SpringBoot 2.2.5 集成 MongoDB 4.2.3 实践

作者：cloudlang
原文链接：https://ld246.com/article/1584694811681
来源网站：链滴

工具准备

- IDEA
- FinalShell SSH 工具
- 3T for **MongoDB**

下载地址：：http://zhang.love/tools.html

参考资料

1. MongoDB 基础入门到高级进阶 https://ke.qq.com/course/475530?taid=4927628864078218
2. 阿里云 CentOS7 安装 MongoDB https://www.jianshu.com/p/a0f5d62a5737

CentOS7 安装 MongoDB 4.2 命令

```
vi /etc/yum.repos.d/mongodb-org-4.2.repo

[mogodb-org]

name=MongoDB Repository

baseurl=http://mirrors.aliyun.com/mongodb/yum/redhat/7Server/mongodb-org/4.2/x86_64/

gpgcheck=0

enabled=1


yum install -y mongodb-org
```

启动 MongoDB 测试

```
systemctl start mongod

# 查看状态
systemctl status mongod

# 默认端口 测试链接
mongo
show dbs
use test
db.user.insertOne({name:'st',age:'23'})
db.user.findOne({name:'st'})
```

配置用户

```
vi /etc/mongod.conf

# network interfaces
```

```
net:
  port: 8090 #自定义端口
  bindIp: 0.0.0.0  #开启外网访问

# 开启用户授权
security:
  authorization: enabled


systemctl restart mongod
mongo --port 你的端口

 use admin
 #创建管理员账户 不能进行关闭数据库等操作。
 db.createUser({ user: "admin", pwd: "你的密码", roles: [{ role: "userAdminAnyDatabase", db: "
dmin" }] })

 db.auth("admin", "你的密码")

 # 创建你数据库和用户
 use test
 db.createUser({ user: "test_user", pwd: "test_pwd", roles: [{ role: "dbOwner", db: "test" }] })
```

application.yml

```
#配置mongodb数据库名称，服务ip，端口号和用户密码
spring:
  data:
    mongodb:
      database: test
      host: 127.0.0.1
      username: test_user
      password: test_pwd
      port: 8090
```

pom.xml

```
<dependency>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-data-mongodb</artifactId>
    <!-- 如果日志使用log4j2，需要排除，-->
          <exclusions>
             <exclusion>
                <artifactId>logback-classic</artifactId>
                <groupId>ch.qos.logback</groupId>
             </exclusion>
             <exclusion>
                <artifactId>spring-boot-starter-logging</artifactId>
                <groupId>org.springframework.boot</groupId>
             </exclusion>
          </exclusions>
```

```
        </dependency>
```

