ADVANN - Payment Service (Razorpay Integration)

Complete Code Documentation (Updated with pom.xml Dependencies)

1. Project Overview

This document contains the complete Payment Service code implementation for Advann microservices project. It includes Razorpay integration in Test Mode, APIs to create Razorpay order, verify payment signature, database storage for payment records, and the complete Maven pom.xml dependencies.

2. Folder Structure

```
payment-service
■■■ src/main/java/com/advann/payment_service
■   ■■■ PaymentServiceApplication.java
■   ■■■ config
■   ■   ■■■ RazorpayConfig.java
■   ■■■ controller
■   ■   ■■■ PaymentController.java
■   ■■■ dto
■   ■   ■■■ PaymentOrderRequestDto.java
■   ■   ■■■ PaymentOrderResponseDto.java
■   ■   ■■■ PaymentVerifyRequestDto.java
■   ■■■ entity
■   ■   ■■■ Payment.java
■   ■■■ enums
■   ■   ■■■ PaymentStatus.java
■   ■■■ exception
■   ■   ■■■ ResourceNotFoundException.java
■   ■   ■■■ GlobalExceptionHandler.java
■   ■■■ payload
■   ■   ■■■ ApiResponse.java
■   ■■■ repository
■   ■   ■■■ PaymentRepository.java
■   ■■■ service
■   ■   ■■■ services
■   ■       ■■■ PaymentService.java
■   ■■■ serviceImpl
■       ■■■ PaymentServiceImpl.java
■■■ src/main/resources
■   ■■■ application.yml
■   ■■■ templates
■       ■■■ razorpay-test.html
■■■ pom.xml
```

3. application.yml

```
server:
  port: 8085

spring:
  application:
    name: payment-service

  datasource:
    url: jdbc:postgresql://localhost:5432/advann_payment_db
    username: postgres
    password: root

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true

eureka:
```

```yaml
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/

razorpay:
  key-id: ${RAZORPAY_KEY_ID}
  key-secret: ${RAZORPAY_KEY_SECRET}

management:
  endpoints:
    web:
      exposure:
        include: health,info

  endpoint:
    health:
      show-details: always
```

4. pom.xml (Complete Dependencies)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsc

    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.3.8</version>
        <relativePath/>
    </parent>

    <groupId>com.advann</groupId>
    <artifactId>payment-service</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>payment-service</name>
    <description>Payment Service for Advann Microservices</description>

    <properties>
        <java.version>17</java.version>
        <spring-cloud.version>2023.0.5</spring-cloud.version>
    </properties>

    <dependencies>

        <!-- Spring Boot Web -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- Spring Data JPA -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <!-- Validation -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>

        <!-- Eureka Client -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>

        <!-- OpenFeign Client -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>

        <!-- PostgreSQL Driver -->
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>

        <!-- Lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.34</version>
            <optional>true</optional>
        </dependency>

        <!-- Swagger OpenAPI -->
        <dependency>
            <groupId>org.springdoc</groupId>
            <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
            <version>2.5.0</version>
        </dependency>

        <!-- Spring Boot Actuator -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>

        <!-- Razorpay Java SDK -->
        <dependency>
            <groupId>com.razorpay</groupId>
            <artifactId>razorpay-java</artifactId>
            <version>1.4.6</version>
        </dependency>

        <!-- Spring Boot Test -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <!-- Spring Cloud Dependency Management -->
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>

            <!-- Maven Compiler Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <annotationProcessorPaths>
                        <path>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                            <version>1.18.34</version>
                        </path>
                    </annotationProcessorPaths>
                </configuration>
            </plugin>

            <!-- Spring Boot Maven Plugin -->
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
```

```
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>

        </plugins>
    </build>

</project>
```

5. Payment Entity

```
package com.advann.payment_service.entity;

import com.advann.payment_service.enums.PaymentStatus;
import jakarta.persistence.*;
import lombok.*;

import java.time.LocalDateTime;

@Entity
@Table(name = "payments")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Payment {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private Long orderId;
    private String razorpayOrderId;
    private String razorpayPaymentId;
    private String razorpaySignature;
    private Double amount;

