**引入依赖**

修改pom.xml

<dependency>

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

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

</dependency>

<dependency>

<groupId>joda-time</groupId>

<artifactId>joda-time</artifactId>

<version>2.10.1</version>

</dependency>

**添加配置**

在application.properties文件添加配置

spring.data.mongodb.uri=mongodb://localhost:27017/learn

**基于MongoTemplate 开发CRUD**

**添加实体**

**实现**

**常用方法**

mongoTemplate.findAll(Try.class): 查询Try文档的全部数据

mongoTemplate.findById(, Try.class): 查询Try文档id为id的数据

mongoTemplate.find(query, Try.class);: 根据query内的查询条件查询

mongoTemplate.upsert(query, update, Try.class): 修改

mongoTemplate.remove(query, Try.class): 删除

mongoTemplate.insert(Try): 新增

**Query对象**

创建一个query对象（用来封装所有条件对象)，再创建一个criteria对象（用来构建条件）

精准条件：criteria.and(“key”).is(“条件”)

模糊条件：criteria.and(“key”).regex(“条件”)

封装条件：query.addCriteria(criteria)

大于（创建新的criteria）：Criteria gt = Criteria.where(“key”).gt（“条件”）

小于（创建新的criteria）：Criteria lt = Criteria.where(“key”).lt（“条件”）

Query.addCriteria(new Criteria().andOperator(gt,lt));

一个query中只能有一个andOperator()。其参数也可以是Criteria数组。

排序 ：query.with（new Sort(Sort.Direction.ASC, “age”). and(new Sort(Sort.Direction.DESC, “date”)))

**添加测试类**

package com.xuda;

import com.mongodb.client.result.DeleteResult;

import com.mongodb.client.result.UpdateResult;

import com.xuda.entity.User;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.data.domain.Example;

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 java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.regex.Pattern;

@SpringBootTest

class MongodbApplicationTests {

//注入MongoTemplate

@Autowired

private MongoTemplate mongoTemplate;

//添加操作

@Test

void contextLoads() {

User user = new User();

user.setAge(20);

user.setName("test");

user.setEmail("4932200@qq.com");

User user1 = mongoTemplate.insert(user);

System.out.println(user1);

}

//查询所有

@Test

public void findUser() {

List<User> userList = mongoTemplate.findAll(User.class);

System.out.println(userList);

}

//id查询

@Test

public void getById() {

User user = mongoTemplate.findById("62501d8d5e835d2841ebcfc7",User.class);

System.out.println(user);

}

//条件查询

@Test

public void findUserList() {

Query query = new Query(Criteria.where("name").is("test").and("age").is(20));

List<User> list = mongoTemplate.find(query,User.class);

System.out.println(list);

}

//模糊查询

@Test

public void findUsersLikeName() {

String name = "est";

String regex = String.format("%s%s%s", "^.\*", name, ".\*$");//采用正则表达式进行匹配

Pattern pattern = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);

Query query = new Query(Criteria.where("name").regex(pattern));

List<User> userList = mongoTemplate.find(query, User.class);

System.out.println(userList);

}

//分页查询

@Test

public void findUsersPage() {

String name = "est";

int pageNo = 1; //当前页

int pageSize = 10; //每页的大小

Query query = new Query(); //条件构建部分

String regex = String.format("%s%s%s", "^.\*", name, ".\*$");

Pattern pattern = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);

query.addCriteria(Criteria.where("name").regex(pattern));

int totalCount = (int) mongoTemplate.count(query, User.class); //查询记录数

//其中的skip表示跳过的记录数，当前页为1则跳过0条，为2则跳过10条，（也就是跳过第一页的10条数据）

List<User> userList = mongoTemplate.find(query.skip((pageNo - 1) \* pageSize).limit(pageSize), User.class); //分页查询

Map<String, Object> pageMap = new HashMap<>();

pageMap.put("list", userList);

pageMap.put("totalCount",totalCount);

System.out.println(pageMap);

}

//修改

@Test

public void updateUser() {

User user = mongoTemplate.findById("62501d8d5e835d2841ebcfc7", User.class); // 先查询数据

user.setName("test\_1");

user.setAge(25);

user.setEmail("493220990@qq.com"); //设置修改的值

Query query = new Query(Criteria.where("\_id").is(user.getId()));//查到数据

Update update = new Update();

update.set("name", user.getName()); //设置mongodb的字段值

update.set("age", user.getAge());

update.set("email", user.getEmail());

UpdateResult result = mongoTemplate.upsert(query, update, User.class);

long count = result.getModifiedCount(); //影响行数

System.out.println(count);

}

//删除操作

@Test

public void delete() {

Query query =

new Query(Criteria.where("\_id").is("62501d8d5e835d2841ebcfc7")); //查询到数据

DeleteResult result = mongoTemplate.remove(query, User.class); //删除方法

long count = result.getDeletedCount();

System.out.println(count);

}

}

**基于测试类完成的mongodb的控制器**

package com.example.demo.controller;

import cn.hutool.core.util.StrUtil;

import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;

import com.baomidou.mybatisplus.core.toolkit.Wrappers;

import com.baomidou.mybatisplus.extension.plugins.pagination.Page;

import com.example.demo.common.Result;

import com.example.demo.entity.User;

import com.example.demo.mapper.UserMapper;

import org.springframework.web.bind.annotation.\*;

import javax.annotation.Resource;

import java.sql.Wrapper;

import com.mongodb.client.result.DeleteResult;

import com.mongodb.client.result.UpdateResult;

import com.example.demo.entity.Try;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Example;

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 java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.regex.Pattern;

//@RestController的意思是表示自己为返回json的控制器

@RestController

@RequestMapping("/try\_mongodb")//@RequestMapping是usercontroller统一的路由

public class TryController {

@Autowired

private MongoTemplate mongoTemplate;

@GetMapping("/shows")//定义post的接口

public Result<?> findTry() {

List<Try> tryList = mongoTemplate.findAll(Try.class);

return Result.success(tryList);

}

}