pom.xml

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

</dependency>

</dependencies>

application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/userdb

spring.datasource.username=root

spring.datasource.password=password

spring.jpa.hibernate.ddl-auto=update

User.java:

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

// getters and setters

}

UserRepository.java

package com.example.userservice.repository;

import com.example.userservice.entity.User;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface UserRepository extends JpaRepository<User, Long> {

// You can add custom queries here if needed, for example:

User findByUsername(String username);

}

UserController.java

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserRepository repository;

@PostMapping

public User save(@RequestBody User user) {

return repository.save(user);

}

@GetMapping("/{id}")

public ResponseEntity<User> get(@PathVariable Long id) {

return ResponseEntity.of(repository.findById(id));

}

}

Order Service :

pom.xml

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-webflux</artifactId>

</dependency>

```

Order.java

@Entity

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private Long userId;

private String product;

// getters and setters

}

```

OrderService.java

@Service

public class OrderService {

@Autowired

private WebClient.Builder webClientBuilder;

public Mono<User> getUser(Long userId) {

return webClientBuilder.build()

.get()

.uri("http://localhost:8081/users/" + userId)

.retrieve()

.bodyToMono(User.class);

}

}

---

## 2. Inventory Management System with Service Discovery

### Eureka Server

\*\*pom.xml\*\*

```xml

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

```

\*\*Application.java\*\*

```java

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

```

\*\*application.yml\*\*

```yaml

eureka:

client:

register-with-eureka: false

fetch-registry: false

server:

port: 8761

```

### Product and Inventory Services

\*\*pom.xml\*\* (Add Eureka Client, Spring Data JPA, MySQL)

```xml

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

```

\*\*application.yml\*\*

```yaml

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

```

---

## 3. API Gateway

\*\*pom.xml\*\*

```xml

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-gateway</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

```

\*\*application.yml\*\*

```yaml

spring:

cloud:

gateway:

routes:

- id: customer-service

uri: lb://CUSTOMER-SERVICE

predicates:

- Path=/customer/\*\*

- id: billing-service

uri: lb://BILLING-SERVICE

predicates:

- Path=/billing/\*\*

default-filters:

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 10

redis-rate-limiter.burstCapacity: 20

```

---

## 4. Resilient Microservices with Circuit Breaker

\*\*pom.xml\*\*

```xml

<dependency>

<groupId>io.github.resilience4j</groupId>

<artifactId>resilience4j-spring-boot2</artifactId>

</dependency>

```

\*\*application.yml\*\*

```yaml

resilience4j:

circuitbreaker:

instances:

payment:

registerHealthIndicator: true

slidingWindowSize: 10

failureRateThreshold: 50

```

\*\*PaymentService.java\*\*

```java

@Service

public class PaymentService {

@Autowired

private CircuitBreakerFactory cbFactory;

public String makePayment() {

CircuitBreaker cb = cbFactory.create("payment");

return cb.run(() -> {

// Simulate call to slow 3rd-party API

return restTemplate.getForObject("https://slow-api.com/pay", String.class);

}, throwable -> "Payment fallback response");

}

}