

# Workshop

MICROSERVICE  
WITH  
SPRING BOOT

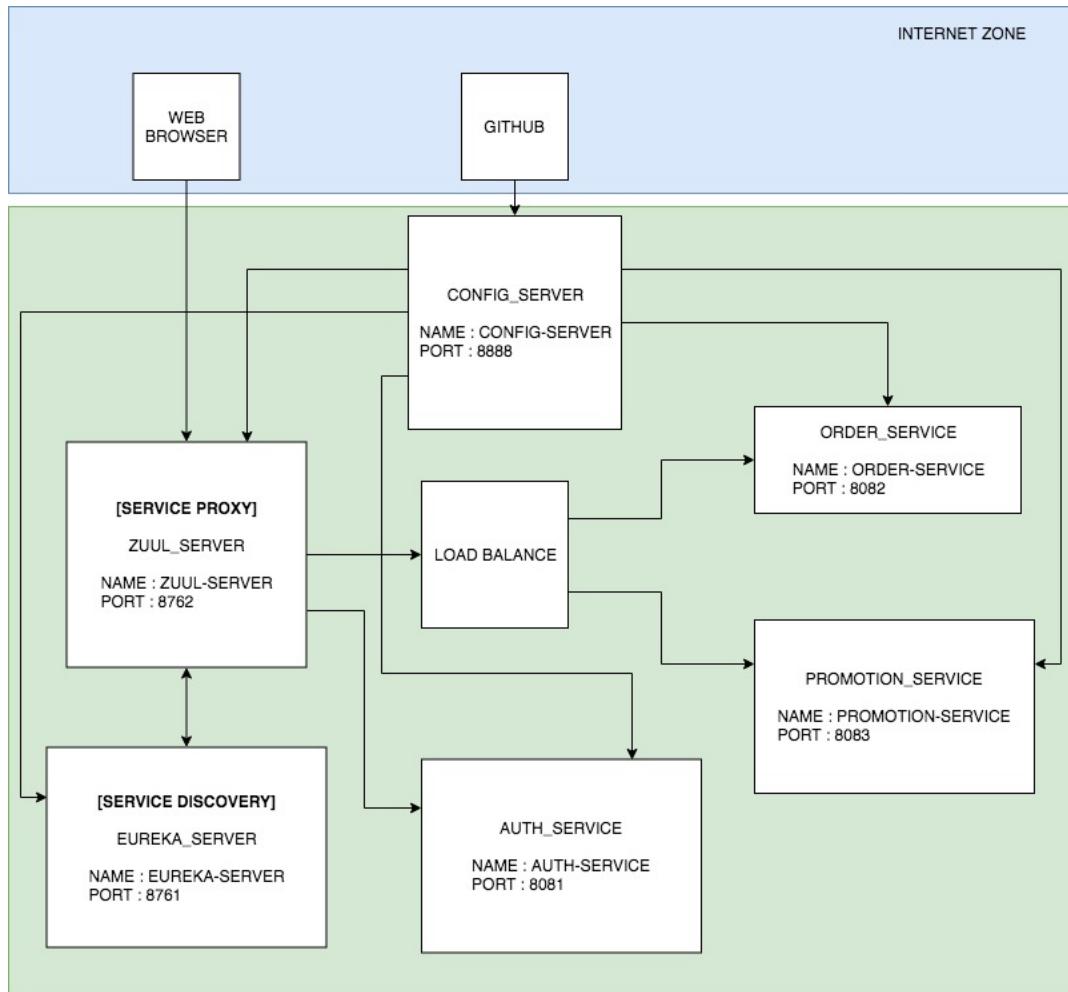
by  
PnP Solution

[www.pnpsw.com](http://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 เสร็จแล้วให้กดสอบบนการติดตั้งดังนี้
  - 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)
```

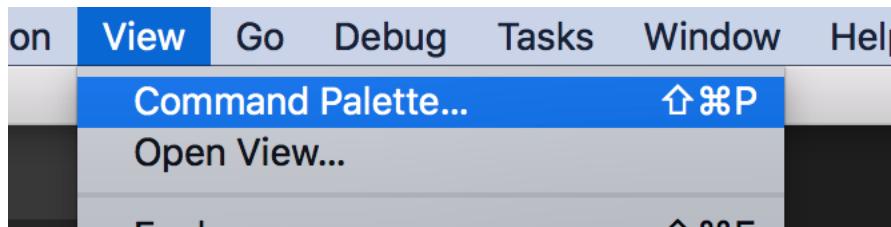
## AUTH SERVICE

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

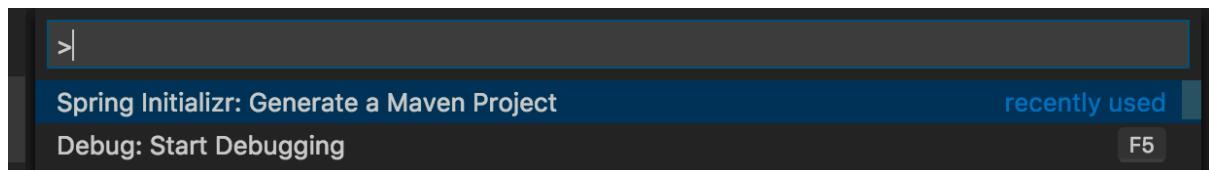
- group id = com.business
- artifact id = auth
- dependencies ดังนี้
  - web
  - actuator
  - jpa (sql)
  - mysql หรือ h2
  - devtools

### วิธีการ

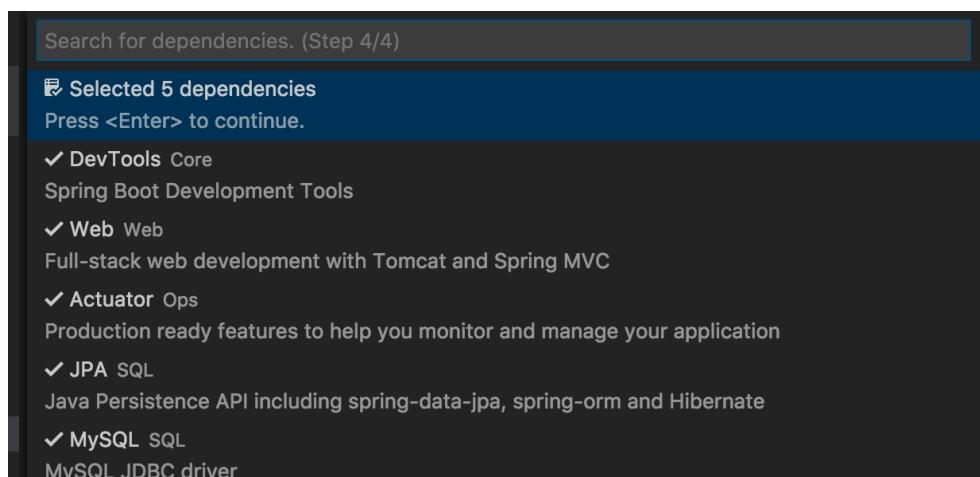
- เปิด Program Visual Studio Code แล้ว ไปที่ เมนู View > Command Palette... ดังภาพ



- 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 ภายใน 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##** 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##** 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

## MICROSERVICE WITH SPRING BOOT

