Spring Boot + Spring Cloud Open Feign Microservices Communication Example

author: Ramesh Fadatare

MICROSERVICES SPRING BOOT SPRING CLOUD

In the previous couple of tutorials we have seen:

Spring Boot Microservices Communication Example using RestTemplate.

Spring Boot Microservices Communication Example using WebClient

In this tutorial, we will learn how to use the `Spring Cloud Open Feign` library to make REST API calls (Synchronous communication) between multiple microservices.

Spring Cloud Open Feign Overview

Feign makes writing web service clients easier with pluggable annotation support, which includes Feign annotations and JAX-RS annotations. Also, Spring Cloud adds support for Spring MVC annotations and for using the same **HttpMessageConverters** as used in Spring Web.

One great thing about using Feign is that we don't have to write any code for calling the service, other than an interface definition.

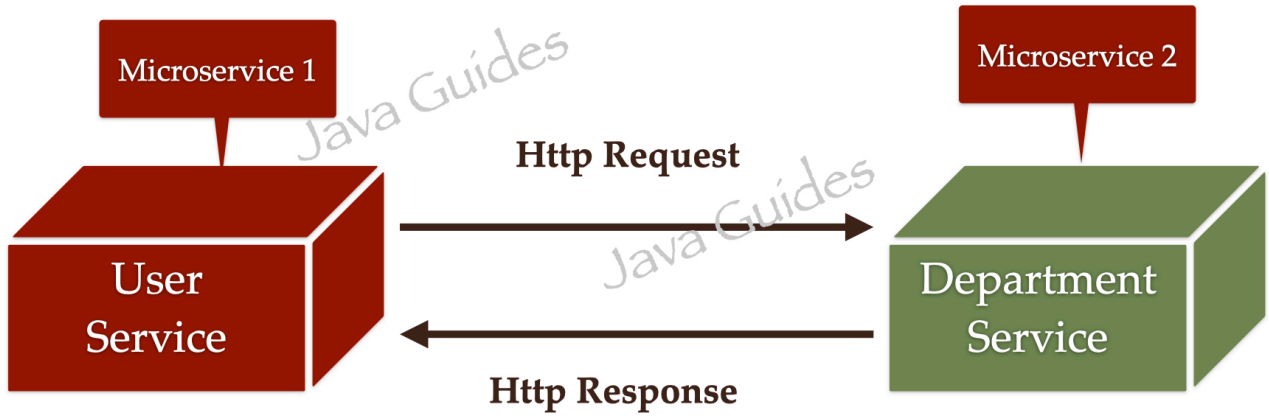
For example:

```
package net.javaguides.userservice.service;

import net.javaguides.userservice.dto.DepartmentDto;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@FeignClient(value = "DEPARTMENT-SERVICE", url = "http://localhost:8080")
public interface APIClient {
    @GetMapping(value = "/api/departments/{id}")
    DepartmentDto getDepartmentById(@PathVariable("id") Long departmentId);
}
```

What we will Build?

Well, we will create two microservices such as `department-service` and `user-service` and we will make a REST API call using `Spring Cloud Open Feign` from `user-service` to `department-service` to fetch a particular user department.

Microservices Communication using Spring Cloud OpenFeign



Prerequisites

Refer to the below tutorial to create `department-service` and `user-service` microservices:

Spring Boot Microservices Communication Example using RestTemplate.

Step 1: Add Spring cloud open feign Maven dependency to User-Service

Open the `pom.xml` file of the `user-service` project and add the below dependency:

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

Make sure to add spring cloud dependencies and their version.

