) {
 SpringApplication.run(DemoApplication.class, args);
 }

 @Bean
 public CommandLineRunner initialProductData(ProductRepository productRepo, SaleOrderRepository saleRepo,
 ItemRepository itemRepo) {
 return (args) -> {
 List<Product> products = new ArrayList<>();
 products.add(new Product("10001", "Java Version 1.0", "pct", 100d));
 products.add(new Product("10002", "Devops", "pct", 230d));
 products.add(new Product("10003", "Spring Boot ver. 2.1", "pct", 200d));
 products.add(new Product("10004", "Flutter", "pct", 350d));
 productRepo.saveAll(products);
 };
 }
}
```

### WORKSHOP#10 สร้าง REST API ชื่อ SaleOrderController เก็บอยู่ใน package com.business.order.item

#### วิธีการ

- เพิ่ม method เพื่อสร้าง Order โดยมีชื่อดังนี้ createOrder มี code ดังนี้  
    @GetMapping("/newOrder")  
    public SaleOrder createOrder() {

## MICROSERVICE WITH SPRING BOOT

```
SaleOrder saleOrder = new SaleOrder(new Date(), "Prepare", null, 0d, 0d, 0d, null);
saleRepo.save(saleOrder);
return saleOrder;
}
```

- เพิ่ม method เพื่อสร้าง item ให้กับ order โดยมีชื่อดังนี้ addItem มี code ดังนี้

```
@PostMapping("/addItem/{orderId}")
public Map<String, Object> addItem(@PathVariable("orderId") long id, @RequestBody
Map<String, Object> body) {
 Map<String, Object> res = new HashMap<>();
 Optional<SaleOrder> sale = saleRepo.findById(id);
 if (sale.isPresent()) {
 SaleOrder saleOrder = sale.get();

 List<Product> product = productRepo.findBySku(body.get("sku").toString());

 Double total = product.get(0).getPrice() * (Integer) body.get("qty");
 saleOrder.setNet(total);
 saleOrder.setSubnet(total);

 Item item = new Item(saleOrder, product.get(0), (Integer) body.get("qty"));
 saleOrder.getItems().add(item);

 saleRepo.save(saleOrder);
 res.put("success", true);
 res.put("data", saleOrder.getId());
 } else {
 res.put("success", false);
 res.put("data", "no data found");
 }
 return res;
}
```

- เพิ่ม method เพื่อดูรายละเอียด order โดยมีชื่อดังนี้ orderDetail มี ขั้นตอนดังนี้

- สร้างไฟล์ View.java โดยมี code ดังนี้

```
public class View {
 interface Summary {}
```

## MICROSERVICE WITH SPRING BOOT

- ให้ใส่ @JsonView(View.Summary.class) ที่ column ที่ต้องการแสดงผล เช่น

```
@Entity
public class SaleOrder {

 @Id
 @GeneratedValue(strategy = GenerationType.AUTO)
 @JsonView(View.Summary.class)
 private Long id;

 @JsonView(View.Summary.class)
 private Date date;

 @JsonView(View.Summary.class)
 private String status;

 @JsonView(View.Summary.class)
 private String promotion;

 @JsonView(View.Summary.class)
 private Double subnet;

 @JsonView(View.Summary.class)
 private Double discount;

 @JsonView(View.Summary.class)
 private Double net;

 @OneToMany(mappedBy = "saleOrder", cascade = CascadeType.ALL, orphanRemoval = true)
 @JsonView(View.Summary.class)
 private Set<Item> items;
```

- เพิ่ม method เพื่อจูราຍละเวียด order โดยมีชื่อดังนี้ orderDetail มี code ดังนี้

```
@JsonView(View.Summary.class)
@GetMapping("/orderDetail/{orderId}")
public SaleOrder orderDetail(@PathVariable("orderId") long id) {
 SaleOrder saleOrder = null;
```

```
Optional<SaleOrder> sale = saleRepo.findById(id);
if (sale.isPresent()) {
 saleOrder = sale.get();
}
return saleOrder;
}
```

## MICROSERVICE WITH SPRING BOOT

- เพิ่ม method เพื่อแก้ไข item ใน order โดยมีชื่อดังนี้ updateItem มี code ดังนี้  
@PutMapping("/updateItem/{itemId}")

```
public Map<String, Object> updateItem(@PathVariable("itemId") long id,
@RequestBody Map<String, Object> body) {
 Map<String, Object> res = new HashMap<>();
 Optional<Item> itemOpt = itemRepo.findById(id);

 if (itemOpt.isPresent()) {
 Item item = itemOpt.get();

 List<Product> product = productRepo.findBySku(body.get("sku").toString());
 Integer qty = (Integer) body.get("qty");

 Double total = product.get(0).getPrice() * qty;

 item.setProduct(product.get(0));
 item.setQty(qty);
 item.getSaleOrder().setNet(total);
 item.getSaleOrder().setSubnet(total);

 itemRepo.save(item);
 res.put("success", true);
 res.put("data", item.getId());
 } else {
 res.put("success", false);
 res.put("data", "no data found");
 }
 return res;
}
```

- เพิ่ม method เพื่อลบ item โดยมีชื่อดังนี้ removeItem มี code ดังนี้

```
@DeleteMapping("/removeItem/{itemId}")
public Map<String, Object> removeItem(@PathVariable("itemId") long id) {
 Map<String, Object> res = new HashMap<>();
 if (itemRepo.existsById(id)) {
 itemRepo.deleteById(id);
 res.put("success", true);
 res.put("data", id);
 } else {
```

## MICROSERVICE WITH SPRING BOOT