```
<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##** สร้าง Bean เพื่อกำกัง 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

```
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##** 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>  
}
```

## MICROSERVICE WITH SPRING BOOT

### វិធានា

- សរាង Class ថែរ UserController ក៏បញ្ជីនៅ package com.business.auth.user ដังនេះ

```
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;

}
```

- ធំនុយកើតឡើងថាអ្នកប្រើប្រាស់ការណែនាំរបស់ខ្លួន ដូចជា code ដោយប្រើប្រាស់ការណែនាំរបស់ខ្លួន

```
@GetMapping("")
public Iterable<User> getUser() {
    Iterable<User> list = userRepository.findAll();
    return list;
}
```

- ធំនុយកើតឡើងថាអ្នកប្រើប្រាស់ការណែនាំរបស់ខ្លួន ដូចជា code ដោយប្រើប្រាស់ការណែនាំរបស់ខ្លួន

```
@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

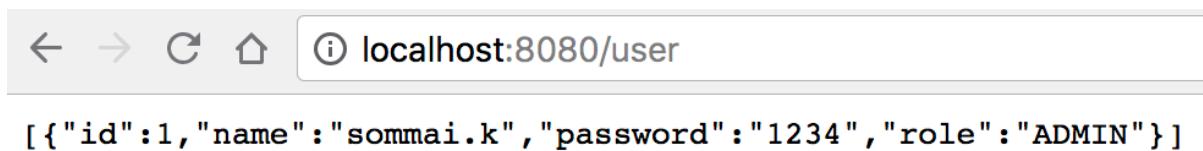
```
@PutMapping("/{id}")
public Map<String, Object> updateUser(
    @RequestBody Map<String, String> body,
    @PathVariable Long id
) {
    Map<String, Object> res = new HashMap<>();
    if(userRepository.existsById(id)){
        User other = new User(
            body.get("email"),
            body.get("user"),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