    @Enumerated(EnumType.STRING)
    private PaymentStatus paymentStatus;

    private LocalDateTime createdAt;
}
```

6. PaymentStatus Enum

```
package com.advann.payment_service.enums;

public enum PaymentStatus {
    CREATED,
    PAID,
    FAILED
}
```

7. DTO Classes

```
package com.advann.payment_service.dto;

import lombok.*;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class PaymentOrderRequestDto {
    private Long orderId;
    private Double amount;
}

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class PaymentOrderResponseDto {
    private Long orderId;
    private String razorpayOrderId;
    private Double amount;
    private String currency;
}

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class PaymentVerifyRequestDto {
    private Long orderId;
    private String razorpayOrderId;
    private String razorpayPaymentId;
    private String razorpaySignature;
}
```

8. ApiResponse Wrapper

```
package com.advann.payment_service.payload;

import lombok.*;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ApiResponse<T> {
    private boolean success;
    private String message;
    private T data;
}
```

9. Repository

```java
package com.advann.payment_service.repository;

import com.advann.payment_service.entity.Payment;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface PaymentRepository extends JpaRepository<Payment, Long> {
    Optional<Payment> findByRazorpayOrderId(String razorpayOrderId);
}
```

10. RazorpayConfig

```java
package com.advann.payment_service.config;

import com.razorpay.RazorpayClient;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class RazorpayConfig {

    @Value("${razorpay.key-id}")
    private String keyId;

    @Value("${razorpay.key-secret}")
    private String keySecret;

    @Bean
    public RazorpayClient razorpayClient() throws Exception {
        return new RazorpayClient(keyId, keySecret);
    }
}
```

11. Service Interface

```
package com.advann.payment_service.service.services;

import com.advann.payment_service.dto.PaymentOrderRequestDto;
import com.advann.payment_service.dto.PaymentOrderResponseDto;
import com.advann.payment_service.dto.PaymentVerifyRequestDto;

public interface PaymentService {

    PaymentOrderResponseDto createPaymentOrder(PaymentOrderRequestDto requestDto);

    String verifyPayment(PaymentVerifyRequestDto requestDto);
}
```

12. Service Implementation

```
package com.advann.payment_service.service.serviceImpl;

import com.advann.payment_service.dto.PaymentOrderRequestDto;
import com.advann.payment_service.dto.PaymentOrderResponseDto;
import com.advann.payment_service.dto.PaymentVerifyRequestDto;
import com.advann.payment_service.entity.Payment;
import com.advann.payment_service.enums.PaymentStatus;
import com.advann.payment_service.exception.ResourceNotFoundException;
import com.advann.payment_service.repository.PaymentRepository;
import com.advann.payment_service.service.services.PaymentService;
import com.razorpay.Order;
import com.razorpay.RazorpayClient;
import com.razorpay.Utils;
import lombok.RequiredArgsConstructor;
import org.json.JSONObject;
import org.springframework.stereotype.Service;

import java.time.LocalDateTime;

@Service
@RequiredArgsConstructor
public class PaymentServiceImpl implements PaymentService {

    private final RazorpayClient razorpayClient;
    private final PaymentRepository paymentRepository;

    @Override
    public PaymentOrderResponseDto createPaymentOrder(PaymentOrderRequestDto requestDto) {
        try {
            JSONObject orderRequest = new JSONObject();
            orderRequest.put("amount", requestDto.getAmount() * 100); // paise
            orderRequest.put("currency", "INR");
            orderRequest.put("receipt", "receipt_" + requestDto.getOrderId());

            Order razorpayOrder = razorpayClient.orders.create(orderRequest);

            Payment payment = Payment.builder()
                    .orderId(requestDto.getOrderId())
                    .razorpayOrderId(razorpayOrder.get("id"))
                    .amount(requestDto.getAmount())
                    .paymentStatus(PaymentStatus.CREATED)
                    .createdAt(LocalDateTime.now())
                    .build();

            paymentRepository.save(payment);

            return PaymentOrderResponseDto.builder()
                    .orderId(requestDto.getOrderId())
                    .razorpayOrderId(razorpayOrder.get("id"))
                    .amount(requestDto.getAmount())
                    .currency("INR")
                    .build();
```

