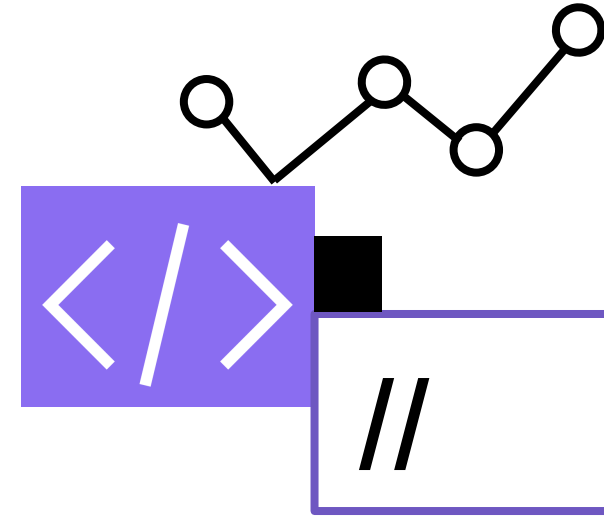
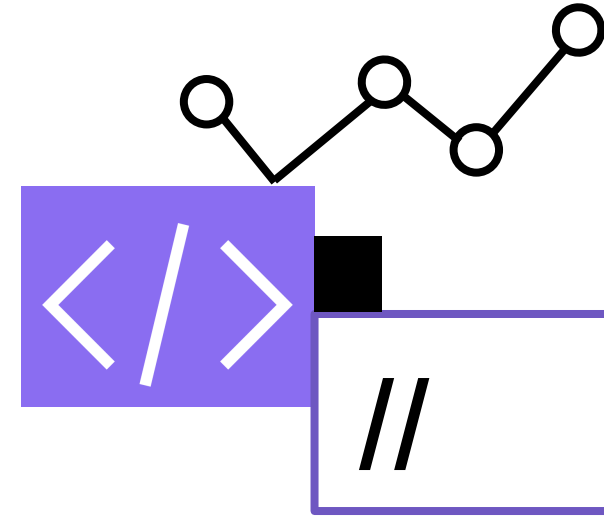


Invocaciones declarativas



¿Para qué sirven las invocaciones declarativas?

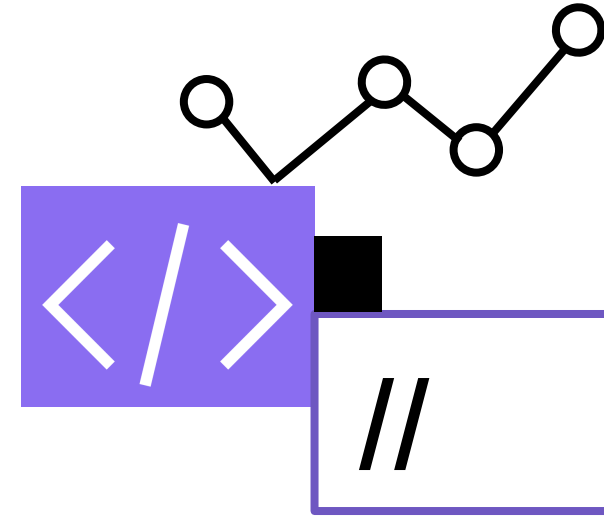
Para facilitar la integración entre microservicios mediante la creación de clientes HTTP de forma declarativa. Esto significa que se simplifica la invocación entre estos microservicios. El desarrollador simplemente necesita crear una interfaz y configurarla mediante anotaciones. No se necesita programar toda la lógica de conexión e invocación de una API, sino que simplemente se declara escribiendo anotaciones en los métodos que necesitamos invocar y/o exponer.



Spring Cloud Feign

Originalmente, fue desarrollado por Netflix. Actualmente, se integró dentro de los componentes de Spring Cloud, rebautizado como **Spring Cloud OpenFeign**. Gracias a la integración con Spring Cloud, dentro de sus ventajas podemos destacar:

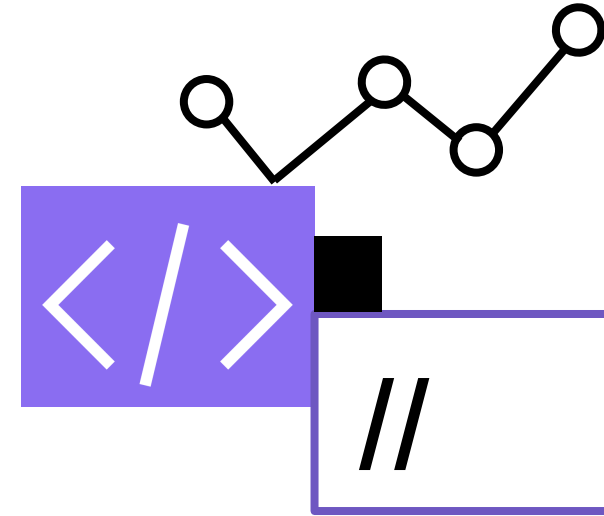
- ➔ **Autodescubrimiento:** utilizando **Eureka**, podemos hacer llamadas a los microservicios utilizando el nombre en lugar de una URL formada por IP y puerto.
- ➔ **Balanceo de carga:** podemos integrar Feign con un balanceador de carga como Spring Cloud LoadBalancer.



