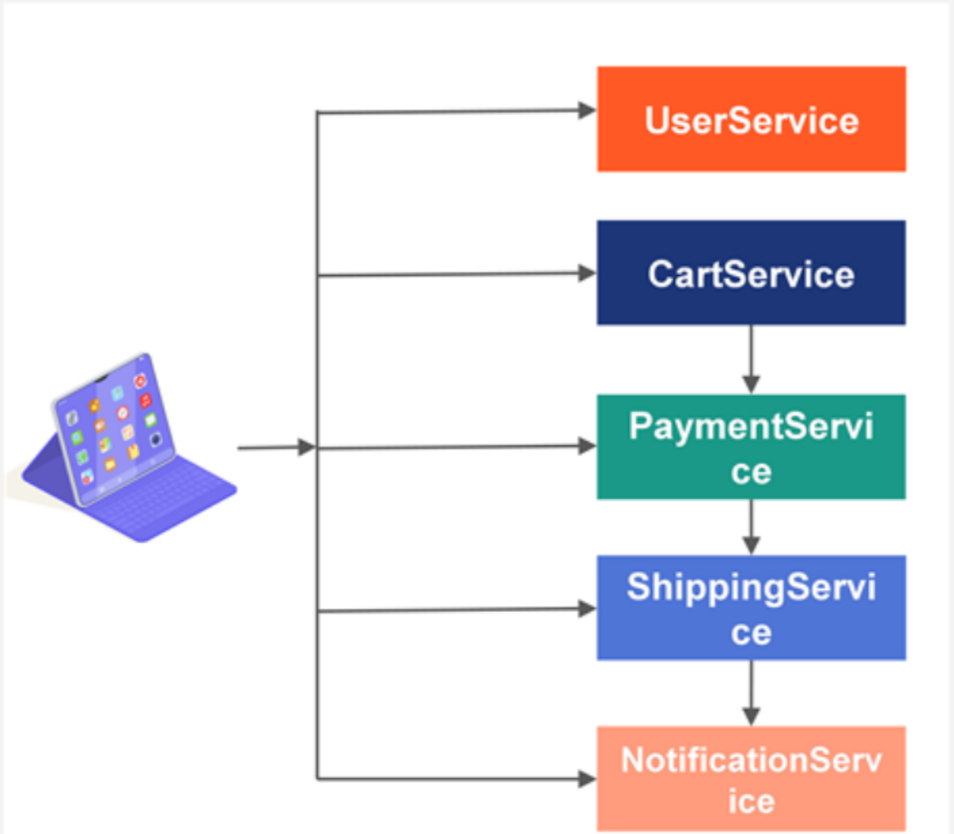
eBay

- A large e-commerce application like eBay caters to many customers around the globe.
- Assume that to build the application, there are multiple microservices involved.
- Can you share the process of buying a product on eBay?

Slide Note

While making an online purchase, a user needs to register in the shopping application and then add products to their cart, after which the user can choose a payment option and be redirected to the payment gateway.



Application Workflow

- To make an online purchase, a user must register for an application and later add products to their cart.
- After users select the payment option, they are redirected to the payment gateway.
- The order is confirmed only after the payment is made.
- An application has multiple services that cater to different requests from multiple clients.

Slide Note

While making an online purchase, you need to register in the shopping application, and then buy products by adding products to the cart, after which we choose a payment option and are redirected to the payment gateway. After payment is made, the shipping details are captured and the order is confirmed.

Think and Tell

- Do you think all these operations occur in a sequential manner?
- Is it important for the services to interact with each other to effectively process an application?
- How do they communicate?
- After a product is shipped to the user, is it mandatory for the `ShippingService` to send an acknowledgment to the `NotificationService` to notify the user that the product has been shipped?

Slide Note

Establish Synchronous Communication Among Microservices by Using Feign Client

Menu

00:04:00:00:23

04/ 23

Slide Note

Learning Objectives

- Explain Microservices Communication
- Explore the Types of Communication
- Implement Feign Client to Establish Synchronous Communication Between Microservices

