**application.properties**:

# Primary Data Source (MySQL)

spring.datasource.primary.url=jdbc:mysql://localhost:3306/primarydb

spring.datasource.primary.username=root

spring.datasource.primary.password=password

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

# Secondary Data Source (PostgreSQL)

spring.datasource.secondary.url=jdbc:postgresql://localhost:5432/secondarydb

spring.datasource.secondary.username=postgres

spring.datasource.secondary.password=password

spring.datasource.secondary.driver-class-name=org.postgresql.Driver

**2. Externalizing Configuration**

**Step 1: Create Configuration Classes for Data Sources**

**Primary Data Source Configuration**:

package com.example.EmployeeManagementSystem.config;

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

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.env.Environment;

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

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

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;

import org.springframework.transaction.PlatformTransactionManager;

import javax.sql.DataSource;

@Configuration

@EnableJpaRepositories(

basePackages = "com.example.repository.primary",

entityManagerFactoryRef = "primaryEntityManagerFactory",

transactionManagerRef = "primaryTransactionManager"

)

public class PrimaryDataSourceConfig {

@Bean(name = "primaryDataSource")

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

public DataSource dataSource() {

return new DriverManagerDataSource();

}

@Bean(name = "primaryEntityManagerFactory")

public LocalContainerEntityManagerFactoryBean entityManagerFactory(

@Qualifier("primaryDataSource") DataSource dataSource) {

LocalContainerEntityManagerFactoryBean factory = new LocalContainerEntityManagerFactoryBean();

factory.setDataSource(dataSource);

factory.setPackagesToScan("com.example.entity.primary");

factory.setJpaVendorAdapter(new HibernateJpaVendorAdapter());

return factory;

}

@Bean(name = "primaryTransactionManager")

public PlatformTransactionManager transactionManager(

@Qualifier("primaryEntityManagerFactory") LocalContainerEntityManagerFactoryBean entityManagerFactory) {

return new JpaTransactionManager(entityManagerFactory.getObject());

}

}

**Secondary Data Source Configuration**:

package com.example.EmployeeManagementSystem.config;

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

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;

import org.springframework.transaction.PlatformTransactionManager;

import javax.sql.DataSource;

@Configuration

@EnableJpaRepositories(

basePackages = "com.example.repository.secondary",

entityManagerFactoryRef = "secondaryEntityManagerFactory",

transactionManagerRef = "secondaryTransactionManager"

)

public class SecondaryDataSourceConfig {

@Bean(name = "secondaryDataSource")

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

public DataSource dataSource() {

return new DriverManagerDataSource();

}

@Bean(name = "secondaryEntityManagerFactory")

public LocalContainerEntityManagerFactoryBean entityManagerFactory(

@Qualifier("secondaryDataSource") DataSource dataSource) {

LocalContainerEntityManagerFactoryBean factory = new LocalContainerEntityManagerFactoryBean();

factory.setDataSource(dataSource);

factory.setPackagesToScan("com.example.entity.secondary");

factory.setJpaVendorAdapter(new HibernateJpaVendorAdapter());

return factory;

}

@Bean(name = "secondaryTransactionManager")

public PlatformTransactionManager transactionManager(

@Qualifier("secondaryEntityManagerFactory") LocalContainerEntityManagerFactoryBean entityManagerFactory) {

return new JpaTransactionManager(entityManagerFactory.getObject());

}

}

**Step 2: Define Custom Configuration Properties**

package com.example.EmployeeManagementSystem.config;

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

import org.springframework.context.annotation.Configuration;

@Configuration

@ConfigurationProperties(prefix = "spring.datasource")

public class DataSourceProperties {

private Primary primary;

private Secondary secondary;

public static class Primary {

private String url;

private String username;

private String password;

private String driverClassName;

// getters and setters

}

public static class Secondary {

private String url;

private String username;

private String password;

private String driverClassName;

// getters and setters

}

// getters and setters

}