¿Cómo incluirlo en nuestro proyecto?

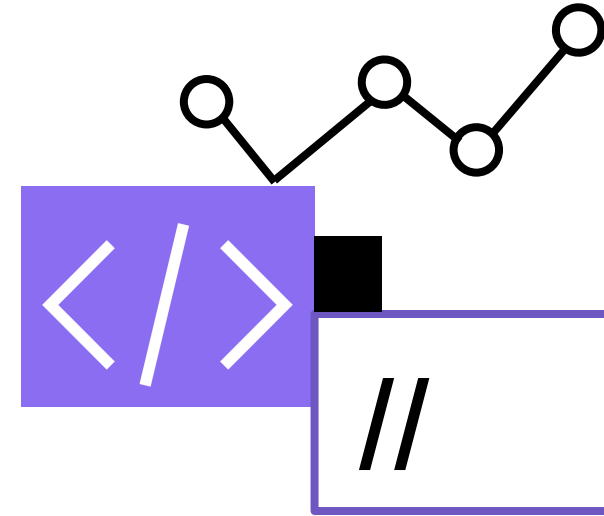
Primero debemos agregar la dependencia `spring-cloud-starter-openfeign` a nuestro proyecto.

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



Para habilitar el uso de Feign en nuestra aplicación vamos a la clase principal y agregamos la anotación `@EnableFeignClients`. Por ejemplo:

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



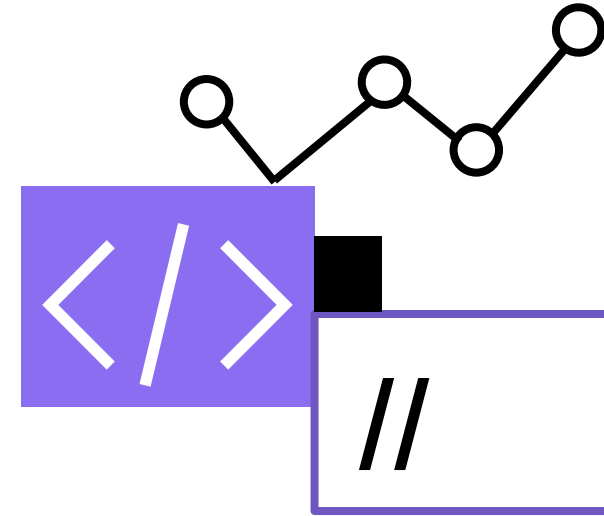
¿Cómo enviar peticiones a otro microservicio?

Envío de peticiones entre microservicios registrados en Eureka

Supongamos que tenemos un servicio de productos llamado **product-service**, el cual expone una API REST. Dentro de sus métodos tiene uno que nos retorna todos los productos mediante GET, la URL es <http://localhost:8080/products>.

Ya que el microservicio se encuentra registrado por Eureka, podemos acceder al mismo recurso reemplazando el hostname (localhost) y el puerto (8080) por el nombre con el que se registró. Entonces la URL nos queda <http://product-service/products>. Ahora, con Feign, utilizamos interfaces para crear los clientes y enviar las peticiones.

Veamos un ejemplo a continuación.



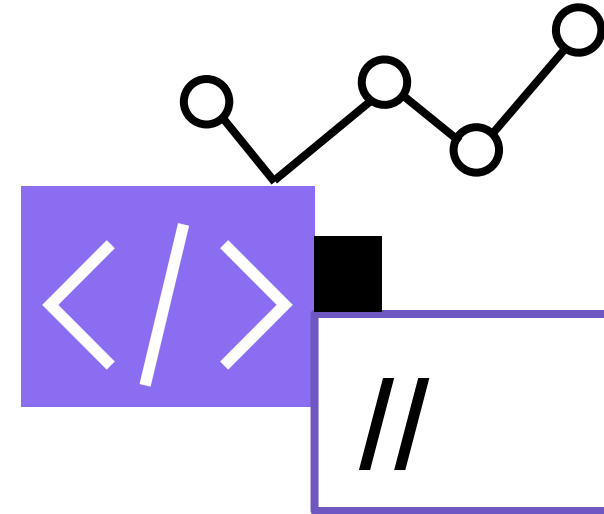
Primero creamos un DTO para guardar la respuesta obtenida.

```
public class ProductDTO {  
    private Integer id;  
    private String title;  
    private String color;  
}
```

Luego creamos la interfaz en donde vamos a configurar Feign para enviar la petición.

```
@FeignClient(name = "product-service")  
public interface ProductClient {  
    @RequestMapping(method = RequestMethod.GET, value = "/products")  
    List<ProductDTO> getProducts();  
}
```

Analicemos esta porción de código en detalle.



