Eureka Server + Product Service + Order Service + Payment Service - Full Spring Boot Code

EurekaServerApplication.java

```
package com.example.eurekaserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

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

eureka-server application.properties

```
spring.application.name=eureka-server
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

Product.java

```
package com.example.productservice.entity;

import jakarta.persistence.*;
import lombok.Data;

@Entity
@Data
public class Product {
    @Id
    private String id;
    private String name;
    private double price;
    private int stock;
}
```

ProductRepository.java

```
package com.example.productservice.repository;

import com.example.productservice.entity.Product;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, String> {}
```

ProductController.java

```
package com.example.productservice.controller;

import com.example.productservice.entity.Product;
import com.example.productservice.repository.ProductRepository;
import org.springframework.web.bind.annotation.*;
import java.util.List;
```

```
@RestController
@RequestMapping("/products")
public class ProductController {

    private final ProductRepository repo;

    public ProductController(ProductRepository repo) {
        this.repo = repo;
    }

    @PostMapping
    public Product addProduct(@RequestBody Product product) {
        return repo.save(product);
    }

    @GetMapping
    public List<Product> getAllProducts() {
        return repo.findAll();
    }

    @GetMapping("/{id}")
    public Product getProduct(@PathVariable String id) {
        return repo.findById(id).orElse(null);
    }

    @PutMapping("/{id}/reduceStock")
    public Product reduceStock(@PathVariable String id, @RequestParam int qty) {
        Product p = repo.findById(id).orElseThrow();
        p.setStock(p.getStock() - qty);
        return repo.save(p);
    }
}
```

product-service application.properties

```
spring.application.name=product-service
server.port=8081
spring.datasource.url=jdbc:mysql://localhost:3306/productdb
spring.datasource.username=root
spring.datasource.password=yourpassword
spring.jpa.hibernate.ddl-auto=update
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

Payment.java

```
package com.example.paymentservice.entity;

import jakarta.persistence.*;
import lombok.Data;

@Entity
@Data
public class Payment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String orderId;
    private double amount;
    private String paymentMethod;
    private String status;
}
```

PaymentRepository.java

```
package com.example.paymentservice.repository;

import com.example.paymentservice.entity.Payment;
import org.springframework.data.jpa.repository.JpaRepository;
public interface PaymentRepository extends JpaRepository<Payment, Long> {}
```

PaymentController.java

```
package com.example.paymentservice.controller;

import com.example.paymentservice.entity.Payment;
import com.example.paymentservice.repository.PaymentRepository;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/payments")
public class PaymentController {

    private final PaymentRepository repo;

    public PaymentController(PaymentRepository repo) {
        this.repo = repo;
    }

    @PostMapping
    public Payment makePayment(@RequestBody Payment payment) {
        payment.setStatus("SUCCESS");
        return repo.save(payment);
    }
}
```

payment-service application.properties

```
spring.application.name=payment-service
server.port=8083
spring.datasource.url=jdbc:mysql://localhost:3306/paymentdb
spring.datasource.username=root
spring.datasource.password=yourpassword
spring.jpa.hibernate.ddl-auto=update
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

Order.java

```
package com.example.orderservice.entity;

import jakarta.persistence.*;
import lombok.Data;

@Entity
@Data
public class Order {
    @Id
    private String id;
    private String productId;
    private int quantity;
    private double totalAmount;
    private String status;
}
```

OrderRepository.java

```
package com.example.orderservice.repository;

import com.example.orderservice.entity.Order;
import org.springframework.data.jpa.repository.JpaRepository;

public interface OrderRepository extends JpaRepository<Order, String> {}
```

OrderController.java

```
package com.example.orderservice.controller;

import com.example.orderservice.entity.Order;
import com.example.orderservice.repository.OrderRepository;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;
import java.util.UUID;

@RestController
@RequestMapping("/orders")
public class OrderController {

    private final OrderRepository repo;
    private final RestTemplate restTemplate;

    public OrderController(OrderRepository repo, RestTemplate restTemplate) {
        this.repo = repo;
        this.restTemplate = restTemplate;
    }

    @PostMapping
    public Order placeOrder(@RequestBody Order order) {
        var product = restTemplate.getForObject(
                "http://product-service/products/" + order.getProductId(),
                ProductDTO.class);

        if (product != null && product.getStock() >= order.getQuantity()) {
            order.setId("O" + UUID.randomUUID().toString().substring(0, 5));
            order.setTotalAmount(product.getPrice() * order.getQuantity());
            order.setStatus("PENDING_PAYMENT");
            repo.save(order);

            PaymentDTO payment = new PaymentDTO(order.getId(), order.getTotalAmount(), "Credit Card"
            var paymentResponse = restTemplate.postForObject(
                    "http://payment-service/payments", payment, PaymentDTO.class);

            if (paymentResponse != null && "SUCCESS".equals(paymentResponse.getStatus())) {
                order.setStatus("CONFIRMED");
                restTemplate.put("http://product-service/products/" + order.getProductId() + "/reduc
            }
            return repo.save(order);
        }
        throw new RuntimeException("Product not available");
    }
}
```

ProductDTO.java

```
package com.example.orderservice.controller;

import lombok.Data;

@Data
public class ProductDTO {
    private String id;
    private String name;
    private double price;
    private int stock;
}
```

PaymentDTO.java

```
package com.example.orderservice.controller;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
```

```
@AllArgsConstructor
@NoArgsConstructor
public class PaymentDTO {
    private String orderId;
    private double amount;
    private String paymentMethod;
    private String status;
}
```

RestTemplateConfig.java

```
package com.example.orderservice.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

@Configuration
public class RestTemplateConfig {
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

order-service application.properties

```
spring.application.name=order-service
server.port=8082
spring.datasource.url=jdbc:mysql://localhost:3306/orderdb
spring.datasource.username=root
spring.datasource.password=yourpassword
spring.jpa.hibernate.ddl-auto=update
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```