**origoni's Blog from Millky**

**millky.com/@origoni/post/1150?language=ko\_kr**

origoni의 스프링 블로그 입니다.

[Home](http://millky.com/@origoni/post/list)

* [스프링으로 블로그 만들기!](http://millky.com/@origoni/folder/30/post/list)
* [밀키 개발일지](http://millky.com/@origoni/folder/2/post/list)

양식의 맨 위

양식의 맨 아래

양식의 맨 위

양식의 맨 아래

**05/31**

**2015**

07:43:24

0, 0

In [프로그래밍](http://millky.com/@origoni/folder/4/post/list)

**SpringBoot + Spring Data JPA 에서 2개 이상의 DB 사용하기**

<http://millky.com/@origoni/post/1150>

이번에 프로젝트를 진행하는데. 두개의 서로 다른 DB에 접근을 해야 하는 이슈가 있었다.

스프링 부트의 아주 강력한(?ㅋ!) 기본 설정만 사용하다가 추가적인 작업을 하려니 바로 되지 않아서 따로 프로젝트를 만들고 이렇게 공유 해본다.

개발자는 역시 코드를 보면 된다.

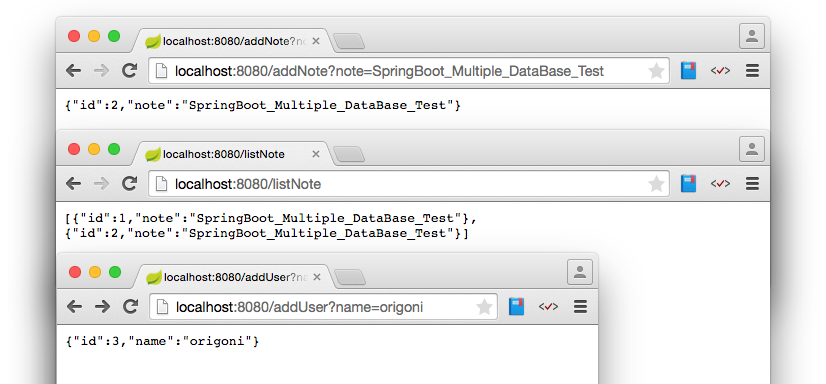
<https://github.com/origoni/MultiDatabase> 에 공개해 두었으니 참고 바란다.

참.. 아직 트랜젝션 부분은 고민하지 않았다.

**우선 프로젝트를 돌려보면~**

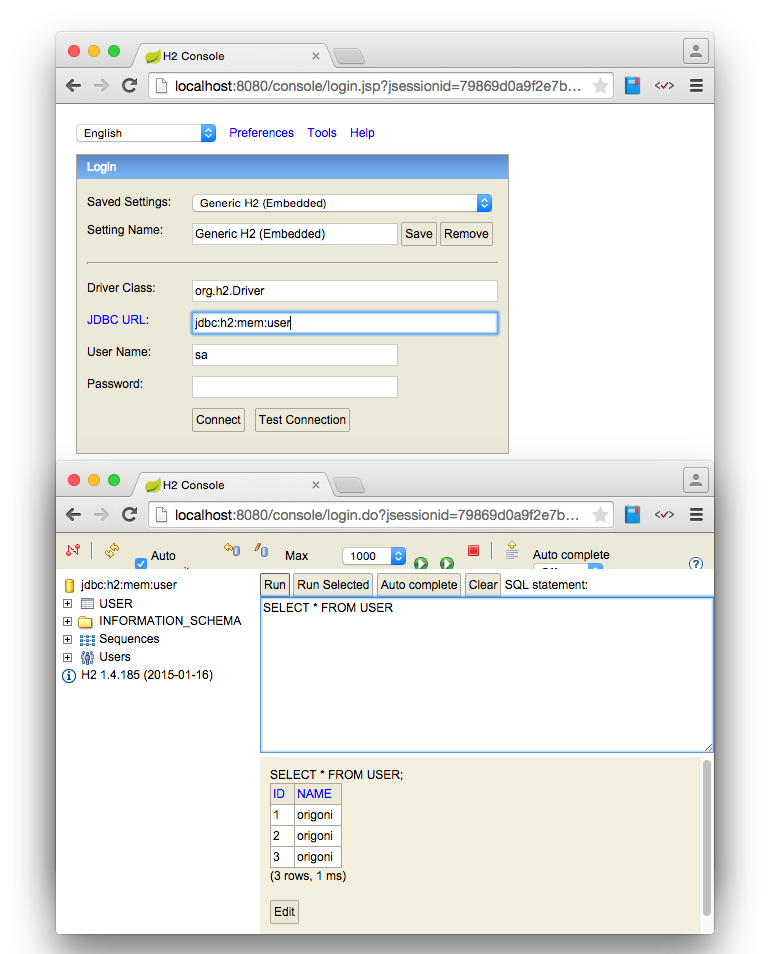
* http://localhost:8080/addUser?name=origoni
* http://localhost:8080/listUser
* http://localhost:8080/addNote?note=SpringBoot\_Multiple\_DataBase\_Test
* http://localhost:8080/listNote