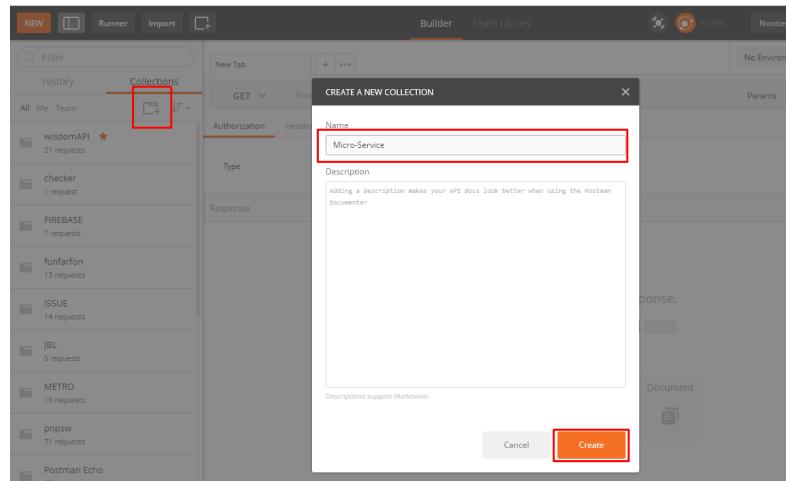
```
@DeleteMapping("/{id}")
public Map<String, Object> deleteUser(
    @PathVariable Long id
) {
    Map<String, Object> res = new HashMap<>();
    if(userRepository.existsById(id)){
        userRepository.deleteById(id);
        res.put("success", true);
        res.put("data", id);
    }else{
    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

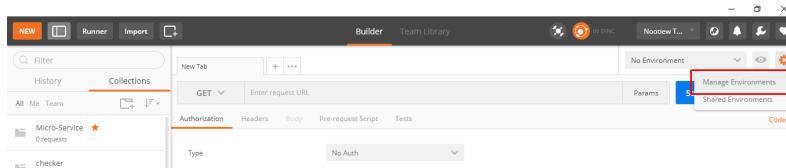


## MICROSERVICE WITH SPRING BOOT

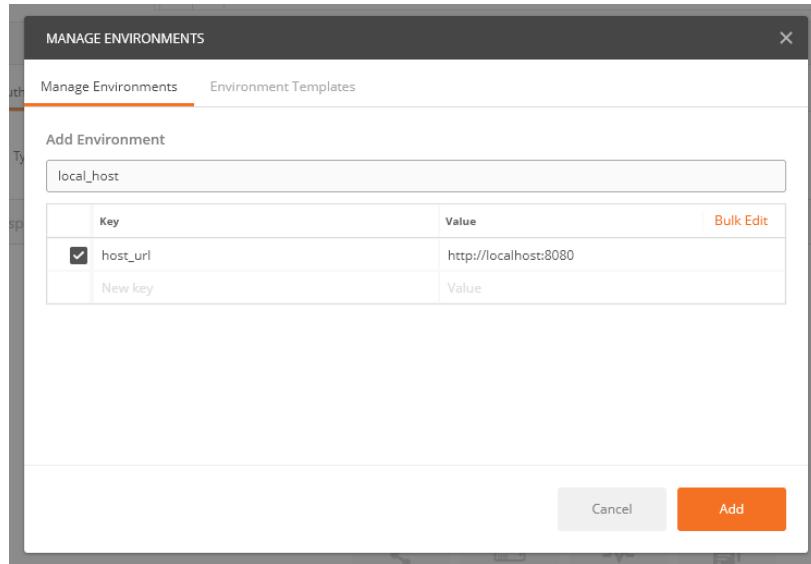
- ทดลองทดสอบ api ทั้งหมด ผ่าน postman โดยมีการตั้งค่าดังนี้
  - เข้าหน้าจอ postman สร้าง collection ชื่อ MicroService



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

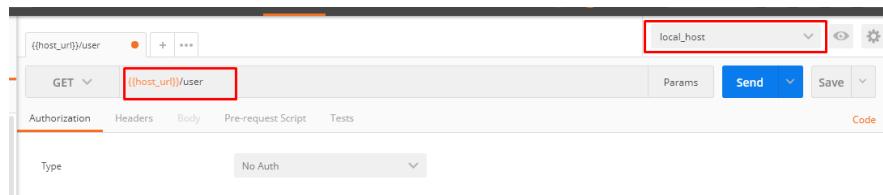


## MICROSERVICE WITH SPRING BOOT

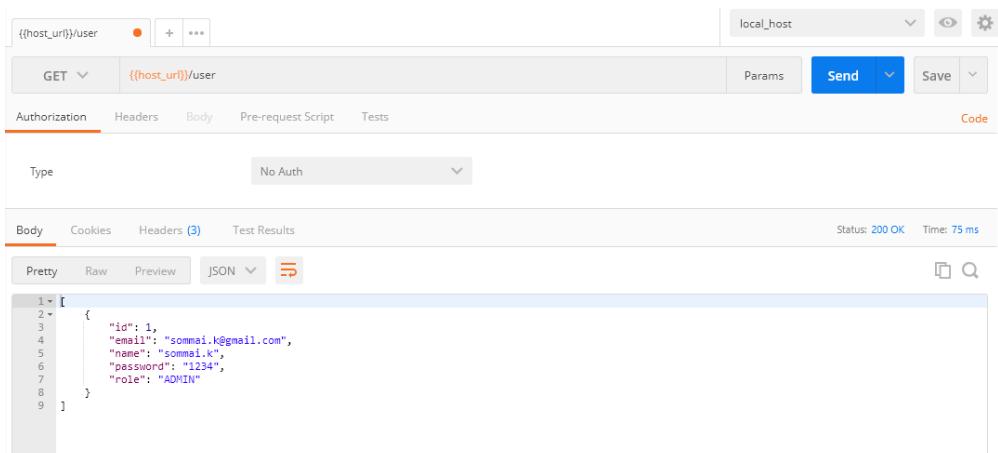


- สร้าง tab เพื่อทดสอบ method get โดยใส่ค่า url ดังนี้

- `{{host_url}}/user`

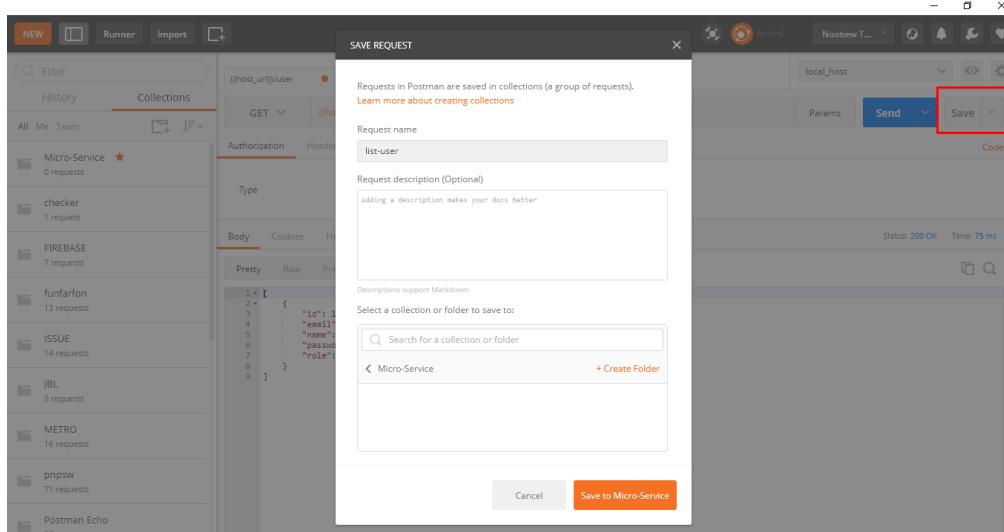


- กดรันจะได้ผลลัพธ์ดังนี้



## MICROSERVICE WITH SPRING BOOT

- บันทึก link สำหรับการเทสไว้ใน collection โดยการกด Save แล้วเลือก collection ที่ต้องการ

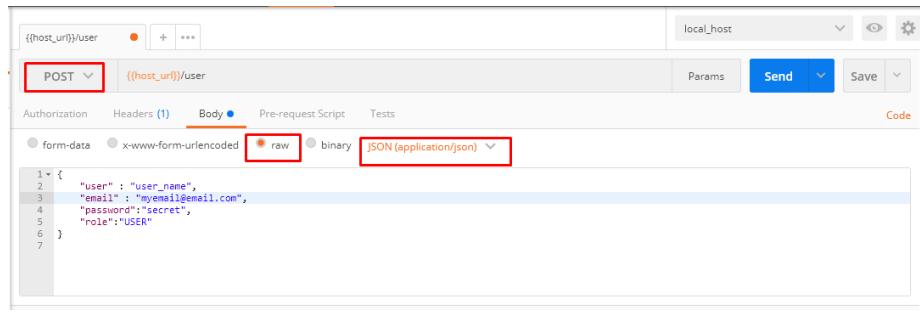


- สร้าง tab เพื่อทดสอบ method post โดยใส่ค่า url ดังนี้

- {{host\_url}}/user
- body มีค่าดังนี้

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

## MICROSERVICE WITH SPRING BOOT



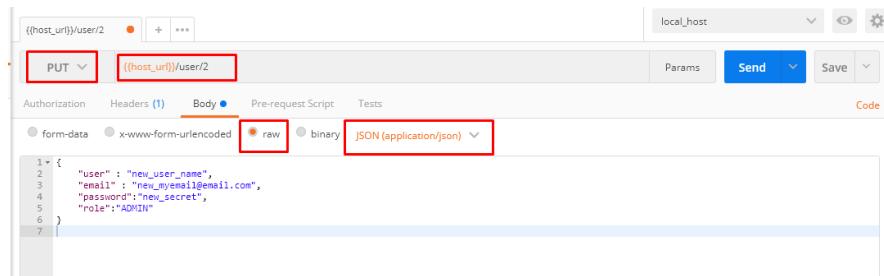
### ▪ กดรับจะได้ผลลัพธ์ดังนี้

The screenshot shows the Postman interface with the response tab selected. The status is 200 OK and the time is 367 ms. The response body is:

```
1 * {
2     "id": 2,
3     "email": "myemail@email.com",
4     "name": "user_name",
5     "password": "secret",
6     "role": "USER"
7 }
```

### ○ 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"
}
```



### ▪ กดรับจะได้ผลลัพธ์ดังนี้

The screenshot shows the Postman interface with the response tab selected. The status is 200 OK and the time 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 }
```

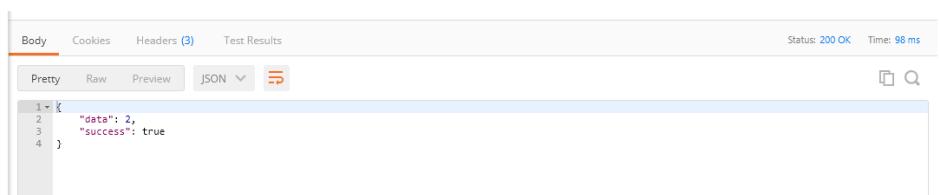
## MICROSERVICE WITH SPRING BOOT

- สร้าง tab เพื่อทดสอบ method delete โดยใส่ค่า url ดังนี้

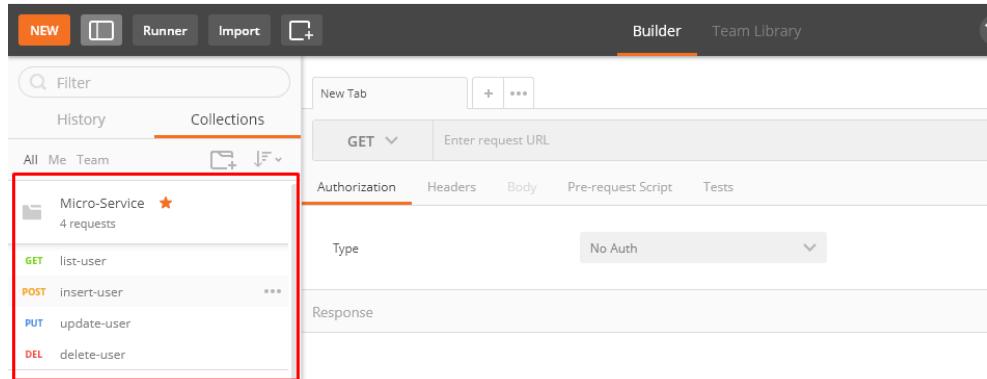
- {{host\_url}}/user/1



- กดรับจะได้ผลลัพธ์ดังนี้



- เมื่อทดสอบครบถ้วน Method และทำการบันทึกไว้ใน collection จะได้ผลลัพธ์ดังนี้



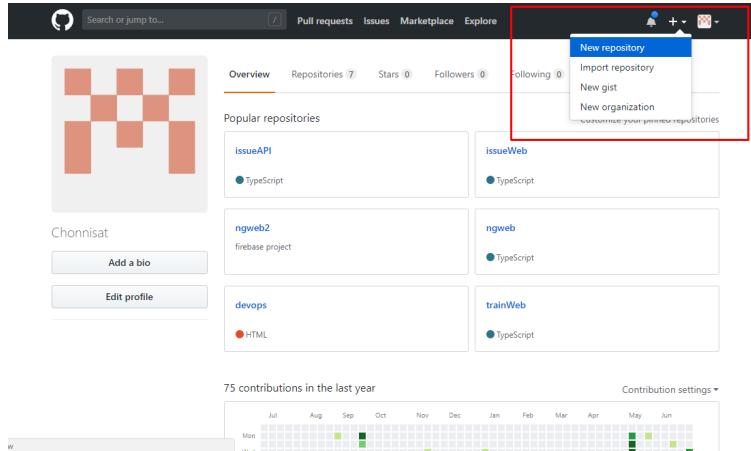
## GIT

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

## วิธีการ

## MICROSERVICE WITH SPRING BOOT

- 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.

Add .gitignore: None ▾ Add a license: None ▾ ?

**Create repository**

## MICROSERVICE WITH SPRING BOOT

- จะได้ผลลัพธ์ดังนี้

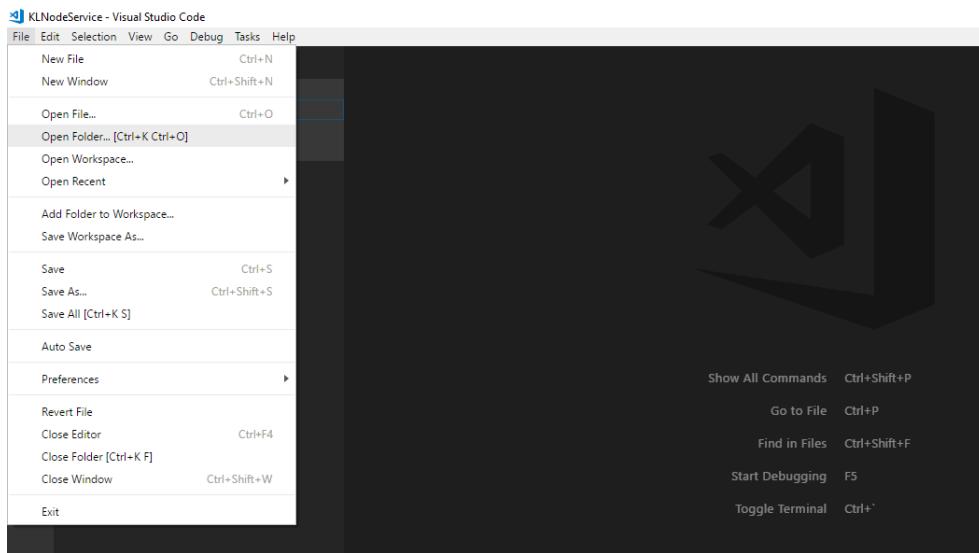
The screenshot shows the GitHub 'Quick setup' interface. It includes sections for 'Quick setup — if you've done this kind of thing before', '...or create a new repository on the command line' (with a terminal command example), '...or push an existing repository from the command line' (with a terminal command example), and '...or import code from another repository' (with a 'Import code' button). A large black circle with the number '1' is overlaid on the top right of the interface.

- 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/<<your_id>/micro_service_config.git)
    - git push -u origin master
- ผลลัพธ์ที่ได้มี่อเราเข้าไปที่หน้า project เราอีกครั้งบน github

The screenshot shows a GitHub repository page for 'Chonnisat / demo'. The repository has 1 commit, 1 branch, 0 releases, and 0 contributors. The latest commit was made a minute ago by 'Chonnisa' with the message 'first commit'. The README.md file contains the text 'hello'.

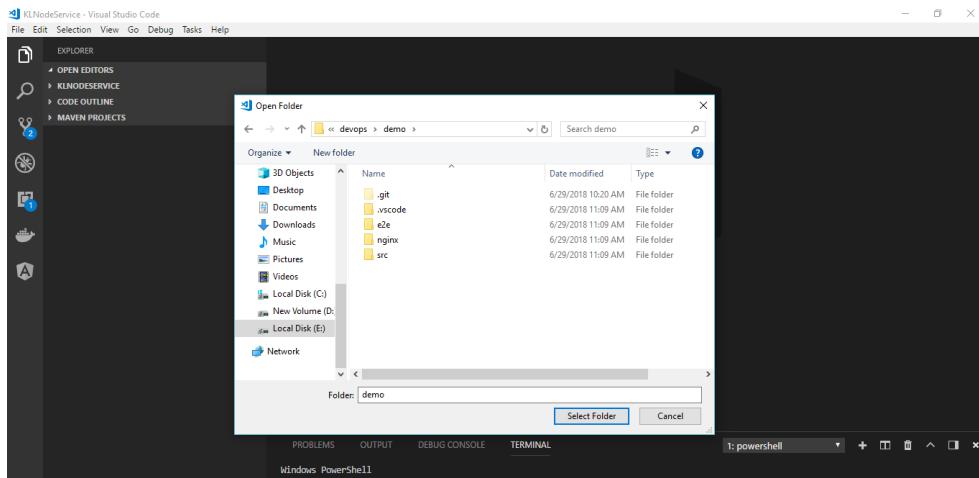
## MICROSERVICE WITH SPRING BOOT

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



## MICROSERVICE WITH SPRING BOOT

- 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
```

