springboot(十一)：Spring boot中mongodb的使用

 2017/05/08

mongodb是最早热门非关系数据库的之一，使用也比较普遍，一般会用做离线数据分析来使用，放到内网的居多。由于很多公司使用了云服务，服务器默认都开放了外网地址，导致前一阵子大批 MongoDB 因配置漏洞被攻击，数据被删，引起了人们的注意，感兴趣的可以看看这篇文章：[场屠戮MongoDB的盛宴反思：超33000个数据库遭遇入侵勒索](http://www.freebuf.com/articles/database/125127.html)，同时也说明了很多公司生产中大量使用mongodb。

**mongodb简介**

MongoDB（来自于英文单词“Humongous”，中文含义为“庞大”）是可以应用于各种规模的企业、各个行业以及各类应用程序的开源数据库。基于分布式文件存储的数据库。由C++语言编写。旨在为WEB应用提供可扩展的高性能数据存储解决方案。MongoDB是一个高性能，开源，无模式的文档型数据库，是当前NoSql数据库中比较热门的一种。

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据结构非常松散，是类似json的bjson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

传统的关系数据库一般由数据库（database）、表（table）、记录（record）三个层次概念组成，MongoDB是由数据库（database）、集合（collection）、文档对象（document）三个层次组成。MongoDB对于关系型数据库里的表，但是集合中没有列、行和关系概念，这体现了模式自由的特点。

MongoDB中的一条记录就是一个文档，是一个数据结构，由字段和值对组成。MongoDB文档与JSON对象类似。字段的值有可能包括其它文档、数组以及文档数组。MongoDB支持OS X、Linux及Windows等操作系统，并提供了Python，PHP，Ruby，Java及C++语言的驱动程序，社区中也提供了对Erlang及.NET等平台的驱动程序。

MongoDB的适合对大量或者无固定格式的数据进行存储，比如：日志、缓存等。对事物支持较弱，不适用复杂的多文档（多表）的级联查询。文中演示mongodb版本为3.4。

**mongodb的增删改查**

Spring Boot对各种流行的数据源都进行了封装，当然也包括了mongodb,下面给大家介绍如何在spring boot中使用mongodb：

**1、pom包配置**

pom包里面添加spring-boot-starter-data-mongodb包引用

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

</dependencies>

**2、在application.properties中添加配置**

spring.data.mongodb.uri=mongodb://name:pass@localhost:27017/test

多个IP集群可以采用以下配置：

spring.data.mongodb.uri=mongodb://user:pwd@ip1:port1,ip2:port2/database

**2、创建数据实体**

**public** **class** **UserEntity** **implements** Serializable **{**

**private** **static** **final** **long** serialVersionUID **=** **-**3258839839160856613L**;**

**private** Long id**;**

**private** String userName**;**

**private** String passWord**;**

*//getter、setter省略*

**}**

**3、创建实体dao的增删改查操作**

dao层实现了UserEntity对象的增删改查

**@Component**

**public** **class** **UserDaoImpl** **implements** UserDao **{**

**@Autowired**

**private** MongoTemplate mongoTemplate**;**

*/\*\**

*\* 创建对象*

*\* @param user*

*\*/*

**@Override**

**public** **void** **saveUser(**UserEntity user**)** **{**

mongoTemplate**.**save**(**user**);**

**}**

*/\*\**

*\* 根据用户名查询对象*

*\* @param userName*

*\* @return*

*\*/*

**@Override**

**public** UserEntity **findUserByUserName(**String userName**)** **{**

Query query**=new** Query**(**Criteria**.**where**(**"userName"**).**is**(**userName**));**

UserEntity user **=** mongoTemplate**.**findOne**(**query **,** UserEntity**.**class**);**

**return** user**;**

**}**

*/\*\**

*\* 更新对象*

*\* @param user*

*\*/*

**@Override**

**public** **void** **updateUser(**UserEntity user**)** **{**

Query query**=new** Query**(**Criteria**.**where**(**"id"**).**is**(**user**.**getId**()));**

Update update**=** **new** Update**().**set**(**"userName"**,** user**.**getUserName**()).**set**(**"passWord"**,** user**.**getPassWord**());**

