Learning Consolidation

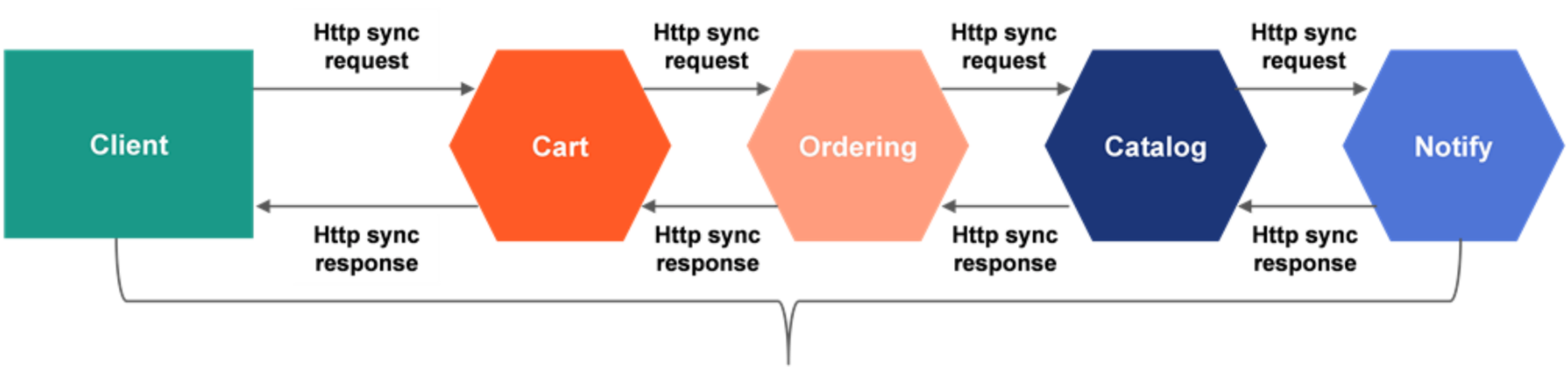
Establish Synchronous Communication Among Microservices by Using Feign Client

Microservices Communication

- The microservice architecture pattern is a distributed system running on different machines as a process or service.
- Each component of the system needs to interact with one another and coordinate its actions to effectively handle client requests.
- Services often collaborate to handle requests. Consequently, they must use an inter-process communication protocol.
- One of the most fundamental decisions when implementing a system based on the microservice architecture is to determine how microservices communicate with one another
- There are two types of microservices communication:
  - Synchronous
  - Asynchronous

Synchronous Communication

- In synchronous communication, one microservice will communicate with another through a rest endpoint over HTTP protocol.
- In this approach, the calling service will wait until the caller service responds.
- In synchronous communication, a "chain" of requests is created between microservices while serving the client request.



Communication Between Microservices

The client sends a request for a service.

/login

/register

/crud

API Gateway

/login

/register

/crud

saveUser

email

password

Feign Client

UserMovieService

UserAuthenticationSevice

MongoDB

MySQL

Eureka Server

Synchronous Communication Between Microservices (contd.)

- A client sends a request to /register with the UserMovieService. The moment the user registers, the user data is stored in the MongoDB of the UserMovieService.
- The details of email and password need to be saved in the UserAuthenticationService. Earlier, we made a REST call using /save after a user registered.
- But now you will make the UserMovieService call the UserAuthenticationService with user data once a user registers.
- UserMovieService calls the UserAuthenticationService once a user registers.
- This is called microservices communication as one microservice is calling another microservice.
- This process is also called orchestration of microservices.
- Only if a user registers can the user data be saved, so only after /register is called, the /save will be called. This is the synchronous type of communication.
- This type of synchronous communication can be achieved using Feign Client.

Slide Note

How to Implement the Feign Client?

