**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.

**Solution**

**application.properties**

# Primary Data Source (e.g., Employee Database)

spring.datasource.primary.url=jdbc:mysql://localhost:3306/employee\_db

spring.datasource.primary.username=root

spring.datasource.primary.password=yourpassword

spring.datasource.primary.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.primary.hikari.pool-name=primaryPool

# Secondary Data Source (e.g., Department Database)

spring.datasource.secondary.url=jdbc:mysql://localhost:3306/department\_db

spring.datasource.secondary.username=root

spring.datasource.secondary.password=yourpassword

spring.datasource.secondary.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.secondary.hikari.pool-name=secondaryPool

**PrimaryDataSourceConfig.java**

package com.employee.employeemanagement.config;

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

import org.springframework.boot.autoconfigure.jdbc.DataSourceProperties;

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

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 javax.sql.DataSource;

@Configuration

@EnableJpaRepositories(

basePackages = "com.employee.employeemanagement.repository.primary",

entityManagerFactoryRef = "primaryEntityManagerFactory",

transactionManagerRef = "primaryTransactionManager"

)

public class PrimaryDataSourceConfig {

@Bean

@ConfigurationProperties("spring.datasource.primary")

public DataSourceProperties primaryDataSourceProperties() {

return new DataSourceProperties();

}

@Bean

public DataSource primaryDataSource() {

return primaryDataSourceProperties().initializeDataSourceBuilder().build();

}

@Bean

public LocalContainerEntityManagerFactoryBean primaryEntityManagerFactory(

EntityManagerFactoryBuilder builder) {

return builder

.dataSource(primaryDataSource())

.packages("com.employee.employeemanagement.entity.primary")

.build();

}

@Bean

public PlatformTransactionManager primaryTransactionManager(

@Qualifier("primaryEntityManagerFactory") LocalContainerEntityManagerFactoryBean primaryEntityManagerFactory) {

return new JpaTransactionManager(primaryEntityManagerFactory.getObject());

}

}

**SecondaryDataSourceConfig.java**

package com.employee.employeemanagement.config;

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

import org.springframework.boot.autoconfigure.jdbc.DataSourceProperties;

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

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 javax.sql.DataSource;

@Configuration

@EnableJpaRepositories(

basePackages = "com.employee.employeemanagement.repository.secondary",

entityManagerFactoryRef = "secondaryEntityManagerFactory",

transactionManagerRef = "secondaryTransactionManager"

)

public class SecondaryDataSourceConfig {

@Bean

@ConfigurationProperties("spring.datasource.secondary")

public DataSourceProperties secondaryDataSourceProperties() {

return new DataSourceProperties();

}

@Bean

public DataSource secondaryDataSource() {

return secondaryDataSourceProperties().initializeDataSourceBuilder().build();

}

@Bean

public LocalContainerEntityManagerFactoryBean secondaryEntityManagerFactory(

EntityManagerFactoryBuilder builder) {

return builder

.dataSource(secondaryDataSource())

.packages("com.employee.employeemanagement.entity.secondary")

.build();

}

@Bean

public PlatformTransactionManager secondaryTransactionManager(

@Qualifier("secondaryEntityManagerFactory") LocalContainerEntityManagerFactoryBean secondaryEntityManagerFactory) {

return new JpaTransactionManager(secondaryEntityManagerFactory.getObject());

}

}

**Main.java**

package com.employee.employeemanagement;

import com.employee.employeemanagement.repository.primary.EmployeeRepository;

import com.employee.employeemanagement.repository.secondary.DepartmentRepository;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class Main implements CommandLineRunner {

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private DepartmentRepository departmentRepository;

@Override

public void run(String... args) throws Exception {

// Work with primary data source (Employee)

employeeRepository.findAll().forEach(System.out::println);

// Work with secondary data source (Department)

departmentRepository.findAll().forEach(System.out::println);

}

}