```
 res.put("success", false);
 res.put("message", "no data found");
 }
 return res;
}
```

- เพิ่ม method ชื่อ "/order/setPromotion/{id}/{promotionCode}" โดย ภายใน method จะมีการเรียกไป service promotion-service เพื่อดึงค่า promotion มาใช้ใน controller โดยมี code ดังนี้

```
package com.business.order.item;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
@RequestMapping("/order")
public class SaleOrderController {

 @Autowired
 RestTemplate restTemplate;

 @GetMapping("/setPromotion/{id}/{promotionCode}")
 public Map<String, String> findOrderByPromotion(
 @PathVariable Long id,
 @PathVariable String promotionCode
){
 @SuppressWarnings("unchecked")
 Map <String, String> promotion = restTemplate.getForObject(
 "http://promotion-service/promotion/" + promotionCode, Map.class);
 return promotion;
 }
}
```

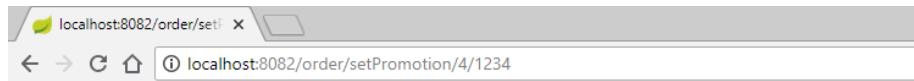
- เพิ่ม Bean RestTemplate เพื่อให้สามารถเรียกใช้ method ของ service อื่นๆ ภายในระบบ โดยต้องไปเพิ่มใน DemoApplication ดังนี้

```
@SpringBootApplication
@EnableEurekaClient
public class DemoApplication {

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

 @Bean
 @LoadBalanced
 public RestTemplate restTemplate(RestTemplateBuilder builder) {
 return builder.build();
 }
}
```

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



### Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Mon Jul 09 09:29:44 ICT 2018  
There was an unexpected error (type=Internal Server Error, status=500).  
No instances available for promotion-service

## HYSTRIX

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

- 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

```
@SpringBootApplication
@EnableEurekaClient
@EnableCircuitBreaker
public class DemoApplication {

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

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

```
@HystrixCommand(fallbackMethod = "getPromotionFallback")
@GetMapping("/setPromotion/{id}/{promotionCode}")
public Map<String, String> findOrderById(
 @PathVariable Long id,
 @PathVariable String promotionCode
){
 @SuppressWarnings("unchecked")
 Map<String, String> promotion = restTemplate.getForObject(
 "http://promotion-service/promotion/" + promotionCode, Map.class);
 return promotion;
}

public Map<String, String> getPromotionFallback(Long id, String promotionCode, Throwable hystrixCommand) {
 Map<String, String> result = new HashMap<>();
 System.out.println(hystrixCommand.getMessage());
 result.put("success", "false");
 result.put("message", hystrixCommand.getMessage());
 return result;
}
```

- start service ด้วยคำสั่งดังนี้
  - mvn spring-boot:run

## MICROSERVICE WITH SPRING BOOT

- ทดสอบการทำงานด้วยการเข้าไปเรียกใช้ผ่าน url ดังนี้
  - <http://localhost:8082/order/setPromotion/4/1234>
  - จะได้ผลลัพธ์ดังนี้



```
{"success": "false", "message": "No instances available for promotion-service"}
```

## 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

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@RestController
@RequestMapping("/order")
public class SaleOrderController {

 private static final Logger LOGGER = LoggerFactory.getLogger(SaleOrderController.class);

 @Autowired
 RestTemplate restTemplate;

 @HystrixCommand(fallbackMethod = "getPromotionFallback")
 @GetMapping("/{id}/{promotionCode}")
 public Map<String, String> findOrderByid(
 @PathVariable Long id,
 @PathVariable String promotionCode
){
 LOGGER.info("call to promotion service");
 @SuppressWarnings("unchecked")
 Map<String, String> promotion = restTemplate.getForObject(
 "http://promotion-service/promotion/" + promotionCode, Map.class);
 LOGGER.info("response from promotion");
 return promotion;
 }
}
```

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

```
INFO [order-service,0531ee3fa36a490f,f446aa38d42d2f02,false] 41579 --- [derController-1] c.b.order.item.SaleOrderController : call to promotion service
INFO [order-service,0531ee3fa36a490f,e444cd00fe1cb0,false] 41579 --- [derController-1] s.c.a.AnnotationConfigApplicationContext : Refreshing SpringClientFactor
g.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@5c3219f
```

## Exercise

**exercise#1** ให้ตั้งค่าที่ zuul server ให้สามารถเรียกใช้งาน order\_service ผ่าน url ดังนี้ได้  
<http://localhost:8762/order/<url>> 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
- artifact 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

## MICROSERVICE WITH SPRING BOOT

- ให้หาค่า discount จาก code ที่ได้รับ
- ส่งค่ากลับเป็น json ในกรณีハウเจอ
  - { valid : true, discount : 100 }
- ส่งค่ากลับเป็น json ในกรณีハウไม่เจอ
  - { 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 }

**exercise#4** start promotion-service 2 instance

- build project promotion\_service ด้วยคำสั่งดังนี้
  - mvn package
- สร้าง file promotion-service-t1.properties เก็บไว้บน config server โดยคัดลอกมาจาก promotion-service.properties และเปลี่ยน port เป็น 8084
- ให้ stop promotion-service
- start promotion-service ตัวแรกด้วยคำสั่งดังนี้
  - เข้าไปที่ folder ที่เก็บ project / target จะมี file XXX.jar อยู่
  - run ด้วยคำสั่ง java -jar xxx.jar
- start promotion-service ตัวสองด้วยคำสั่งดังนี้
  - java -jar xxx.jar --spring.profiles.active=t1
- ทดสอบด้วยการเปิดดู Eureka จะแสดงผลลัพธ์ว่ามี 2 link เกิดขึ้นมา
- ทดสอบด้วยการเรียกใช้ api ผ่าน zuul แล้วเปิด ดู log ของแต่ละ instance ดูผลลัพธ์