
Architecture Overview

Two microservices:

1. UserApp (Client Service)

- Runs on port **8090**
- Has an endpoint: **/accessApp**
- Uses **Feign** (a declarative REST client) to call another service (ProductApp)

2. ProductApp (Provider Service)

- Runs on port **8091**
- Has an endpoint: /getInfo
- Returns some product-related information

How the Request Flows

When you open a browser or use Postman to visit:

http://localhost:8090/accessApp

here's what happens:

Step 1: Request hits UserApp

This URL maps to the method in your controller:

```
@GetMapping("/accessApp")
public String retreiveInfo() {
   return proxy.getDetails();  // Makes call to ProductApp
}
```

This method uses a FeignClient to make a **REST call to another service (ProductApp).**

Step 2: Feign Client Triggers Remote Call

The **proxy.getDetails()** call triggers this Feign client:

```
@FeignClient(name="ProductApp", url="http://localhost:8091/")
public interface ServiceProxy {
    @GetMapping("/getInfo")
    String getDetails();
}
```

This sends a **GET request to:**

http://localhost:8091/getInfo

Step 3: ProductApp Responds

In your ProductApp, this controller handles the call:

```
@GetMapping("/getInfo")
public String getInfo() {
    return "Product information from ProductApp";
}
```

It returns a simple string like:

"Product information from ProductApp"

Step 4: UserApp Returns Response

That string is sent back to the original client — the browser or Postman — as the final response.

Example in Practice

When you call:

http://localhost:8090/accessApp

I**f both services are running**, you'll see:

Product information from ProductApp

If ProductApp is NOT running, you'll get an error:

Connection refused or 500 Internal Server Error

Summary

Component	Purpose
UserApp	Client that exposes /accessApp
FeignClient	Makes HTTP call to ProductApp
ProductApp	Returns data from /getInfo