- **Step 1: Add dependency to the pom.xml of `UserMovieService`.**

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

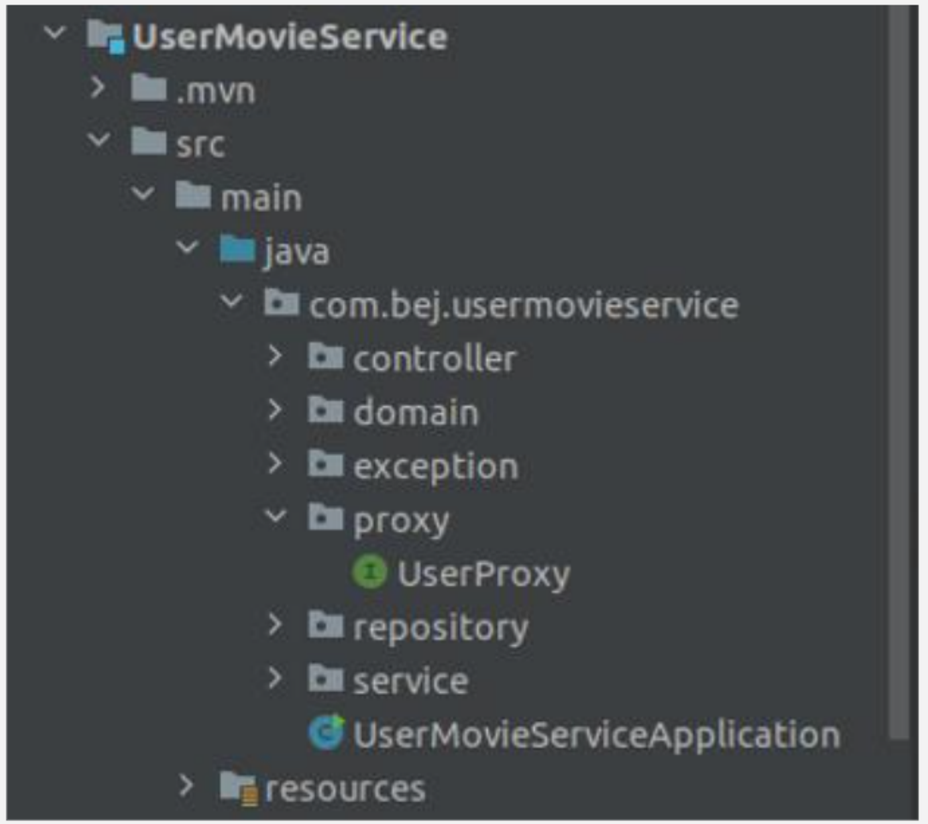
- **Step 2: Enable Feign usage in the main application of UserMovieService.**

```java
@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients
public class UserMovieServiceApplication {

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

}
```

Slide Note

Proxy Users

- **Step 3:**
  - **Create a `UserProxy` interface in the `UserMovieService` that will be used to communicate with the `UserAuthenticationService`.**



Slide Note

User Proxy Interface

```java
@FeignClient(name="user-authentication-service",url="localhost:8085")
public interface UserProxy {
    @PostMapping("/api/v1/user")
    public ResponseEntity<?> saveUser(@RequestBody User user);
}
```

- The proxy interface is used to make outbound API calls to the other service; in this case, it is the UserAuthenticationService
- Annotate the UserProxy with @FeignClient annotation.
- The annotation takes two parameters:
  - name – The service for which the call is made.
  - url – This is the path to the service.

Slide Note

Service Layer

```java
@Service
public class UserMovieServiceImpl implements UserMovieService{
    private UserProxy userProxy;
    private UserMovieRepository userMovieRepository;
    @Autowired
    public UserMovieServiceImpl(UserProxy userProxy, UserMovieRepository userMovieRepository) {
        this.userProxy = userProxy;
        this.userMovieRepository = userMovieRepository;
    }
```

- **Step 4:**
  - Autowire the proxy in the service layer.

Slide Note

Service Layer (contd.)

```java
@Override
public User registerUser(User user) throws UserAlreadyExistsException {
    if(userMovieRepository.findById(user.getEmail()).isPresent())
    {
        throw new UserAlreadyExistsException();
    }
    User savedUser = userMovieRepository.save(user);
    if(!(savedUser.getEmail().isEmpty())) {
        ResponseEntity r = userProxy.saveUser(user);
        System.out.println(r.getBody());
    }
    return savedUser;
}
```

- **Step 5:**
  - The user is getting registered; if the user gets registered successfully, then the proxy is calling the saveUser method.
  - UseProxy is used to send the data to the `UserAuthenticationService`.
- **Step 6:**
  - Run the application and test in Postman.