Here is the complete `pom.xml` file after adding **Spring cloud open feign** dependency:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.7.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>net.javaguides</groupId>
	<artifactId>user-service</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>user-service</name>
	<description>user-service</description>
	<properties>
		<java.version>17</java.version>
```

```
		<spring-cloud.version>2021.0.4</spring-cloud.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-openfeign</artifactId>
		</dependency>

		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>
	<dependencyManagement>
		<dependencies>
			<dependency>
				<groupId>org.springframework.cloud</groupId>
				<artifactId>spring-cloud-dependencies</artifactId>
				<version>${spring-cloud.version}</version>
				<type>pom</type>
				<scope>import</scope>
			</dependency>
		</dependencies>
	</dependencyManagement>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
				<configuration>
					<excludes>
						<exclude>
							<groupId>org.projectlombok</groupId>
							<artifactId>lombok</artifactId>
						</exclude>
					</excludes>
				</configuration>
			</plugin>
		</plugins>
	</build>

</project>
```

Step 2: Enable Feign Client using @EnableFeignClients

```
package net.javaguides.userservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.openfeign.EnableFeignClients;
```

```
@SpringBootApplication
@EnableFeignClients
public class UserServiceApplication {

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

Note that `@EnableFeignClients` annotation enables component scanning for interfaces that declare they are Feign clients.

Step 3: Create feign API client

After that, we need to have a feign API client with the necessary methods, requests, and responses.

Let's create an interface named `APIClient` and add the following code:

```
package net.javaguides.userservice.service;

import net.javaguides.userservice.dto.DepartmentDto;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@FeignClient(value = "DEPARTMENT-SERVICE", url = "http://localhost:8080")
public interface APIClient {
    @GetMapping(value = "/api/departments/{id}")
    DepartmentDto getDepartmentById(@PathVariable("id") Long departmentId);
}
```

We declare a Feign client using the `@FeignClient` annotation:

```
@FeignClient(value = "DEPARTMENT-SERVICE")
```

The value argument passed in the `@FeignClient` annotation is a mandatory, arbitrary client name, while with the URL argument, we specify the API base URL.

```
@FeignClient(value = "DEPARTMENT-SERVICE", url = "http://localhost:8080")
```

Furthermore, since this interface is a Feign client, we can use the Spring Web annotations to declare the APIs that we want to reach out to.

Step 4: Change the getUser method to call APIClient

First, inject `APIClient` and then use it:

```
        DepartmentDto departmentDto = apiClient.getDepartmentById(user.getDepartmentId());
```

Here is the complete code of `UserServiceImpl` using Feign client for your reference:

```
package net.javaguides.userservice.service.impl;

import lombok.AllArgsConstructor;
import net.javaguides.userservice.dto.DepartmentDto;
import net.javaguides.userservice.dto.ResponseDto;
import net.javaguides.userservice.dto.UserDto;
import net.javaguides.userservice.entity.User;
import net.javaguides.userservice.repository.UserRepository;
import net.javaguides.userservice.service.APIClient;
import net.javaguides.userservice.service.UserService;
import org.springframework.stereotype.Service;
```

```
@Service
@AllArgsConstructor
public class UserServiceImpl implements UserService {

    private UserRepository userRepository;

    private APIClient apiClient;

    @Override
    public User saveUser(User user) {
        return userRepository.save(user);
    }

    @Override
    public ResponseDto getUser(Long userId) {

        ResponseDto responseDto = new ResponseDto();
        User user = userRepository.findById(userId).get();
        UserDto userDto = mapToUser(user);

        DepartmentDto departmentDto = apiClient.getDepartmentById(user.getDepartmentId());
        responseDto.setUser(userDto);
        responseDto.setDepartment(departmentDto);

        return responseDto;
    }

    private UserDto mapToUser(User user){
        UserDto userDto = new UserDto();
        userDto.setId(user.getId());
        userDto.setFirstName(user.getFirstName());
        userDto.setLastName(user.getLastName());
        userDto.setEmail(user.getEmail());
        return userDto;
    }
}
```