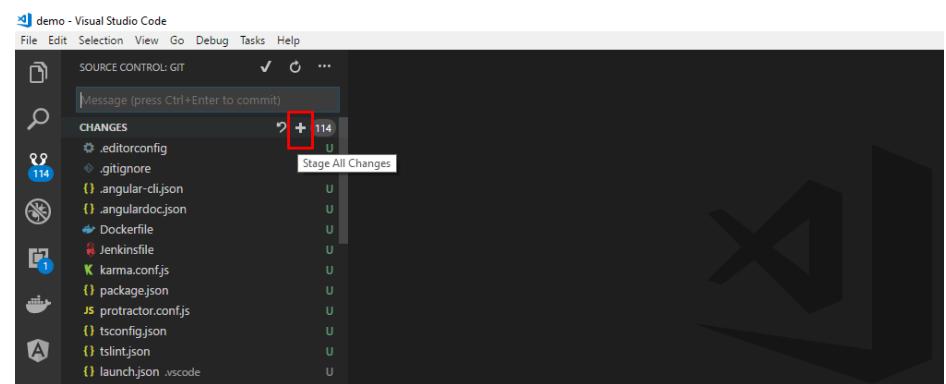
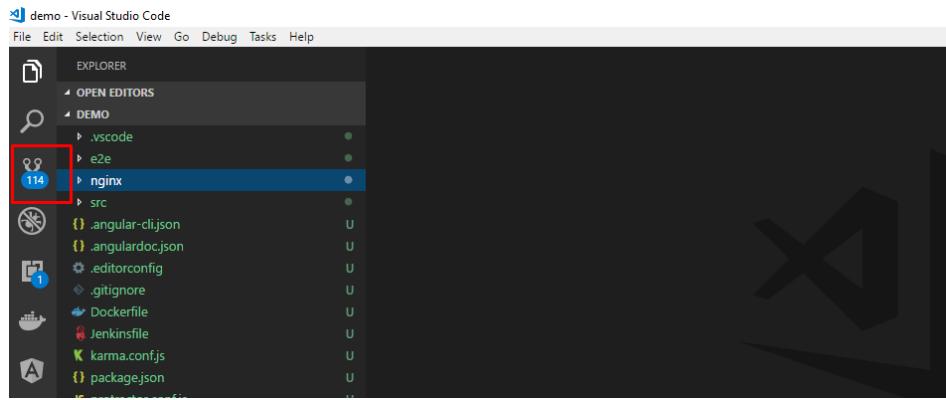
info.app.name=Auth Service