```java
        } catch (Exception e) {
            throw new RuntimeException("Error while creating Razorpay order: " + e.getMessage());
        }
    }

    @Override
    public String verifyPayment(PaymentVerifyRequestDto requestDto) {
        try {
            Payment payment = paymentRepository.findByRazorpayOrderId(requestDto.getRazorpayOrderId())
                    .orElseThrow(() -> new ResourceNotFoundException("Payment not found"));

            JSONObject options = new JSONObject();
            options.put("razorpay_order_id", requestDto.getRazorpayOrderId());
            options.put("razorpay_payment_id", requestDto.getRazorpayPaymentId());
            options.put("razorpay_signature", requestDto.getRazorpaySignature());

            boolean isValid = Utils.verifyPaymentSignature(options, razorpayClient.getKeySecret());

            if (isValid) {
                payment.setRazorpayPaymentId(requestDto.getRazorpayPaymentId());
                payment.setRazorpaySignature(requestDto.getRazorpaySignature());
                payment.setPaymentStatus(PaymentStatus.PAID);
                paymentRepository.save(payment);

                return "Payment Verified Successfully";
            } else {
                payment.setPaymentStatus(PaymentStatus.FAILED);
                paymentRepository.save(payment);

                return "Payment Verification Failed";
            }

        } catch (Exception e) {
            throw new RuntimeException("Error while verifying payment: " + e.getMessage());
        }
    }
}
```

13. Controller

```
package com.advann.payment_service.controller;

import com.advann.payment_service.dto.PaymentOrderRequestDto;
import com.advann.payment_service.dto.PaymentOrderResponseDto;
import com.advann.payment_service.dto.PaymentVerifyRequestDto;
import com.advann.payment_service.payload.ApiResponse;
import com.advann.payment_service.service.services.PaymentService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/payments")
@RequiredArgsConstructor
@CrossOrigin(origins = "*")
public class PaymentController {

    private final PaymentService paymentService;

    @PostMapping("/create-order")
    public ResponseEntity<ApiResponse<PaymentOrderResponseDto>> createOrder(
            @RequestBody PaymentOrderRequestDto requestDto
    ) {
        PaymentOrderResponseDto response = paymentService.createPaymentOrder(requestDto);

        return ResponseEntity.ok(
                ApiResponse.<PaymentOrderResponseDto>builder()
                        .success(true)
                        .message("Razorpay order created successfully")
                        .data(response)
                        .build()
        );
    }

    @PostMapping("/verify")
    public ResponseEntity<ApiResponse<String>> verifyPayment(
            @RequestBody PaymentVerifyRequestDto requestDto
    ) {
        String response = paymentService.verifyPayment(requestDto);

        return ResponseEntity.ok(
                ApiResponse.<String>builder()
                        .success(true)
                        .message(response)
                        .data(response)
                        .build()
        );
    }
}
```

14. Exception Handling

```
package com.advann.payment_service.exception;

public class ResourceNotFoundException extends RuntimeException {
    public ResourceNotFoundException(String message) {
        super(message);
    }
}

package com.advann.payment_service.exception;

import com.advann.payment_service.payload.ApiResponse;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestControllerAdvice
```

```java
public class GlobalExceptionHandler {

    @ExceptionHandler(ResourceNotFoundException.class)
    public ResponseEntity<ApiResponse<Object>> handleResourceNotFound(ResourceNotFoundException ex) {
        return new ResponseEntity<>(
                ApiResponse.builder()
                        .success(false)
                        .message(ex.getMessage())
                        .data(null)
                        .build(),
                HttpStatus.NOT_FOUND
        );
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<ApiResponse<Object>> handleGeneric(Exception ex) {
        return new ResponseEntity<>(
                ApiResponse.builder()
                        .success(false)
                        .message("Something went wrong: " + ex.getMessage())
                        .data(null)
                        .build(),
                HttpStatus.INTERNAL_SERVER_ERROR
        );
    }
}
```