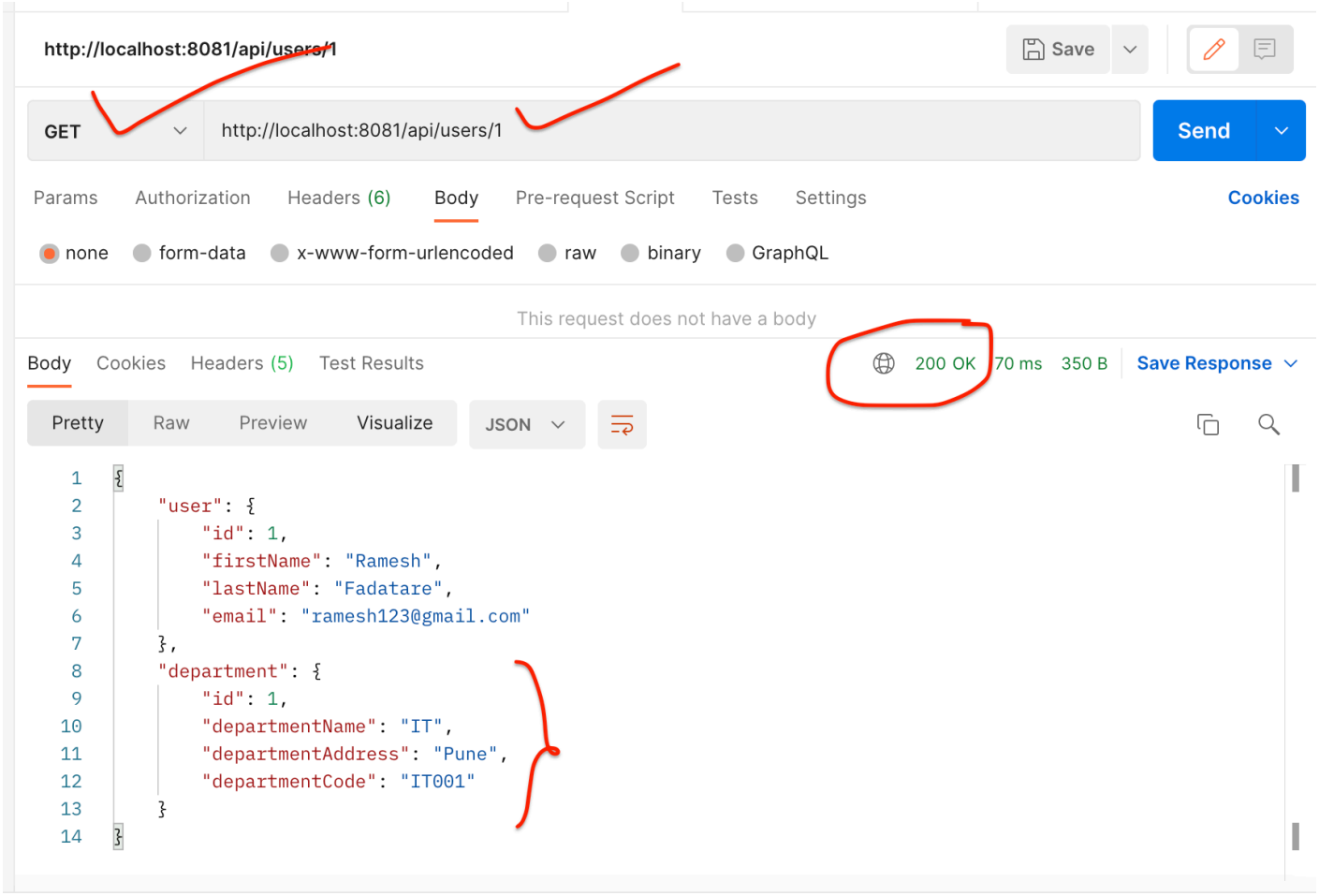
That's it. Now run both the Microservices and let's test.

Demo: Start Both Microservices

First, start the `department-service` project and then start a `user-service` project.

Once both the projects are up and running on different ports. Next, let's call the **Get User REST API** to test the `user-service` REST API call to the `department-service`.

Get User REST API:



Note that the response contains a Department for a User. This demonstrates that we have successfully made a REST API call from `user-service` to `department-service` using `WebClient`.

Conclusion

In this tutorial, we learned how to use the Spring Cloud Open Feign library to make REST API calls (Synchronous communication) between multiple microservices.