Slide Note

Explain Microservices Communication

Slide Note

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

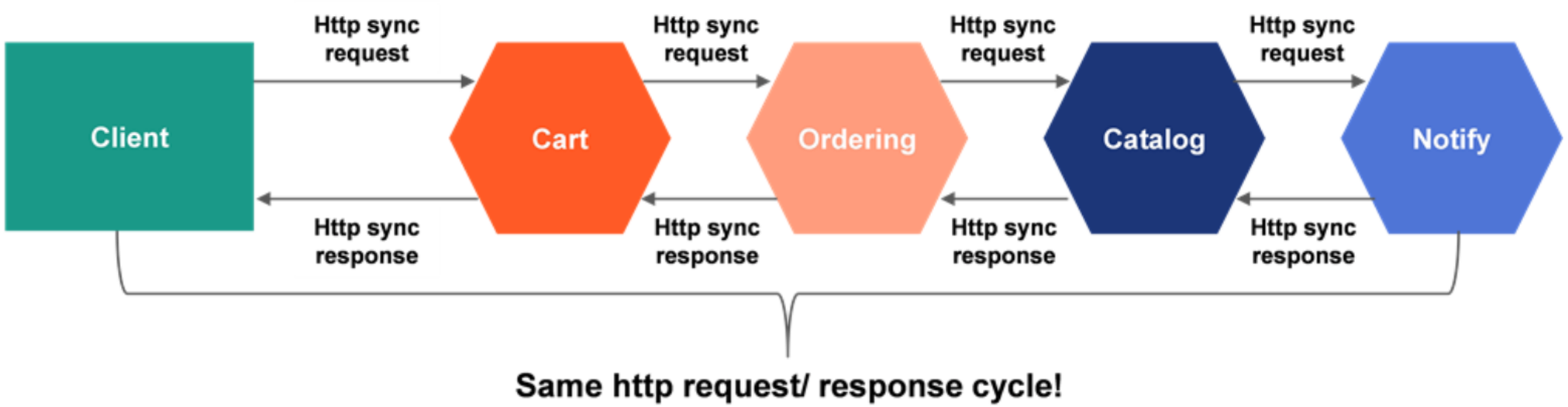
Slide Note

Explore the Types of Communication

Slide Note

Synchronous Communication

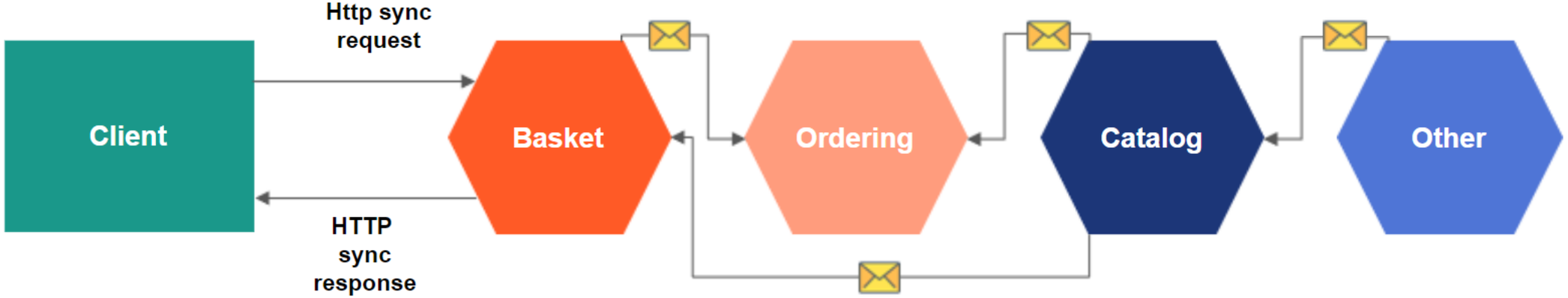
- In synchronous communication, one microservice will communicate with another through a rest endpoint over HTTP protocol.
- In this approach, the calling service will wait until the caller service responds.
- In synchronous communication, a "chain" of requests is created between microservices while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



Slide Note

Asynchronous Communication

- In asynchronous communication, microservices use asynchronous messages or HTTP polling to communicate with other microservices.
- The calling service will not wait for a response from the caller service.
- Asynchronous communication in microservices can be accomplished through message brokers like Apache Kafka, RabbitMQ, etc.



Slide Note

Synchronous Communication Between Microservices

/register

/register

Feign Cleint

/saveUser

/saveUser

UserMovieService

UserAuthenticationSevice

Slide Note

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

Implement Feign Client to Establish Synchronous Communication Between Microservices

