**Exercise 9: Employee Management System - Customizing Data Source Configuration**

**Business Scenario:**

Customize your data source configuration and manage multiple data sources.

**Instructions:**

1. **Spring Boot Auto-Configuration:**
   * Leverage Spring Boot auto-configuration for data sources.
2. **Externalizing Configuration:**
   * Externalize configuration with application.properties.
   * Manage multiple data sources within your application.

**application.properties🡪**

spring.application.name=EmployeeProjectManagement

# Primary Data Source (Employee DB)

spring.datasource.primary.jdbc-url=jdbc:h2:mem:employee\_db;

spring.datasource.primary.username=sa

spring.datasource.primary.password=password

# Secondary Data Source (Department DB)

spring.datasource.secondary.jdbc-url=jdbc:h2:mem:department\_db

spring.datasource.secondary.username=sa

spring.datasource.secondary.password=password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.transaction.jta.platform=org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform

spring.h2.console.enabled=true

spring.jpa.generate-ddl=true

spring.jpa.properties.hibernate.show\_sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect

server.port=8187

**DataSourceConfig.java 🡪**

package com.code.employee.config;

import javax.sql.DataSource;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.jdbc.DataSourceBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Primary;

@Configuration

public class DataSourceConfig {

    @Primary

    @Bean(name = "primaryDataSource")

    @ConfigurationProperties(prefix = "spring.datasource.primary")

    public DataSource primaryDataSource() {

        return DataSourceBuilder.create().build();

    }

    @Bean(name = "secondaryDataSource")

    @ConfigurationProperties(prefix = "spring.datasource.secondary")

    public DataSource secondaryDataSource() {

        return DataSourceBuilder.create().build();

    }

}

**JpaConfig.java 🡪**

package com.code.employee.config;

import java.util.HashMap;

import java.util.Map;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.jdbc.DataSourceBuilder;

import org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Primary;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

import org.springframework.transaction.PlatformTransactionManager;

import org.springframework.transaction.annotation.EnableTransactionManagement;

import jakarta.persistence.EntityManagerFactory;

@Configuration

@EnableTransactionManagement

@EnableJpaRepositories(

        basePackages = "com.code.employee.repositories", // Primary Repository package

        entityManagerFactoryRef = "primaryEntityManagerFactory",

        transactionManagerRef = "primaryTransactionManager"

)

public class JpaConfig {

@Primary

@Bean(name = "primaryEntityManagerFactory")

public LocalContainerEntityManagerFactoryBean primaryEntityManagerFactory(

        EntityManagerFactoryBuilder builder, @Qualifier("primaryDataSource") DataSource dataSource) {

    return builder

            .dataSource(dataSource)

            .packages("com.code.employee.entity")  // Package with Employee entities

            .persistenceUnit("primary")

           .properties(hibernateProperties("org.hibernate.dialect.H2Dialect"))

            .build();

}

private Map<String, Object> hibernateProperties(String dialect) {

    Map<String, Object> props = new HashMap<>();

    props.put("hibernate.dialect", dialect);

    return props;

}

@Primary

@Bean(name = "primaryTransactionManager")

public PlatformTransactionManager primaryTransactionManager(

        @Qualifier("primaryEntityManagerFactory") EntityManagerFactory entityManagerFactory) {

    return new JpaTransactionManager(entityManagerFactory);

}

}

**SecondaryJpaConfig.java🡪**

package com.code.employee.config;

import java.util.HashMap;

import java.util.Map;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

import org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

import org.springframework.transaction.PlatformTransactionManager;

import jakarta.persistence.EntityManagerFactory;

@Configuration

@EnableAutoConfiguration

@EnableJpaRepositories(

        basePackages = "com.code.department.repositories",  // Secondary Repository package

        entityManagerFactoryRef = "secondaryEntityManagerFactory",

        transactionManagerRef = "secondaryTransactionManager"

    )

public class SecondaryJpaConfig {

    @Bean(name = "secondaryEntityManagerFactory")

    public LocalContainerEntityManagerFactoryBean secondaryEntityManagerFactory(

            EntityManagerFactoryBuilder builder, @Qualifier("secondaryDataSource") DataSource dataSource) {

        return builder

                .dataSource(dataSource)

                .packages("com.code.department.entity")  // Package with Department entities

                .persistenceUnit("secondary")

               // .properties(hibernateProperties("org.hibernate.dialect.MySQL8Dialect"))

                .build();

    }

    private Map<String, Object> hibernateProperties(String dialect) {

        Map<String, Object> props = new HashMap<>();

        props.put("hibernate.dialect", dialect);

        return props;

    }

    @Bean(name = "secondaryTransactionManager")

    public PlatformTransactionManager secondaryTransactionManager(

            @Qualifier("secondaryEntityManagerFactory") EntityManagerFactory entityManagerFactory) {

        return new JpaTransactionManager(entityManagerFactory);

    }

}