Spring Boot Multiple Databases Example - Mr.RAGHU

pom.xml

```
<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jpa</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>

	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-devtools</artifactId>
		<scope>runtime</scope>
		<optional>true</optional>
	</dependency>
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<scope>runtime</scope>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-test</artifactId>
		<scope>test</scope>
		<exclusions>
			<exclusion>
				<groupId>org.junit.vintage</groupId>
				<artifactId>junit-vintage-engine</artifactId>
			</exclusion>
		</exclusions>
	</dependency>
	<dependency>
		<groupId>org.projectlombok</groupId>
		<artifactId>lombok</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-configuration-processor</artifactId>
        <optional>true</optional>
    </dependency>

    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
```

1. application.properties

```
db1.datasource.jdbc-url=jdbc:postgresql://localhost:5432/db1
db1.datasource.username=postgres
db1.datasource.password=root
db1.datasource.driver-class-name=org.postgresql.Driver


db2.datasource.jdbc-url=jdbc:mysql://localhost:3306/db2
db2.datasource.username=root
db2.datasource.password=root
db2.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

2. Model classes

```
package in.nareshit.raghu.model.product;

import javax.persistence.Entity;
import javax.persistence.Id;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Product {

    @Id
    private int id;
```

```
	private String code;
	private String name;
}
```

```
package in.nareshit.raghu.model.customer;

import javax.persistence.Entity;
import javax.persistence.Id;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Customer {

	@Id
	private Integer id;
	private String email;
	private String cname;

}
```

3. Repository

```
package in.nareshit.raghu.repo.product;

import org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.model.product.Product;

public interface ProductRepository
extends JpaRepository<Product, Integer> {

}
```

```
package in.nareshit.raghu.repo.customer;

import org.springframework.data.jpa.repository.JpaRepository;
```

```
import in.nareshit.raghu.model.customer.Customer;

public interface CustomerRepository
extends JpaRepository<Customer, Integer> {

}
```

4. DataSource Config Files using Spring Java Configuration

```
package in.nareshit.raghu.config.product;

import java.util.HashMap;

import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder;
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;
import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(
                entityManagerFactoryRef = "db1EntityManagerFactory",
                transactionManagerRef = "db1TransactionManager",
                basePackages = "in.nareshit.raghu.repo.product"
                )
public class DbOneConfig {


        //DataSource
```

```
	@Primary
	@Bean
	@ConfigurationProperties(prefix="db1.datasource")
	public DataSource db1DataSource() {
		return DataSourceBuilder.create().build();
	}

	//EntityManagerFactory
	@Primary
	@Bean
	public LocalContainerEntityManagerFactoryBean db1EntityManagerFactory(
			EntityManagerFactoryBuilder emfb)
	{
		HashMap<String, Object> properties = new HashMap<>();
		properties.put("hibernate.hbm2ddl.auto", "create");
		properties.put("hibernate.dialect", "org.hibernate.dialect.PostgreSQL10Dialect");
		return emfb
				.dataSource(db1DataSource())
				.packages("in.nareshit.raghu.model.product")
				.properties(properties)
				.persistenceUnit("db1")
				.build();
	}

	//TransactionManager
	@Primary
	@Bean
	public PlatformTransactionManager db1TransactionManager(
			@Qualifier("db1EntityManagerFactory") EntityManagerFactory
		entityManagerFactory)
	{
		return new JpaTransactionManager(entityManagerFactory);
	}
}
```

```
package in.nareshit.raghu.config.customer;

import java.util.HashMap;

import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;
```

```
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.jdbc.DataSourceBuilder;
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 org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(
                entityManagerFactoryRef = "db2EntityManagerFactory",
                transactionManagerRef = "db2TransactionManager",
                basePackages = "in.nareshit.raghu.repo.customer"
                )
public class DbTwoConfig {


        //DataSource
        @Bean
        @ConfigurationProperties(prefix="db2.datasource")
        public DataSource db2DataSource() {
                return DataSourceBuilder.create().build();
        }

        //EntityManagerFactory
        @Bean
        public LocalContainerEntityManagerFactoryBean db2EntityManagerFactory(
                        EntityManagerFactoryBuilder emfb)
        {
                HashMap<String, Object> properties = new HashMap<>();
                properties.put("hibernate.hbm2ddl.auto", "create");
                properties.put("hibernate.dialect", "org.hibernate.dialect.MySQL5Dialect");
                return emfb
                                .dataSource(db2DataSource())
                                .packages("in.nareshit.raghu.model.customer")
                                .properties(properties)
                                .persistenceUnit("db2")
                                .build();
```