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

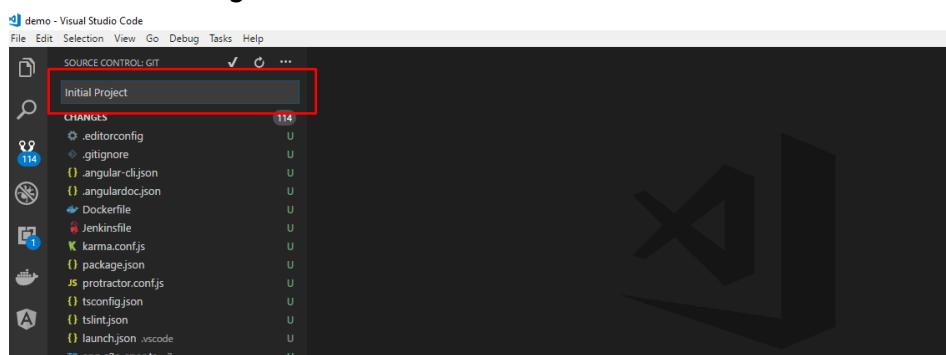
info.app.version=1.0.0

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

## MICROSERVICE WITH SPRING BOOT

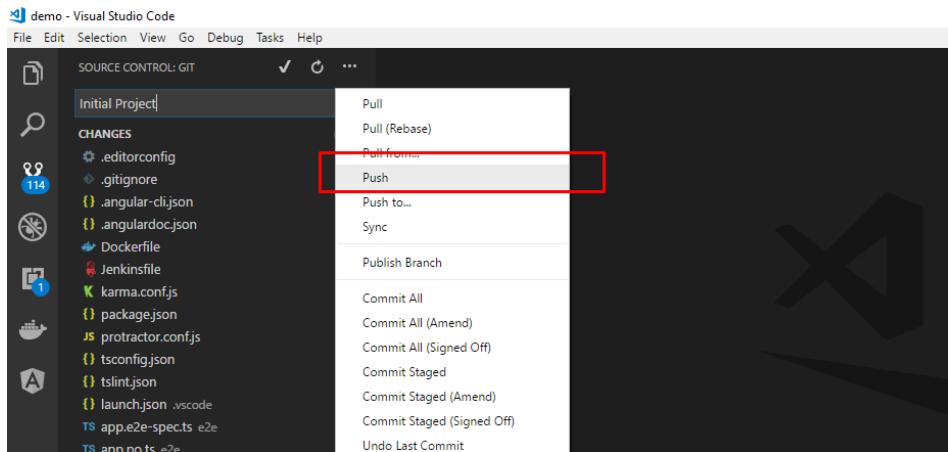


- พิมพ์ message ที่ต้องการจะ commit

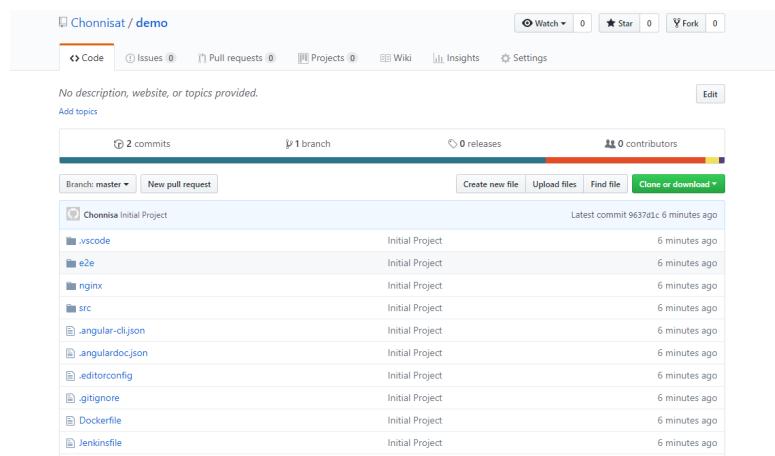


- กดปุ่ม เพื่อกำการ push code ขึ้นไปบน github

# MICROSERVICE WITH SPRING BOOT



- 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

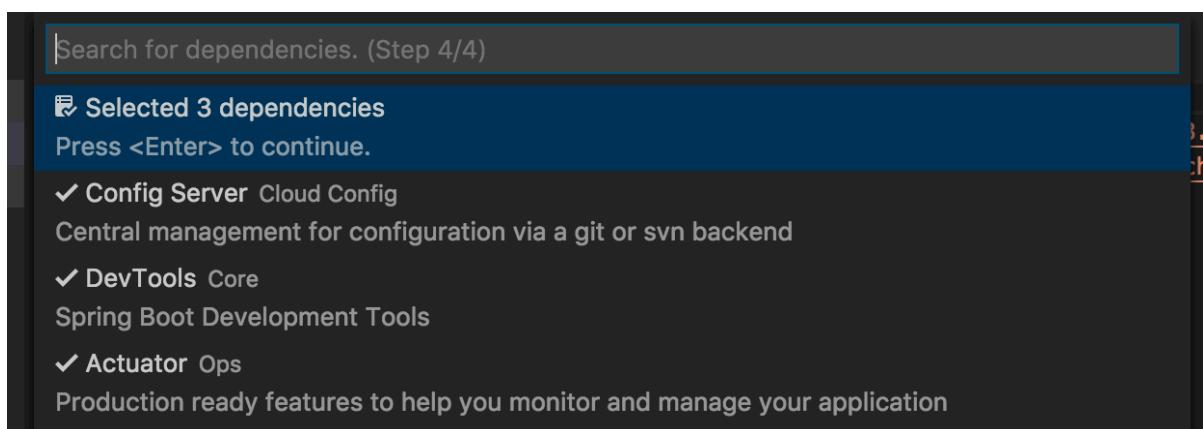
สมมุติว่าชื่อ project เป็น auth-service และรัน profile prod ก็จะไปหา file ชื่อ auth-service-prod.properties

## 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
- กดสอบการทำงานของ config server ด้วยการกดลองเปิด url ดังนี้
  - <http://localhost:8888/auth-service/default>
  - จะได้ผลลัพธ์ดังนี้



\*\*หมายเหตุ\*\*

default หมายถึง ค่า profile ที่ต้องการ

## CONFIG CLIENT

**WORKSHOP#1** เรียนรู้ 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>
```

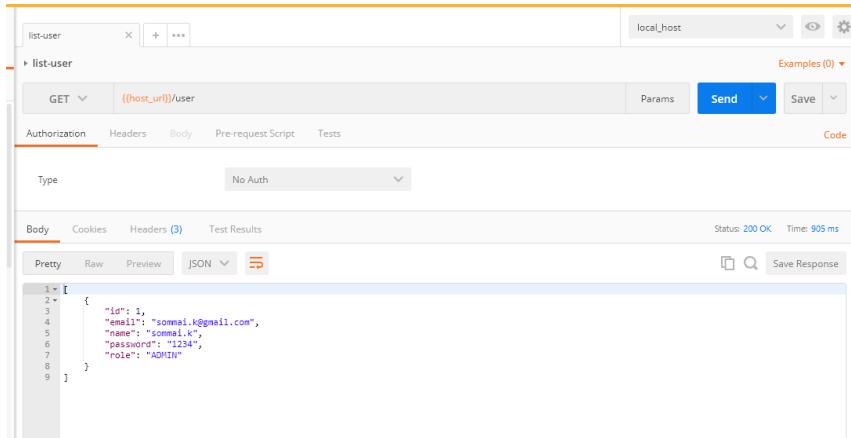
- เพิ่ม dependencyManagement ต่อจาก dependencies เข้าไปดังนี้

