Feign Client

Microservices

Beschreibung

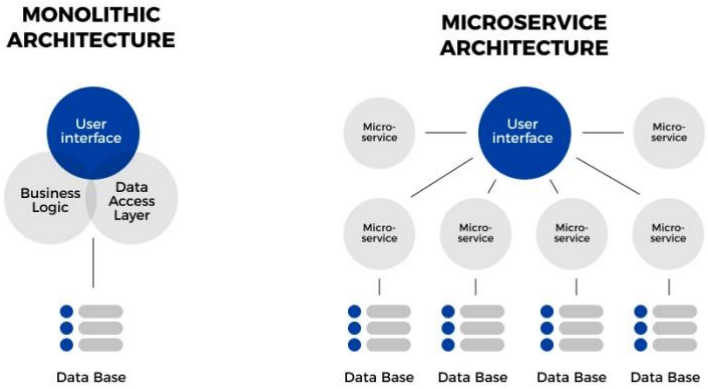
In diesem Dokument wird das Feign Client Konzept eingeführt. Feign ist ein deklarativer Web-Service-Client. Das Framework erleichtert das Schreiben von Clients in einer Microservice Umgebung und abstrahiert dabei den Zugriff auf eine REST-API.

Inhaltsverzeichnis

1 Feign Client

1.1 Einleitung

Während traditionelle monolithische Architekturen aus eng gekoppelten Anwendungen mit einer einzigen Codebasis bestehen, sind Microservices im Wesentlichen das Gegenteil. Sie sind eine Sammlung lose gekoppelter Dienste, von denen jeder für eine bestimmte Funktionalität verantwortlich ist. Diese Dienste müssen effizient kommunizieren. Die Kommunikation zwischen den Diensten erfolgt meistens via REST-APIs.



Wenn ein Microservice die API eines anderen Dienstes aufrufen möchte, verwenden Entwickler:innen häufig HTTP-Clients oder REST-Vorlagen, um diese Aufrufe durchzuführen. Diese erfordern eine Menge Standardcode, was die Pflege und das Verständnis der Codebasis erschwert.

→ Die @FeignClient-Annotation erleichtert dieses Vorgehen, indem sie die HTTP-Client-Schicht abstrahiert, sodass sich Entwickler:innen mehr auf die Geschäftslogik und weniger auf Infrastrukturbelange konzentrieren können. Feign ist ein deklarativer Web-Service-Client. Das Framework erleichtert das Schreiben von Clients. Um Feign zu verwenden, erstellen Sie ein Interface und annotieren diese entsprechend.

1.2 Microservices Aufbau

Im vorliegenden Beispiel soll eine Kommunikation zwischen dem Service-01 und dem Service-02 aufgebaut werden. Der Service-02 stellt eine neue REST-Schnittstelle zur Verfügung während der Service-01 mit einem Feign-Rest-Client auf eine API des Service-02 zugreifen soll.

Architektur Basisanwendung:

- App-Gateway
- Eureka-Server
- Service-01, API und Feign-Client auf Service-02
- Service-02, API

1.3 Vorbereitung Service-02

Der Service-02 soll mit einem neuen Rest-Controller erweitert werden. Dieser `RestController` liefert eine Liste von Büchern mit den Autoren. Die Liste wird als einfache Liste ohne Datenbank implementiert und mit einigen Büchern befüllt. Die Methode `getBooks()` stellt den Endpunkt dar.

```
@RestController
public class RESTBookController {

    private List<Book> books;

    public RESTBookController() {
        books = new ArrayList<>();
        books.add(new Book(1, "The Metamorphosis", "Franz Kafka"));
        books.add(new Book(2, "The Trial", "Franz Kafka"));
        books.add(new Book(3, "1984", "George Orwell"));
        books.add(new Book(4, "Animal Farm", "George Orwell"));
    }

    @GetMapping("books")
    public List<Book> getBooks() {
        return books;
    }
}
```

Die API ist unter folgender URL erreichbar: http://localhost:8081/books

1.4 Vorbereitung Service-01 als Feign-Client

Nun soll der Service-01 als Feign-Client vorbereitet werden. Die Client Funktionen werden nicht mit konventionellem Code implementiert, sondern die Abstraktion von Feign genutzt. Hierfür wird ein Interface definiert, welches der neuen Rest-Schnittstelle (API) des Services-02 entspricht.