```
	}

	//TransactionManager
	@Bean
	public PlatformTransactionManager db2TransactionManager(
			@Qualifier("db2EntityManagerFactory") EntityManagerFactory
		entityManagerFactory)
	{
		return new JpaTransactionManager(entityManagerFactory);
	}

}
```

5. Starter class with Runnercode

```
package in.nareshit.raghu;

import java.util.Arrays;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import in.nareshit.raghu.model.customer.Customer;
import in.nareshit.raghu.model.product.Product;
import in.nareshit.raghu.repo.customer.CustomerRepository;
import in.nareshit.raghu.repo.product.ProductRepository;

@SpringBootApplication
public class SpringBoot2MultipleDatabasesApplication implements CommandLineRunner{

	public static void main(String[] args) {
		SpringApplication.run(SpringBoot2MultipleDatabasesApplication.class, args);
	}

	@Autowired
	private ProductRepository productRepo;

	@Autowired
	private CustomerRepository customerRepo;
```

```
	@Override
	public void run(String... args) throws Exception {
		productRepo.saveAll(
				Arrays.asList(
					new Product(101, "P-1", "PEN"),
					new Product(102, "P-2", "BOOK"),
					new Product(103, "P-3", "TEST")
					)
				);

		customerRepo.saveAll(
				Arrays.asList(
					new Customer(550, "sam@gmail.com", "sam"),
					new Customer(551, "ram@gmail.com", "ram"),
					new Customer(552, "khan@gmail.com", "khan")
					)
				);
	}

}
```

6. RestController

```
package in.nareshit.raghu.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import in.nareshit.raghu.model.customer.Customer;
import in.nareshit.raghu.model.product.Product;
import in.nareshit.raghu.repo.customer.CustomerRepository;
import in.nareshit.raghu.repo.product.ProductRepository;

@RestController
public class MultiDataRestController {

	@Autowired
	private ProductRepository productRepo;
```

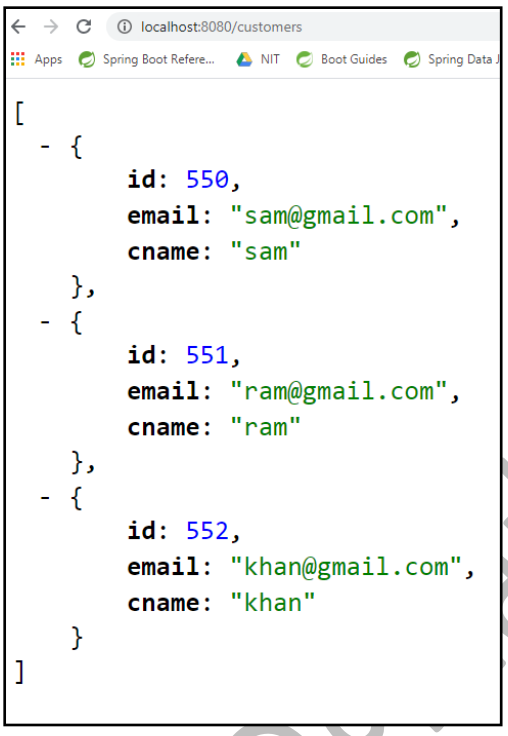
```
	@Autowired
	private CustomerRepository customerRepo;

	@GetMapping("/products")
	public List<Product> getProducts(){
		return productRepo.findAll();
	}

	@GetMapping("/customers")
	public List<Customer> getCustomer(){
		return customerRepo.findAll();
	}
}
```

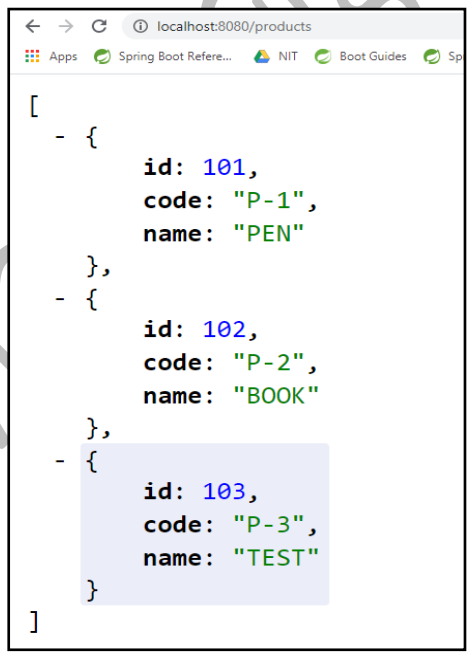
Postgres Database Commands:

