**Microservices with Spring Boot 3 and Spring Cloud**

**Microservices with API gateway/Create Eureka Discovery Server and register**

**Register a Microservice:**

**Dependencies:**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

**Main class:**

@SpringBootApplication

public class CustomerServiceApplication {

public static void main(String[] args) {

SpringApplication.run(CustomerServiceApplication.class, args);

}

}

**Application.yml:**

server:

port: 8081

spring:

application:

name: customer-service

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

**Eureka Server:**

**Main class:**

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

**Application.yml:**

server:

port: 8761

spring:

application:

name: eureka-server

eureka:

client:

register-with-eureka: false

fetch-registry: false

**Dependency Management:**

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-dependencies</artifactId>

<version>2021.0.8</version> <!-- or latest -->

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

**Pom.xml:**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

**Microservices with Spring Boot 3 and Spring Cloud:**

**Build a User and Order Management System**

**Mainapp.java:**

@SpringBootApplication

@EnableFeignClients

public class OrderServiceApplication {

public static void main(String[] args) {

SpringApplication.run(OrderServiceApplication.class, args);

}

}

**Order.java:**

@Entity

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private Long userId;

private String product;

private Double amount;

}

**OrderController.java:**

@RestController

@RequestMapping("/orders")

public class OrderController {

@Autowired

private OrderRepository repo;

@Autowired

private UserClient userClient;

@PostMapping

public Order placeOrder(@RequestBody Order order) {

userClient.getUser(order.getUserId());

return repo.save(order);

}

@GetMapping("/{id}")

public Order getOrder(@PathVariable Long id) {

return repo.findById(id).orElseThrow();

}

@GetMapping

public List<Order> getAllOrders() {

return repo.findAll();

}

}

**OrderRepository.java:**

public interface OrderRepository extends JpaRepository<Order, Long> {}

**User.java:**

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

}

**UserClient.java:**

@FeignClient(name = "user-service", url = "http://localhost:8081")

public interface UserClient {

@GetMapping("/users/{id}")

User getUser(@PathVariable Long id);

}

**UserController.java:**

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserRepository repo;

@PostMapping

public User saveUser(@RequestBody User user) {

return repo.save(user);

}

@GetMapping("/{id}")

public User getUser(@PathVariable Long id) {

return repo.findById(id).orElseThrow();

}

@GetMapping

public List<User> getAllUsers() {

return repo.findAll();

}

}

**UserRepository.java:**

public interface UserRepository extends JpaRepository<User, Long> {}

**Application.properties:**

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

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

server.port=8081

**Inventory Management System with Service Discovery**

**Enable rate limiting:**

@Configuration

public class RateLimiterConfig {

@Bean

public KeyResolver userKeyResolver() {

return exchange -> Mono.just("global"); // Or extract user/IP for more granularity

}

}

**Example requests:**

GET http://localhost:8080/customers/101

→ Routes to → http://localhost:8081/101

GET http://localhost:8080/billing/invoice/567

→ Routes to → <http://localhost:8082/invoice/567>

**Application.yml:**

server:

port: 8080

spring:

application:

name: api-gateway

redis:

host: localhost

port: 6379

cloud:

gateway:

routes:

- id: customer-service

uri: http://localhost:8081

predicates:

- Path=/customers/\*\*

filters:

- RewritePath=/customers/(?<segment>.\*), /$\{segment}

- RequestRateLimiter=redis-rate-limiter[replenishRate=5,burstCapacity=10]

- id: billing-service

uri: http://localhost:8082

predicates:

- Path=/billing/\*\*

filters:

- RewritePath=/billing/(?<segment>.\*), /$\{segment}

- RequestRateLimiter=redis-rate-limiter[replenishRate=3,burstCapacity=5]

default-filters:

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 10

redis-rate-limiter.burstCapacity: 20

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka>

**Pom.xml:**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-data-redis-reactive</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

**Implement a Gateway**

**Configo repo**

**Inventory-service.yml:**

server:

port: 8082

spring:

datasource:

url: jdbc:mysql://localhost:3306/inventorydb

username: root

password: yourpassword

jpa:

hibernate:

ddl-auto: update

**Product-service.yml:**

server:

port: 8081

spring:

datasource:

url: jdbc:mysql://localhost:3306/productdb

username: root

password: yourpassword

jpa:

hibernate:

ddl-auto: update

**Eureka Server**

**Dependencies:**

<dependency>

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

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

</dependency>

**Main class:**

@EnableEurekaServer

@SpringBootApplication

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

**Application.yml:**

server:

port: 8761

eureka:

client:

register-with-eureka: false

fetch-registry: false

**Inventory Service**

**Inventory.java:**

@Entity

public class Inventory {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private Long productId;

private Integer quantity;

}

**InventoryController.java:**

@RestController

@RequestMapping("/inventory")

public class InventoryController {

@Autowired

private InventoryRepository repo;

@PostMapping

public Inventory updateInventory(@RequestBody Inventory inventory) {

return repo.save(inventory);

}

@GetMapping("/product/{productId}")

public Inventory getByProduct(@PathVariable Long productId) {

return repo.findByProductId(productId);

}

}

**InventoryRpository.java:**

public interface InventoryRepository extends JpaRepository<Inventory, Long> {

Inventory findByProductId(Long productId);

}

**Bootstrap.yml:**

spring:

application:

name: inventory-service

cloud:

config:

uri: http://localhost:8888

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka>

**Product Service**

**Dependencies:**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

</dependencies>

**Product.java:**

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private Double price;

}

**ProductController.java:**

@RestController

@RequestMapping("/products")

public class ProductController {

@Autowired

private ProductRepository repo;

@PostMapping

public Product saveProduct(@RequestBody Product product) {

return repo.save(product);

}

@GetMapping

public List<Product> getAll() {

return repo.findAll();

}

}

**ProductRepository.java:**

public interface ProductRepository extends JpaRepository<Product, Long> {}

**Bootstrap.yml:**

spring:

application:

name: product-service

cloud:

config:

uri: http://localhost:8888

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka>

**Dependenies:**

<dependency>

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

<artifactId>spring-cloud-config-server</artifactId>

</dependency>

**Main Class:**

@EnableConfigServer

@SpringBootApplication

public class ConfigServerApplication {

public static void main(String[] args) {

SpringApplication.run(ConfigServerApplication.class, args);

}

}

**Application.yml:**

server:

port: 8888

spring:

cloud:

config:

server:

git:

uri: file:///C:/path-to-local-git/config-repo