Die Dependency für den Feign-Client sieht wie folgt aus:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-openfeign</artifactId>
  <version>4.0.2</version>
</dependency>
```

→ Die Anwendung wird mit der Annotation `@EnableFeignClient` annotiert.
Durch das Hinzufügen von `@EnableFeignClients` weisen Sie Spring an, nach Schnittstellen zu suchen, die mit @FeignClient annotiert sind. Anhand von diesen wir Feign die entsprechenden Proxy-Implementierungen generieren.

```
@SpringBootApplication
@EnableFeignClients
public class Service01Application {
    public static void main(String[] args) {
        SpringApplication.run(Service01Application.class, args);
    }
}
```

Feign gibt im Wesentlichen eine Möglichkeit, vereinfachte HTTP-Clients zu schreiben. Die Grundidee seiner Funktionsweise besteht darin, eine Schnittstelle zu schreiben und diese zu annotieren. Spring füllt dann die Lücken, indem es zur Laufzeit eine Implementierung bereitstellt.

Im Folgenden gehen wird eine Schnittstelle definiert, welche einen Feign-Client verwendet. Dieser Client ist für die Interaktion mit Service-02 konfiguriert.

```
@FeignClient("service-02")
public interface BooksClient {
    @GetMapping("books")
    List<Book> getBooks();
}
```

Die entsprechende Datenklasse muss ebenfalls übernommen werden.

```
@Data
@AllArgsConstructor
public class Book {
  private int id;
  private String title;
  private String author;
}
```

2 Aufgaben

2.1 Aufgabe 1 – Service-01 und 02 vorbereiten

Erweitern Sie den Service-01 und 02 gemäss der obigen Anleitung. Testen Sie die REST-Schnittstelle vom Service-02 mit der entsprechenden URL bevor Sie weiterfahren.

2.2 Aufgabe 2 – Feign-Client nutzen

Im MainController soll der Feign-Client genutzt werden. In einer Rest-Methode soll auf den Service-02 zugegriffen und die geladenen Daten verarbeitet werden.

Der Feign-Client kann als Interface über die `@Autowired` Injected werden.

```
@RestController
public class MainController {

    @Autowired
    private BooksClient booksClient;

}
```

Der Service-01 stellt einen neuen Endpunkt zur Verfügung. Dieser Endpunkt wandelt die Bücher-Objekte in einfache Titel um und sendet die Liste an die aufrufende Anwendung.

```
@GetMapping("api/books")
public List<String> getBooks() {
  return booksClient.getBooks().stream()
    .map(book -> String.format("%s (%s)",book.getTitle(),book.getAuthor()))
    .toList();
}
```

2.3 Aufgabe 3 – Bücher hinzufügen

Erweitern Sie den Service-02, sodass er über die REST-Schnittstelle ein neues Buch erfassen kann.

Fügen Sie das folgende Buch hinzu:

```
{
  "id": 5,
  "title": "Harry Potter and the Philosopher's Stone",
  "author": "J.K. Rowling"
}
```

3 Anhang

3.1 Weiterführende Dokumentation

Feign Client aus den folgenden zwei Artikeln und weiteren Recherchen:

Introduction to Spring Cloud Netflix – Eureka
https://medium.com/@AlexanderObregon/navigating-client-server-communication-with-springs-feignclient-annotation-70376157cefd
https://www.baeldung.com/spring-cloud-netflix-eureka

Ribbon (Load Balancer)
Chapter 6: 6. Client-Side Load Balancer: Ribbon
https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-ribbon.html

Chapter 7: Declarative REST Client: Feign
https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-feign.html

→ Das Kapitel 7 zeigt sehr viele Details bei der Anwendung von Feign.
→ mit Feign Hystrix Fallbacks (7.5)