```
cmd>psql -U postgres
create database db1
\c db1
\dt
select * from product;
```

MySQL Database Commands:

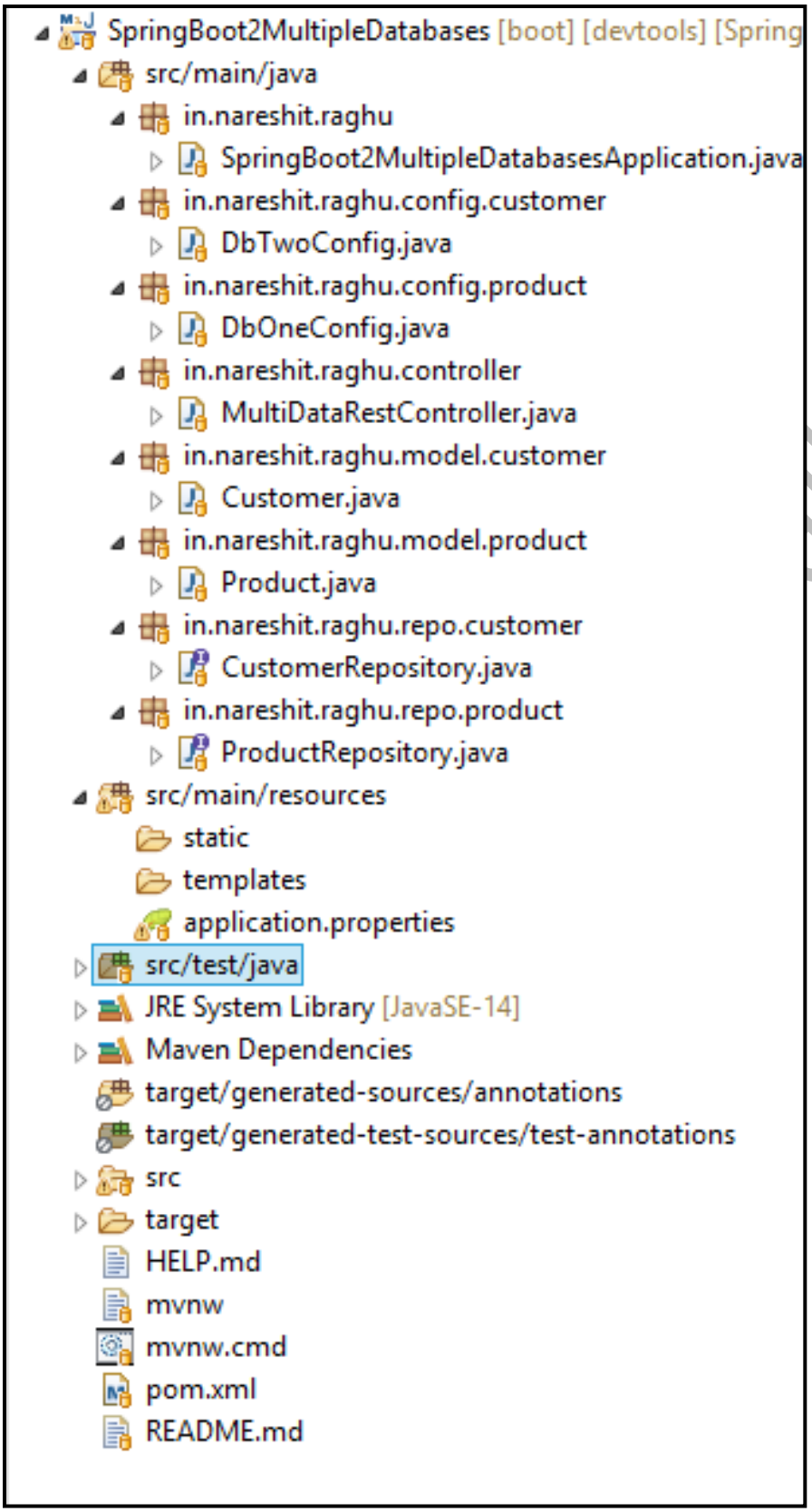
```
create database db2;
use db2;
show tables;
select * from customer;
```





FB: https://www.facebook.com/groups/thejavatemple/

Email: javabyraghu@gmail.com


SpringBoot2MultipleDatabases [boot] [devtools] [Spring
src/main/java
in.nareshit.raghu
SpringBoot2MultipleDatabasesApplication.java
in.nareshit.raghu.config.customer
DbTwoConfig.java
in.nareshit.raghu.config.product
DbOneConfig.java
in.nareshit.raghu.controller
MultiDataRestController.java
in.nareshit.raghu.model.customer
Customer.java
in.nareshit.raghu.model.product
Product.java
in.nareshit.raghu.repo.customer
CustomerRepository.java
in.nareshit.raghu.repo.product
ProductRepository.java
src/main/resources
static
templates
application.properties
src/test/java
JRE System Library [JavaSE-14]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotations
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
README.md