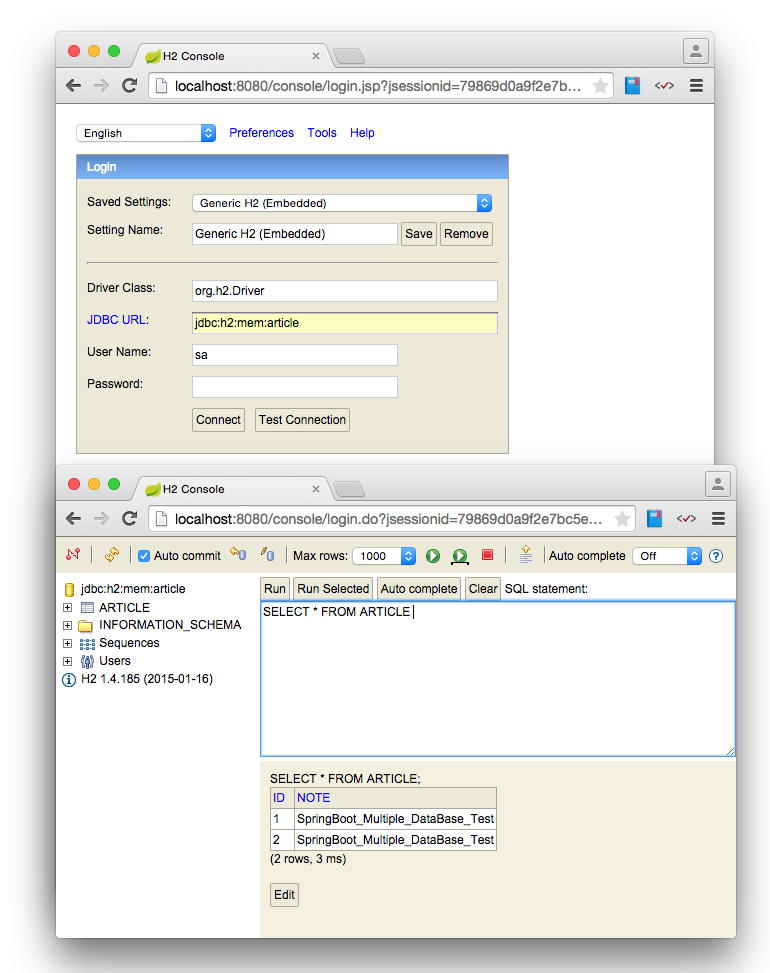
이렇게 데이터를 넣고 확인해 볼 수 있다.



그리고 H2 콘솔을 붙여 놓았다.

* http://localhost:8080/console
* JDBC URL: jdbc:h2:mem:article
* JDBC URL: jdbc:h2:mem:user





접속해보면 서로 다른 DB에 데이터가 잘 들어가 있는 것을 확인 할 수 있다.

**이제 핵심 파일을 보면..**

https://github.com/origoni/MultiDatabase/blob/v0.0.1-SNAPSHOT/src/main/java/com/millky/dev/database/multi/config/DataBaseConfig.java

import java.util.HashMap;

import java.util.Map;

import javax.sql.DataSource;

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

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

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

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

import org.springframework.transaction.PlatformTransactionManager;

@Configuration

public class DataBaseConfig {

private static final String DEFAULT\_NAMING\_STRATEGY

= "org.springframework.boot.orm.jpa.hibernate.SpringNamingStrategy";

@Bean

@Primary

@ConfigurationProperties(prefix = "datasource.article")

public DataSource articleDataSource() {

return DataSourceBuilder.create().build();

}

@Primary

@Bean(name = "entityManagerFactory")

public LocalContainerEntityManagerFactoryBean entityManagerFactory(

EntityManagerFactoryBuilder builder) {

Map<String, String> propertiesHashMap = new HashMap<>();

propertiesHashMap.put("hibernate.ejb.naming\_strategy",DEFAULT\_NAMING\_STRATEGY);

return builder.dataSource(articleDataSource())

.packages("com.millky.dev.database.multi.domain.article")

.properties(propertiesHashMap)

.build();

}

@Primary

@Bean(name = "transactionManager")

PlatformTransactionManager transactionManager(

EntityManagerFactoryBuilder builder) {

return new JpaTransactionManager(entityManagerFactory(builder).getObject());

}

@Configuration

@EnableJpaRepositories(

basePackages="com.millky.dev.database.multi.infra.article",

entityManagerFactoryRef = "entityManagerFactory",

transactionManagerRef = "transactionManager")

static class DbArticleJpaRepositoriesConfig {

}

@Bean

@ConfigurationProperties(prefix = "datasource.user")

public DataSource userDataSource() {

return DataSourceBuilder.create().build();

}

@Bean(name = "entityManagerFactoryUser")

public LocalContainerEntityManagerFactoryBean userEntityManagerFactory(

EntityManagerFactoryBuilder builder) {

return builder.dataSource(userDataSource())

.packages("com.millky.dev.database.multi.domain.user")

.build();

}

@Bean(name = "transactionManagerUser")

@Primary

PlatformTransactionManager userTransactionManagerMain(

EntityManagerFactoryBuilder builder) {

return new JpaTransactionManager(userEntityManagerFactory(builder).getObject());

}

@Configuration

@EnableJpaRepositories(

basePackages="com.millky.dev.database.multi.infra.user",

entityManagerFactoryRef = "entityManagerFactoryUser",

transactionManagerRef = "transactionManagerUser")

static class DbUserJpaRepositoriesConfig {

}

}

요렇게 되어있다.

**요점은**

1. 사용할 각 DB의 DataSource를 생성하고.
2. 각각의 EntityManger, TransactionManager를 EntityManagerFactory로 만든다.
3. 그리고 @EnableJpaRepositories를 설정해준다.

**여기서 entityManagerFactory의 .packages(...) 는 Entity의 위치이고.**

**@EnableJpaRepositories(basePackages="...) 는 Repository의 위치이다. (extends JpaRepository)**

**그리고 하나는 @Primary로 지정해준다.**

자세한 사항은 코드를 확인해보면 금방 이해가 갈 것이라 생각한다 ^^

<https://github.com/origoni/MultiDatabase/tree/v0.0.1-SNAPSHOT>