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>
```

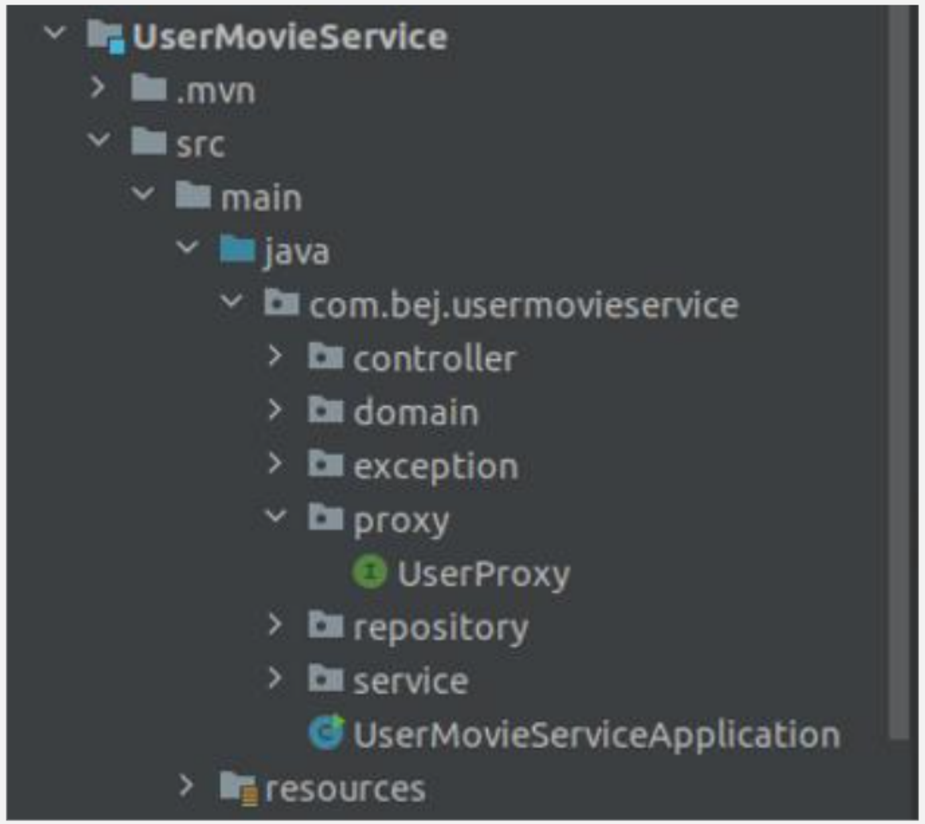
- **Step 2: Enable Feign usage in the main application of UserMovieService.**

```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`.**



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.

Slide Note

Streaming Application

Consider a streaming application that allows users to watch movies on any smart device. The application provides multiple features for all its registered users. Users must register with the application to access some of its features. Let us create multiple microservices for the streaming application.

1. A user must first register with the application.
2. Use credentials such as Id and password to log in.
3. Access the features provided by the streaming application, like adding favorites, compiling a watch list, etc.

DEMO

Slide Note

Now, let us create a parent project called MovieApplication. This will contain the UserAuthenticationService and the UserMovieService as microservices. Implement Feign Client to send the user data to the UserAuthenticationService once a new user registers.

Check the solution here.

DEMO

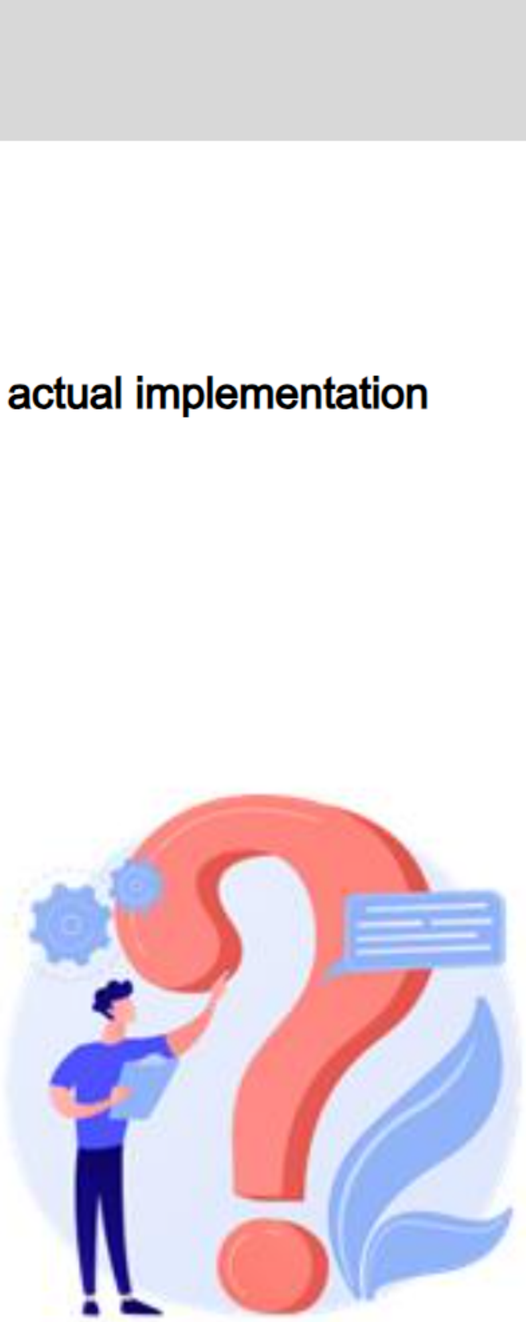
Slide Note

Quick Check

In Feign Client implementation, we declare and annotate a proxy interface while the actual implementation is provided at _____.

1. compile time

2. runtime



Slide Note

Quick Check: Solution

In Feign Client implementation, we declare and annotate a proxy interface while the actual implementation is provided at _____.

1. compile time
2. runtime