**WEEK 3**

**EXERCISE 9**

**Employee Management System - Customizing Data Source Configuration**

**1. Spring Boot Auto-Configuration:**

**a. Leveraging Spring Boot Auto-Configuration for Data Sources**

* **Purpose:** Spring Boot’s auto-configuration simplifies setting up a data source by automatically configuring the DataSource bean based on the properties defined in application.properties.

**Basic Data Source Configuration Example:**

**application.properties**

# Basic H2 Data Source Configuration

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

**2. Externalizing Configuration:**

**a. Externalizing Configuration with application.properties**

* **Purpose:** By externalizing configuration, you can easily switch between different environments (e.g., development, testing, production) without changing the application code.

**Example: Configuring a MySQL Data Source in application.properties:**

# MySQL Data Source Configuration

spring.datasource.url=jdbc:mysql://localhost:3306/employeedb

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect

spring.jpa.hibernate.ddl-auto=update

**b. Managing Multiple Data Sources within Your Application**

* **Purpose:** In some scenarios, you may need to connect to multiple databases. This can be managed by defining multiple DataSource beans in your application.

**Example: Multiple Data Source Configuration**

**1.Define the Primary Data Source:**

# Primary Data Source (H2)

spring.datasource.primary.url=jdbc:h2:mem:primarydb

spring.datasource.primary.username=sa

spring.datasource.primary.password=password

spring.datasource.primary.driverClassName=org.h2.Driver

**2.Define a Secondary Data Source:**

# Secondary Data Source (MySQL)

spring.datasource.secondary.url=jdbc:mysql://localhost:3306/secondarydb

spring.datasource.secondary.username=root

spring.datasource.secondary.password=yourpassword

spring.datasource.secondary.driverClassName=com.mysql.cj.jdbc.Driver

1. **Java Configuration for Multiple Data Sources:**

Create a configuration class to define multiple DataSource beans.

package com.example.employeemanagementsystem.config;

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

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

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Primary;

import javax.sql.DataSource;

@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();

}

}

1. **Using Multiple Data Sources:**

You can now use @Qualifier to inject the specific DataSource you need in your repositories or services.

package com.example.employeemanagementsystem.repository;

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

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

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Repository;

@Repository

public class SomeRepository {

private final JdbcTemplate primaryJdbcTemplate;

private final JdbcTemplate secondaryJdbcTemplate;

@Autowired

public SomeRepository(@Qualifier("primaryDataSource") DataSource primaryDataSource,

@Qualifier("secondaryDataSource") DataSource secondaryDataSource) {

this.primaryJdbcTemplate = new JdbcTemplate(primaryDataSource);

this.secondaryJdbcTemplate = new JdbcTemplate(secondaryDataSource);

}

// Methods to interact with both databases

}