CONNEXION DE SPRING BOOT À PLUSIEURS BASES DE DONNÉES MONGO

La connexion d'une application spring boot à plusieurs bases de données mongodb se fait en plusieurs étapes. Nous présenterons chacune des étapes dans la suite du document.

Etape 1: Configuration des bases de données dans le fichier application.yml

Désactiver la configuration par défaut utilisée par spring boot en ajoutant les lignes suivantes dans application.yml

```
spring:
  autoconfigure:
    exclude:
org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguratio
n
```

Etape 2: Configuration des bases de données dans le fichier application.yml

Spécifier les paramètres de connexion aux deux bases de données dans le fichier config de l'application.

A la fin des étapes 1 et 2 votre application.yml devrait ressembler à ceci:

```yaml
spring:
  autoconfigure:
    exclude:
org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguration
  data:
    mongodb:
      primary:
        host: localhost
        port: 27017
        database: first
      secondary:
        host: localhost
        port: 27017
        database: second
```

Etape 3: Créer une classe pour définir les propriétés primary et secondary que nous avons ajoutées

Créer une classe MultipleMongoProperties qui se présentera comme suit:

```
package com.itns.test.config;

import org.springframework.boot.autoconfigure.mongo.MongoProperties;
import
org.springframework.boot.context.properties.ConfigurationProperties;

import lombok.Data;

@Data
@ConfigurationProperties(prefix = "spring.data.mongodb")
public class MultipleMongoProperties {
    private MongoProperties primary = new MongoProperties();
    private MongoProperties secondary = new MongoProperties();
}
```

<u>Etape 4:</u> Créer une classe MultipleMongoConfig pour définir les templates et le fonctionnement des propriétés primary et secondary avec le contenu suivant

```
package com.itns.test.config;

import com.mongodb.client.MongoClients;
import org.springframework.boot.autoconfigure.mongo.MongoProperties;
import
org.springframework.boot.context.properties.EnableConfigurationProperti
es;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.data.mongodb.MongoDatabaseFactory;
import org.springframework.data.mongodb.core.MongoTemplate;

import lombok.RequiredArgsConstructor;
import
org.springframework.data.mongodb.core.SimpleMongoClientDatabaseFactory;

@Configuration
@RequiredArgsConstructor
@EnableConfigurationProperties(MultipleMongoProperties.class)
public class MultipleMongoConfig {
```

```
    private final MultipleMongoProperties mongoProperties;

    @Primary
    @Bean(name = "primaryMongoTemplate")
    public MongoTemplate primaryMongoTemplate() throws Exception {
        return new
MongoTemplate(primaryFactory(this.mongoProperties.getPrimary()));
    }

    @Bean(name = "secondaryMongoTemplate")
    public MongoTemplate secondaryMongoTemplate() throws Exception {
        return new
MongoTemplate(secondaryFactory(this.mongoProperties.getSecondary()));
    }

    @Bean
    @Primary
    public MongoDatabaseFactory primaryFactory(final MongoProperties
mongo) throws Exception {
        return new
SimpleMongoClientDatabaseFactory(MongoClients.create("mongodb://"+mongo
.getHost()+":"+mongo.getPort()), mongo.getDatabase());
    }

    @Bean
    public MongoDatabaseFactory secondaryFactory(final MongoProperties
mongo) throws Exception {
        return new
SimpleMongoClientDatabaseFactory(MongoClients.create("mongodb://"+mongo
.getHost()+":"+mongo.getPort()), mongo.getDatabase());
    }

}
```

Etape 4: Créer deux classes PrimaryMongoConfig et SecondaryMongoConfig pour configurer les repository qui seront utilisés pour joindre soit la première base(primary) soit la seconde (secondary)

```
package com.itns.test.config;

import org.springframework.context.annotation.Configuration;
```

```
import
org.springframework.data.mongodb.repository.config.EnableMongoRepositor
ies;

@Configuration
@EnableMongoRepositories(basePackages =
"com.itns.test.repositories.primary",
        mongoTemplateRef = "primaryMongoTemplate")
public class PrimaryMongoConfig {

}
```

```
package com.itns.test.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.data.mongodb.repository.config.EnableMongoRepositor
ies;

@Configuration
@EnableMongoRepositories(basePackages =
"com.itns.test.repositories.secondary",
        mongoTemplateRef = "secondaryMongoTemplate")
public class SecondaryMongoConfig {

}
```

Ici, les repositories situés dans le package com.itns.repositories.primary auront accès à la première base de données et ceux dans com.itns.repositories.secondary à la seconde base de données.

Après cette dernière configuration, la procédure est la même qu'à l'accoutumée, créer les modèles et placer leurs repositories dans l'un des deux packages selon que vous voulez qu'ils soient ajoutés dans la première ou la seconde base de données.