```
<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

- ทดลองการทำงานของ auth-service ด้วย postman



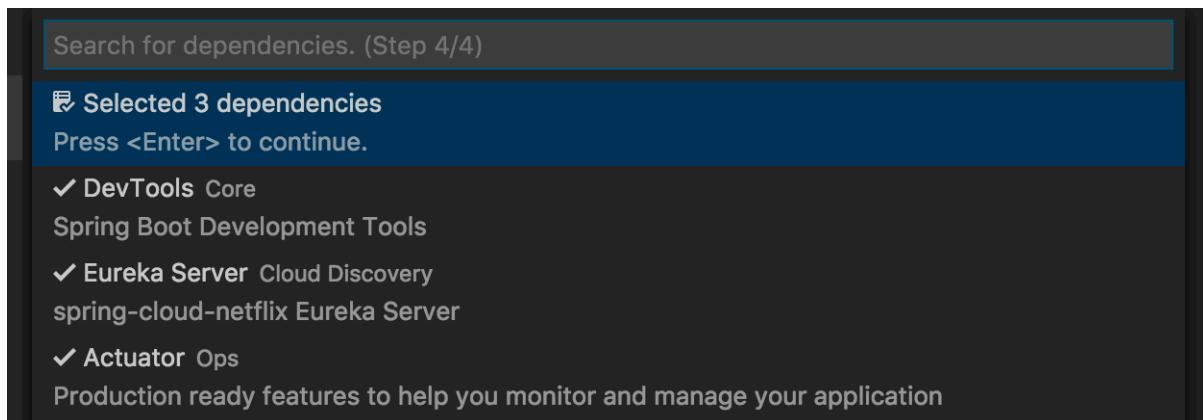
The screenshot shows the Postman interface with a successful API call. The URL is `({{host_url}})/user`. The response body is a JSON array containing one user object:

```
[{"id": 1, "email": "sonmai.k@gmail.com", "name": "sonmai.k", "password": "1234", "role": "ADMIN"}]
```

### 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

```
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);
    }
}
```

## MICROSERVICE WITH SPRING BOOT

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

The screenshot shows the Spring Eureka dashboard at <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:** Displays environment details (Environment: test, Data center: default), system metrics (Current time: 2018-07-07T16:28:12 +0700, Uptime: 00:01, Lease expiration enabled: false, Renews threshold: 1, Renews (last min): 0), and a log entry indicating 1000 instances since startup.

**DS Replicas:** Shows a single instance registered under the application name "localhost".

**General Info:** A table with columns Application, AMIs, Availability Zones, and Status, currently showing "No instances available".

## Exercise

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

### **\*\*แนวทarc\*\***

- 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

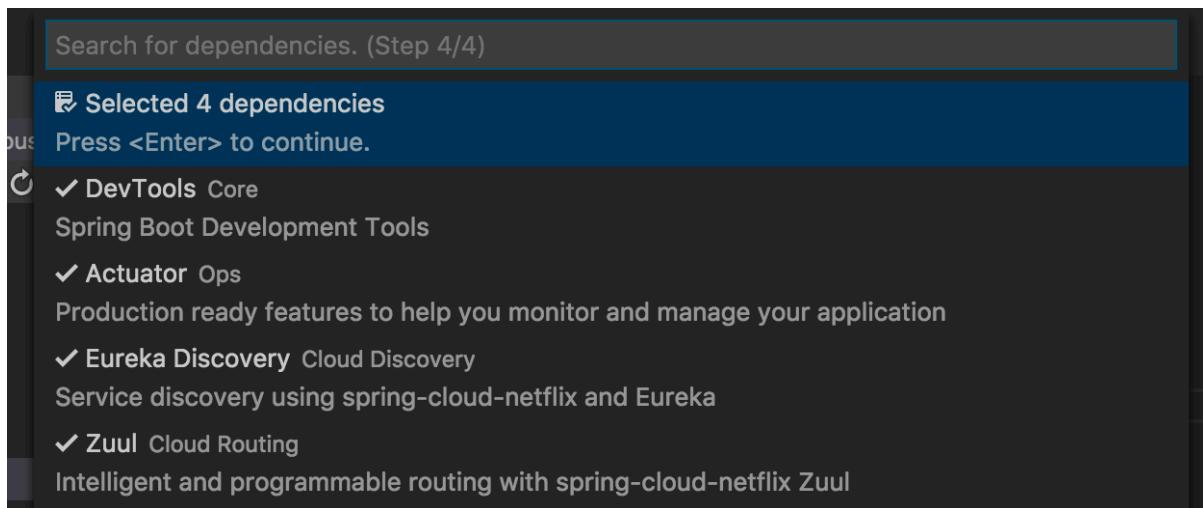
**General Info:**

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;

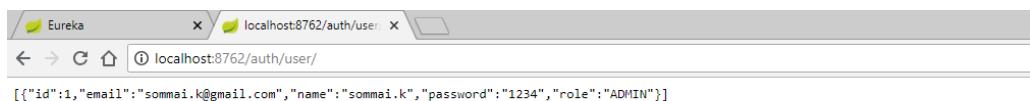
@SpringBootApplication
@EnableZuulProxy
@EnableEurekaClient
public class DemoApplication {

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

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

The screenshot shows the 'DS Replicas' section of the Eureka UI. It displays two instances registered with Eureka: 'AUTH-SERVICE' and 'ZUUL-SERVER'. Both instances are marked as 'UP (1) - localhost:auth-service:8081' and 'UP (1) - localhost:zuul-server:8762' respectively. Below this, the 'General Info' section provides system-level metrics such as total available memory (523mb), environment (test), number of cpus (4), current memory usage (113mb, 21%), and server uptime (00:26).

