**Day6\_Assignment(Reactive Spring)**

**Problem:**

Day 6: Reactive Spring - Real-time Claim Status Updates

Task 1: Introduce Spring WebFlux for handling real-time claim status updates using reactive streams.

Task 2: Configure R2DBC for reactive database connectivity to update claim status dynamically.

Task 3: Implement WebSocket communication for real-time interaction between the client and the server."

**Solution:**

**Pom.xml**

<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 http://maven.apache.org/POM/4.0.0">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>claimstatus</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<java.version>17</java.version>

<spring-boot.version>3.1.0</spring-boot.version>

<r2dbc.version>1.1.4</r2dbc.version>

<r2dbc-postgresql.version>1.1.8</r2dbc-postgresql.version>

<spring-webflux.version>6.1.0</spring-webflux.version>

<spring-messaging.version>6.1.0</spring-messaging.version>

<spring-data-r2dbc.version>1.8.0</spring-data-r2dbc.version>

</properties>

<dependencies>

<dependency>

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

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

<version>${spring-boot.version}</version>

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-data-r2dbc</artifactId>

<version>${spring-boot.version}</version>

</dependency>

<dependency>

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

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

<version>${spring-boot.version}</version>

</dependency>

<dependency>

<groupId>io.r2dbc</groupId>

<artifactId>r2dbc-postgresql</artifactId>

<version>${r2dbc-postgresql.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-r2dbc</artifactId>

<version>${spring-data-r2dbc.version}</version>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**ClaimStatusApplication class**

package com.example.claimstatus;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ClaimStatusApplication {

public static void main(String[] args) {

SpringApplication.run(ClaimStatusApplication.class, args);

}

}

**Claim class**

package com.example.claimstatus.model;

import org.springframework.data.annotation.Id;

import org.springframework.data.relational.core.mapping.Table;

@Table("claims")

public class Claim {

@Id

private Long id;

private String status;

// Getters and setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

**ClaimRepository Interface**

package com.example.claimstatus.repository;

import com.example.claimstatus.model.Claim;

import org.springframework.data.repository.reactive.ReactiveCrudRepository;

import reactor.core.publisher.Flux;

public interface ClaimRepository extends ReactiveCrudRepository<Claim, Long> {

Flux<Claim> findByStatus(String status);

}

**ClaimService class**

package com.example.claimstatus.service;

import com.example.claimstatus.model.Claim;

import com.example.claimstatus.repository.ClaimRepository;

import org.springframework.stereotype.Service;

import reactor.core.publisher.Flux;

import reactor.core.publisher.Mono;

@Service

public class ClaimService {

private final ClaimRepository claimRepository;

public ClaimService(ClaimRepository claimRepository) {

this.claimRepository = claimRepository;

}

public Flux<Claim> getAllClaims() {

return claimRepository.findAll();

}

public Mono<Claim> updateClaimStatus(Long id, String status) {

return claimRepository.findById(id)

.flatMap(claim -> {

claim.setStatus(status);

return claimRepository.save(claim);

});

}

public Flux<Claim> getClaimsByStatus(String status) {

return claimRepository.findByStatus(status);

}

}

**ClaimStatusController class**

package com.example.claimstatus.controller;

import com.example.claimstatus.model.Claim;

import com.example.claimstatus.service.ClaimService;

import org.springframework.messaging.handler.annotation.MessageMapping;

import org.springframework.messaging.handler.annotation.SendTo;

import org.springframework.stereotype.Controller;

import reactor.core.publisher.Flux;

@Controller

public class ClaimStatusController {

private final ClaimService claimService;

public ClaimStatusController(ClaimService claimService) {

this.claimService = claimService;

}

@MessageMapping("/update-status")

@SendTo("/topic/status")

public Flux<Claim> updateStatus(Long claimId, String status) {

return claimService.updateClaimStatus(claimId, status).flux();

}

}

**R2dbcConfig class**

package com.example.claimstatus.config;

import io.r2dbc.spi.ConnectionFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.r2dbc.config.AbstractR2dbcConfiguration;

@Configuration

public class R2dbcConfig extends AbstractR2dbcConfiguration {

@Override

@Bean

public ConnectionFactory connectionFactory() {

return ConnectionFactories.get("r2dbc:postgresql://localhost:5432/claims\_db");

}

}

**WebSocketConfig class**

package com.example.claimstatus.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.messaging.simp.config.MessageBrokerRegistry;

import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;

import org.springframework.web.socket.config.annotation.StompEndpointRegistry;

import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;

@Configuration

@EnableWebSocketMessageBroker

public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

@Override

public void configureMessageBroker(MessageBrokerRegistry config) {

config.enableSimpleBroker("/topic");

config.setApplicationDestinationPrefixes("/app");

}

@Override

public void registerStompEndpoints(StompEndpointRegistry registry) {

registry.addEndpoint("/websocket").setAllowedOrigins("\*").withSockJS();

}

}

**application.properties**

# R2DBC configuration

spring.r2dbc.url=r2dbc:postgresql://localhost:5432/claims\_db

spring.r2dbc.username=your-username

spring.r2dbc.password=your-password

# WebFlux configuration

spring.webflux.base-path=/

# R2DBC repositories

spring.data.r2dbc.repositories.enabled=true