*//更新查询返回结果集的第一条*

mongoTemplate**.**updateFirst**(**query**,**update**,**UserEntity**.**class**);**

*//更新查询返回结果集的所有*

*// mongoTemplate.updateMulti(query,update,UserEntity.class);*

**}**

*/\*\**

*\* 删除对象*

*\* @param id*

*\*/*

**@Override**

**public** **void** **deleteUserById(**Long id**)** **{**

Query query**=new** Query**(**Criteria**.**where**(**"id"**).**is**(**id**));**

mongoTemplate**.**remove**(**query**,**UserEntity**.**class**);**

**}**

**}**

**4、开发对应的测试方法**

**@RunWith(**SpringRunner**.**class**)**

**@SpringBootTest**

**public** **class** **UserDaoTest** **{**

**@Autowired**

**private** UserDao userDao**;**

**@Test**

**public** **void** **testSaveUser()** **throws** Exception **{**

UserEntity user**=new** UserEntity**();**

user**.**setId**(**2l**);**

user**.**setUserName**(**"小明"**);**

user**.**setPassWord**(**"fffooo123"**);**

userDao**.**saveUser**(**user**);**

**}**

**@Test**

**public** **void** **findUserByUserName(){**

UserEntity user**=** userDao**.**findUserByUserName**(**"小明"**);**

System**.**out**.**println**(**"user is "**+**user**);**

**}**

**@Test**

**public** **void** **updateUser(){**

UserEntity user**=new** UserEntity**();**

user**.**setId**(**2l**);**

user**.**setUserName**(**"天空"**);**

user**.**setPassWord**(**"fffxxxx"**);**

userDao**.**updateUser**(**user**);**

**}**

**@Test**

**public** **void** **deleteUserById(){**

userDao**.**deleteUserById**(**1l**);**

**}**

**}**

**5、查看验证结果**

可以使用工具mongoVUE工具来连接后直接图形化展示查看，也可以登录服务器用命令来查看

1.登录mongos

bin/mongo -host localhost -port 20000

2、切换到test库

use test

3、查询userEntity集合数据

db.userEntity.find()

根据3查询的结果来观察测试用例的执行是否正确。

到此springboot对应mongodb的增删改查功能已经全部实现。

**多数据源mongodb的使用**

在多mongodb数据源的情况下，我们换种更优雅的方式来实现

**1、pom包配置**

添加lombok和spring-boot-autoconfigure包引用

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-autoconfigure</artifactId>

<version>RELEASE</version>

</dependency>

* Lombok - 是一个可以通过简单的注解形式来帮助我们简化消除一些必须有但显得很臃肿的Java代码的工具，通过使用对应的注解，可以在编译源码的时候生成对应的方法。简单试了以下这个工具还挺好玩的，加上注解我们就不用手动写 getter\setter、构建方式类似的代码了。
* spring-boot-autoconfigure - 就是spring boot的自动化配置

**2、配置文件使用YAML的形式添加两条数据源，如下：**

mongodb:

primary:

host: 192.168.9.60

port: 20000

database: test

secondary:

host: 192.168.9.60

port: 20000

database: test1

**3、配置两个库的数据源**

封装读取以mongodb开头的两个配置文件

**@Data**

**@ConfigurationProperties(**prefix **=** "mongodb"**)**

**public** **class** **MultipleMongoProperties** **{**

**private** MongoProperties primary **=** **new** MongoProperties**();**

**private** MongoProperties secondary **=** **new** MongoProperties**();**

**}**

配置不同包路径下使用不同的数据源

第一个库的封装

**@Configuration**

**@EnableMongoRepositories(**basePackages **=** "com.neo.model.repository.primary"**,**

mongoTemplateRef **=** PrimaryMongoConfig**.**MONGO\_TEMPLATE**)**

**public** **class** **PrimaryMongoConfig** **{**

**protected** **static** **final** String MONGO\_TEMPLATE **=** "primaryMongoTemplate"**;**

**}**

第二个库的封装

**@Configuration**

**@EnableMongoRepositories(**basePackages **=** "com.neo.model.repository.secondary"**,**

mongoTemplateRef **=** SecondaryMongoConfig**.**MONGO\_TEMPLATE**)**

**public** **class** **SecondaryMongoConfig** **{**

**protected** **static** **final** String MONGO\_TEMPLATE **=** "secondaryMongoTemplate"**;**

**}**

读取对应的配置信息并且构造对应的MongoTemplate

**@Configuration**

**public** **class** **MultipleMongoConfig** **{**

**@Autowired**

**private** MultipleMongoProperties mongoProperties**;**

**@Primary**

**@Bean(**name **=** PrimaryMongoConfig**.**MONGO\_TEMPLATE**)**

**public** MongoTemplate **primaryMongoTemplate()** **throws** Exception **{**

**return** **new** **MongoTemplate(**primaryFactory**(this.**mongoProperties**.**getPrimary**()));**

**}**

**@Bean**

**@Qualifier(**SecondaryMongoConfig**.**MONGO\_TEMPLATE**)**

**public** MongoTemplate **secondaryMongoTemplate()** **throws** Exception **{**

**return** **new** **MongoTemplate(**secondaryFactory**(this.**mongoProperties**.**getSecondary**()));**

**}**

**@Bean**

**@Primary**

**public** MongoDbFactory **primaryFactory(**MongoProperties mongo**)** **throws** Exception **{**

**return** **new** **SimpleMongoDbFactory(new** MongoClient**(**mongo**.**getHost**(),** mongo**.**getPort**()),**

mongo**.**getDatabase**());**

**}**

**@Bean**

**public** MongoDbFactory **secondaryFactory(**MongoProperties mongo**)** **throws** Exception **{**

**return** **new** **SimpleMongoDbFactory(new** MongoClient**(**mongo**.**getHost**(),** mongo**.**getPort**()),**

mongo**.**getDatabase**());**

**}**

**}**

两个库的配置信息已经完成。

**4、创建两个库分别对应的对象和Repository**

借助lombok来构建对象

**@Data**

**@AllArgsConstructor**

**@NoArgsConstructor**

**@Document(**collection **=** "first\_mongo"**)**

**public** **class** **PrimaryMongoObject** **{**

**@Id**

**private** String id**;**

**private** String value**;**

**@Override**

**public** String **toString()** **{**

**return** "PrimaryMongoObject{" **+** "id='" **+** id **+** '\'' **+** ", value='" **+** value **+** '\''

**+** '}'**;**

**}**

**}**

对应的Repository

**public** **interface** **PrimaryRepository** **extends** MongoRepository**<**PrimaryMongoObject**,** String**>** **{**

**}**

继承了 MongoRepository 会默认实现很多基本的增删改查，省了很多自己写dao层的代码

Secondary和上面的代码类似就不贴出来了

**5、最后测试**

**@RunWith(**SpringRunner**.**class**)**

**@SpringBootTest**

**public** **class** **MuliDatabaseTest** **{**

**@Autowired**

**private** PrimaryRepository primaryRepository**;**

**@Autowired**

**private** SecondaryRepository secondaryRepository**;**

**@Test**

**public** **void** **TestSave()** **{**

System**.**out**.**println**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**);**

System**.**out**.**println**(**"测试开始"**);**

System**.**out**.**println**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**);**

**this.**primaryRepository

**.**save**(new** PrimaryMongoObject**(null,** "第一个库的对象"**));**

**this.**secondaryRepository

**.**save**(new** SecondaryMongoObject**(null,** "第二个库的对象"**));**

List**<**PrimaryMongoObject**>** primaries **=** **this.**primaryRepository**.**findAll**();**

**for** **(**PrimaryMongoObject primary **:** primaries**)** **{**

System**.**out**.**println**(**primary**.**toString**());**

**}**

List**<**SecondaryMongoObject**>** secondaries **=** **this.**secondaryRepository**.**findAll**();**

**for** **(**SecondaryMongoObject secondary **:** secondaries**)** **{**

System**.**out**.**println**(**secondary**.**toString**());**

**}**

System**.**out**.**println**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**);**

System**.**out**.**println**(**"测试完成"**);**

System**.**out**.**println**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**);**

**}**

**}**

到此，mongodb多数据源的使用已经完成。

[**示例代码-github**](https://github.com/ityouknow/spring-boot-examples)

[**示例代码-码云**](https://gitee.com/ityouknow/spring-boot-examples)