**Using multiple datasources with Spring Boot and Spring Data**

[**Spring Boot**](https://projects.spring.io/spring-boot/) with [**Spring Data**](http://projects.spring.io/spring-data/) makes it easy to access a database through so called Repositories. But what if you want to access **multiple databases** maybe even with different Database Management Systems?

Luckily Spring provides a way of doing this.

I provided an example project with two PostgreSQL datasources on GitHub:

<https://github.com/jahe/spring-boot-multiple-datasources>

1. Add an additional datasource configuration to your application.properties

|  |
| --- |
| **# Oracle DB - "foo"** spring.datasource.url=jdbc:oracle:thin:@//db-server-foo:1521/FOO spring.datasource.username=fooadmin spring.datasource.password=foo123spring.datasource.driver-class-name=oracle.jdbc.OracleDriver  **# PostgreSQL DB - "bar"** bar.datasource.url=jdbc:postgresql://db-server-bar:5432/bar bar.datasource.username=baradmin bar.datasource.password=bar123bar.datasource.driver-class-name=org.postgresql.Driver |

**2. Set the SQL Dialect to “default” in your application.properties to let Spring autodetect the different SQL Dialects of each datasource**

|  |
| --- |
| spring.jpa.database=**default** |

**3. Create a Java Package for each datasource with two nested Packages “domain” and “repo”**

|  |
| --- |
| **src/main/java - com.foobar - foo  - domain  - repo - bar  - domain  - repo** |

**4. Create a Configuration Class for the Oracle database “foo” named “FooDbConfig.java”**

|  |
| --- |
| package com.foobar;  @Configuration @EnableTransactionManagement @EnableJpaRepositories( entityManagerFactoryRef = "entityManagerFactory", basePackages = { "com.foobar.foo.repo" } ) public class FooDbConfig {  @Primary @Bean(name = "dataSource") @ConfigurationProperties(prefix = "spring.datasource") public DataSource dataSource() { return DataSourceBuilder.create().build(); }  @Primary @Bean(name = "entityManagerFactory") public LocalContainerEntityManagerFactoryBean entityManagerFactory( EntityManagerFactoryBuilder builder, @Qualifier("dataSource") DataSource dataSource ) { return builder .dataSource(dataSource) .packages("com.foobar.foo.domain") .persistenceUnit("foo") .build(); }  @Primary @Bean(name = "transactionManager") public PlatformTransactionManager transactionManager( @Qualifier("entityManagerFactory") EntityManagerFactory  entityManagerFactory ) { return new JpaTransactionManager(entityManagerFactory); } } |