代码参考

```
import com.alibaba.fastjson.JSON;
import com.mongodb.client.result.DeleteResult;
import com.mongodb.client.result.UpdateResult;

import org.bson.Document;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.annotation.Id;
import org.springframework.data.domain.Sort;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.data.mongodb.core.query.Update;
import org.springframework.stereotype.Repository;
import org.springframework.util.Assert;

import java.lang.reflect.ParameterizedType;
import java.util.List;
import java.util.Objects;
import java.util.Optional;

/**
 @Author:MrHuang
 @Date: 2019/9/4 16:19
 @DESC: TODO
 @VERSION: 1.0
 **/
@Repository
public class MongoDao<K,V> {

    @Autowired
    private MongoTemplate mongoTemplate;

    /**
     * 新增
     * @param v
     * @return
     */
    public V insert(V v) {
        return mongoTemplate.insert(v);
    }

    /**
     * 查询
     * @param id
     * @return
     */
    public V findById(K id) {
        Class<V> vClass = (Class<V>)((ParameterizedType)getClass().getGenericSuperclass()).ge
ActualTypeArguments()[1];
        return mongoTemplate.findById(id, vClass);
```

```
    }

    /**
     * 根据ID批量查询
     * @param ids
     * @return
     */
    public List<V> findByIds(List<K> ids) {
        Class<V> vClass = (Class<V>)((ParameterizedType)getClass().getGenericSuperclass()).ge
ActualTypeArguments()[1];
        String idFieldName = ReflectUtil.getFieldName(vClass, Id.class);
        Assert.notNull(idFieldName, "@Id not find");
        return mongoTemplate.find(Query.query(Criteria.where(idFieldName).in(ids)), vClass);
    }

    /**
     * 强制更新
     * @param v
     * @return
     */
    public UpdateResult updateById(V v) {
        Update update = Update.fromDocument(Document.parse(JSON.toJSONString(v)));
        ReflectUtil.FieldNameValue id = ReflectUtil.getFieldNameValue(v, Id.class);
        Assert.notNull(id, "@Id not find");
        return mongoTemplate.updateFirst(Query.query(Criteria.where(id.getFieldName()).is(id.g
tFieldValue())), update, v.getClass());
    }

    /**
     * 乐观锁更新
     * @param v
     * @return
     */
    public UpdateResult updateByIdWithVersion(V v) {
        ReflectUtil.FieldNameValue id = ReflectUtil.getFieldNameValue(v, Id.class);
        Assert.notNull(id, "@Id not find");
        ReflectUtil.FieldNameValue version = ReflectUtil.getFieldNameValue(v, MongoVersion.cl
ss);
        Assert.notNull(version, "@MongoVersion not find");
        Object fieldValue = Optional.ofNullable(version.getFieldValue()).orElse("0");
        Criteria criteria = Criteria.where(id.getFieldName()).is(id.getFieldValue()).and(version.getF
eldName()).is(fieldValue);
        // 版本号+1
        ReflectUtil.setFieldValue(v, version.getFieldName(), (Integer.parseInt(fieldValue.toString())
+ 1) + "");
        Update update = Update.fromDocument(Document.parse(JSON.toJSONString(v)));
        update.set("_class", v.getClass().getName());
        return mongoTemplate.updateFirst(Query.query(criteria), update, v.getClass());
    }

    /**
     * 物理删除
     * @param id
     * @return
```

```
     */
    public DeleteResult deleteById(K id) {
        Class<V> vClass = (Class<V>)((ParameterizedType)getClass().getGenericSuperclass()).ge
ActualTypeArguments()[1];
        String idFieldName = ReflectUtil.getFieldName(vClass, Id.class);
        Assert.notNull(idFieldName, "@Id not find");
        return mongoTemplate.remove(Query.query(Criteria.where(idFieldName).is(id)), vClass);
    }


    /**
     * 根据查询条件查找列表
     * @param criteria
     * @return
     */
    public List<V> find(Criteria criteria) {
        Class<V> vClass = (Class<V>)((ParameterizedType)getClass().getGenericSuperclass()).ge
ActualTypeArguments()[1];
        return mongoTemplate.find(Query.query(criteria), vClass);
    }

    /**
     * 根据查询条件查找列表
     * @param criteria
     * @return
     */
    public List<V> find(Criteria criteria, Sort sort, Long skip, Integer limit) {
        Class<V> vClass = (Class<V>)((ParameterizedType)getClass().getGenericSuperclass()).ge
ActualTypeArguments()[1];
        Query query = Query.query(criteria);
        if (Objects.nonNull(sort)) {
            query.with(sort);
        }
        if (Objects.nonNull(skip)) {
            query.skip(skip);
        }
        if (Objects.nonNull(limit)) {
            query.limit(limit);
        }
        return mongoTemplate.find(query, vClass);
    }

    /**
     * 根据查询条件查找条数
     * @param criteria
     * @return
     */
    public long count(Criteria criteria) {
        Class<V> vClass = (Class<V>)((ParameterizedType)getClass().getGenericSuperclass()).ge
ActualTypeArguments()[1];
        return mongoTemplate.count(Query.query(criteria), vClass);
    }
```

```
    /**
     * 根据查询条件分页查询
     * @param criteria 查询条件
     * @param sort 排序条件
     * @param pageNow 查找的页数
     * @param pageSize 每页显示大小
     */
//    public RTPaging<V> paging(Criteria criteria, Sort sort, long pageNow, int pageSize) {
//        long totalRecord = this.count(criteria);
//        long totalPage = RTPaging.getTotalPage(totalRecord, pageSize);
//        long skip = RTPaging.getSkip(pageNow, pageSize);
//        List<V> recond = this.find(criteria, sort, skip, pageSize);
//        return new RTPaging<V>().setPageNow(pageNow).setPageSize(pageSize)
//                .setTotalRecord(totalRecord).setTotalPage(totalPage)
//                .setRecord(recond);
//    }


    /**
     * 更新第一条匹配到的
     * @param criteria
     * @param update
     * @return
     */
    public UpdateResult updateFirst(Criteria criteria, Update update) {
        Class<V> vClass = (Class<V>)((ParameterizedType)getClass().getGenericSuperclass()).ge
ActualTypeArguments()[1];
        return mongoTemplate.updateFirst(Query.query(criteria), update, vClass);
    }

    /**
     * 更新所有匹配到的
     * @param criteria
     * @param update
     * @return
     */
    public UpdateResult updateMulti(Criteria criteria, Update update) {
        Class<V> vClass = (Class<V>)((ParameterizedType)getClass().getGenericSuperclass()).ge
ActualTypeArguments()[1];
        return mongoTemplate.updateMulti(Query.query(criteria), update, vClass);
    }
}
```