- 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

## MICROSERVICE WITH SPRING BOOT

### \*\*ແນວທາງ\*\*

- 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

#### វិធានការ

- clone project จาก github เพื่อເວົາ source code security มาໃຫ້ງານ โดยการរັນคำສັ່ງດັ່ງນີ້
  - git clone <https://github.com/Sommaik/springboot-workshop.git>
  - ຈະໄດ້ folder ດັ່ງນີ້

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 ແກ້ໄຂ proxy server ໃກ້ກຳກັນ check security ກ່ອນກາຣເຮຍກໃຊ້ service

#### វິທີການ

- ແກ້ໄຂ file pom.xml ຂອງ zuul\_server ເພີ່ມຮາຍກາຣດັ່ງນີ້ເຂົ້າໄປ

```
<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>
```

- ແກ້ໄຂ file application.properties ຂອງ zuul\_server ເພີ່ມຮາຍກາຣດັ່ງນີ້

```
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 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 แล้วนำ 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

The screenshot shows the Postman interface with the following details:

- Method: POST
- URL: localhost:8762/auth
- Tab: Body (selected)
- Type: raw (JSON)
- Body content:

```
1 {  
2   "username": "sommai.k@hotmail.com",  
3   "password": "12345"  
4 }
```
- Tab: Headers (9) (selected)

**authorization → Bearer**  
eyJhbGciOiJIUzUxMiJ9eyJzdWliOiJzb21tYWkua0Bob3RtYWlsLmNvbSIsImF1dGhvcmI0aWVzIjpblJPTI  
0jWEgEvsrfP9uNgvx7Y6XKDoGK9dm-yQzPAuz9PivTOldgQ

- เมื่อสั่ง run จะได้ token มาดังภาพ

## MICROSERVICE WITH SPRING BOOT

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

authorization → Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWliOiJzb21tYWkua0BnbWFpbC5jb20iLCJhdXRb3JpdGllcyI6WyJST0xFX0FETUIOll0slmIhdCl6MTUzMTAzMDoMywiZXhwIjoxNTMxMTE2NjlzfQ,pVrMBLN6ksU2\_8NJ0NHa17hsUZvDbbf9MK\_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

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

- เมื่อสั่ง run จะได้ผลลัพธ์เป็น 401 unauthorized มาดังภาพ

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

Authorization Headers (1) Body Pre-request Script Tests

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWliOiJzb21tYWkua0B...	
New key	Value	Description

- 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 eyJhbGciOiJIUzUxMiJ9eyJzdWliOiJzb21tYWkua0B...	
New key	Value	Description

- เมื่อสั่ง run จะได้ผลลัพธ์เป็นข้อมูลมาดังภาพ

```
[  
  {  
    "id": 1,  
    "email": "sommai.k@gmail.com",  
    "name": "sommai.k",  
    "password": "1234",  
    "role": "ADMIN"  
  }  
]
```

## MICROSERVICE WITH SPRING BOOT

### 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
```

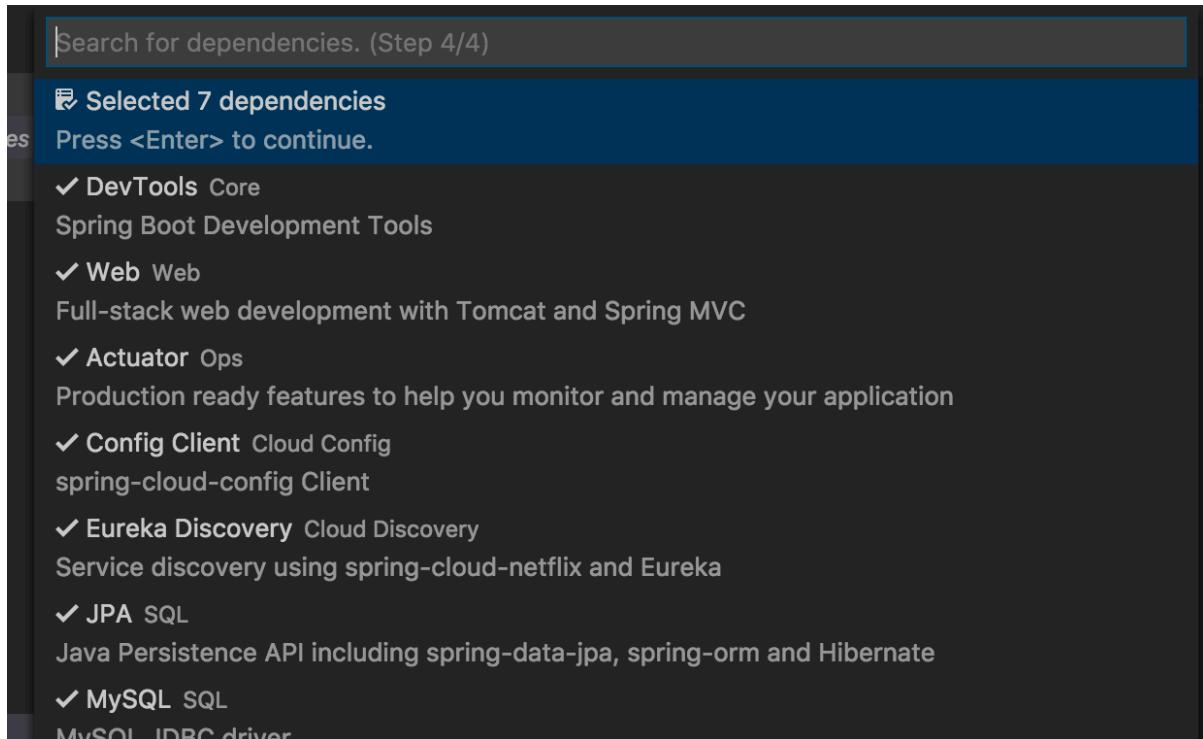
```
info.app.name=Order Service Application
info.app.description=Order Service on spring boot application
info.app.version=1.0.0
server.port=8082
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

**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

## MICROSERVICE WITH SPRING BOOT

```
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 ให้ครบถ้วนทุก field ด้วยนะครับ

**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 ឱ្យកសបុក្រោម field ដែលបានគ្រប់

**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;
    }
}

```

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

## MICROSERVICE WITH SPRING BOOT

### WORKSHOP#8 สร้าง Repository ของ Entity Item

#### วิธีการ

- สร้าง interface ชื่อ ItemRepository ภายใน package com.business.order.item หรือ 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 เพื่อกำเป็น rest api

#### วิธีการ

- สร้าง class ชื่อ SaleOrderController ภายใน 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

```
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;
    }
}
```

## MICROSERVICE WITH SPRING BOOT

- 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}