{código}

```
@FeignClient(name = "product-service")
public interface ProductClient {
    @RequestMapping(method = RequestMethod.GET,
value = "/products")
    List<ProductDTO> getProducts();
}
```

El valor pasado en la anotación `@FeignClient` es el nombre del cliente que estamos creando y también es utilizado para buscar la dirección física del microservicio al que queremos enviar la petición. Es decir, buscar en el registro de Eureka qué dirección (o direcciones) están asociadas a ese nombre.

{código}

```
@FeignClient(name = "product-service")
public interface ProductClient {
    @RequestMapping(method = RequestMethod.GET,
value = "/products")
    List<ProductDTO> getProducts();
}
```

Con la anotación `@RequestMapping` configuramos el método que vamos a utilizar y la URL. En este caso vamos a enviar GET a **/products**.

{código}

```
@FeignClient(name = "product-service")
public interface ProductClient {
    @RequestMapping(method = RequestMethod.GET,
value = "/products")
    List<ProductDTO> getProducts();
}
```

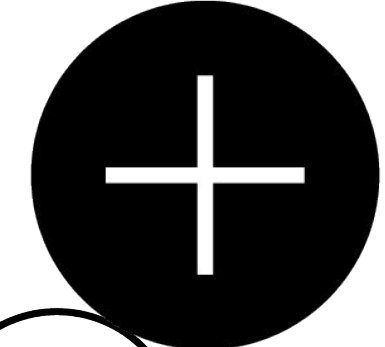
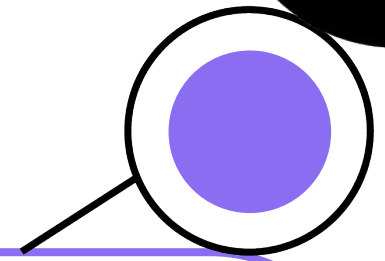
Podemos notar que el método `getProducts()` retorna una lista de `ProductDTO`. Feign cuenta con un decoder por defecto, que parseará el JSON recibido a la clase que nosotros le indiquemos.

¿Cómo enviar peticiones a otro microservicio?

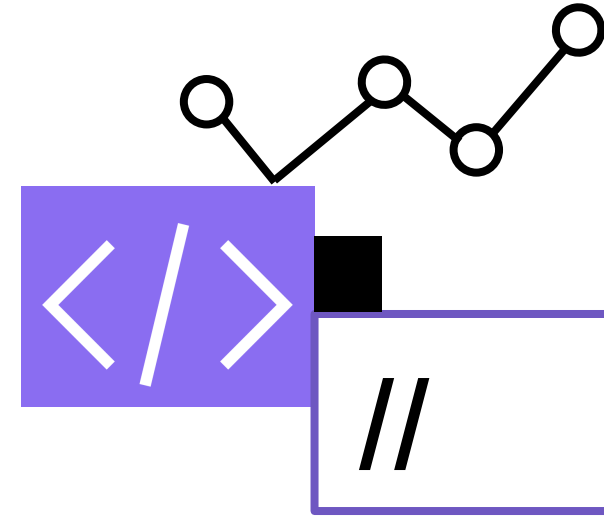
Envío de peticiones a servicios externos

También podemos enviar peticiones a servicios que sean externos a nuestro ecosistema o que no estén registrados con Eureka. **Para hacer esto debemos indicarle la URL.** Por ejemplo, supongamos que queremos saber desde qué parte del mundo visitan nuestro sitio. Para resolver esto realizamos una petición a una API que, dada una dirección IP, nos devuelve su ubicación. Nuestro endpoint es <http://ipwhois.app/json/{IP}> y configuramos el cliente Feign:

```
@FeignClient(name = "ip-service", url ="http://ipwhois.app/json")
public interface IpClient {
    @RequestMapping(method = RequestMethod.GET, value = "/{ip}")
    List<JsonNode> getStores(@PathVariable("ip") String ip);
}
```



Además de agregar el parámetro URL con la dirección, indicamos el nombre. En este caso, solo se utilizará para registrar el cliente en la aplicación.



Test

Por último, para ejecutar los métodos de la interfaz debemos inyectar la misma en un servicio utilizando Spring Boot, como muestra el siguiente ejemplo:

```
@Service
public class ProductService{

@Autowired
private ProductClient productFeingClient;

public List<ProductDTO> fetchAllProducts(){
    return productFeingClient.getProducts();
}
```

¡Muchas gracias!