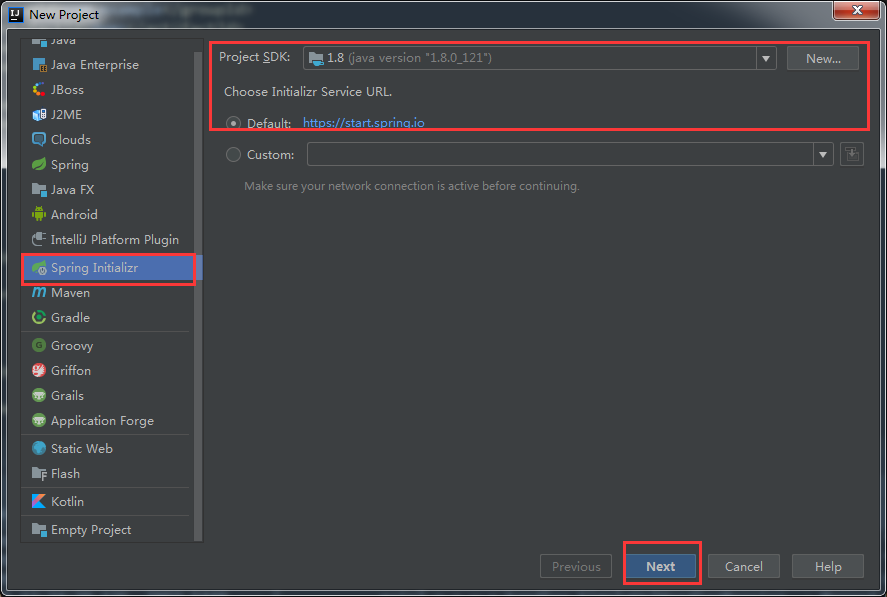
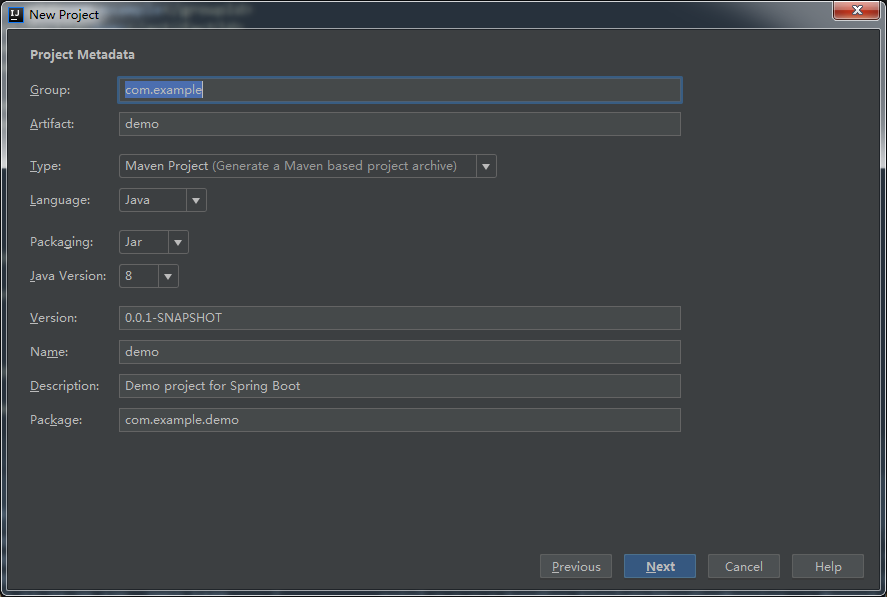
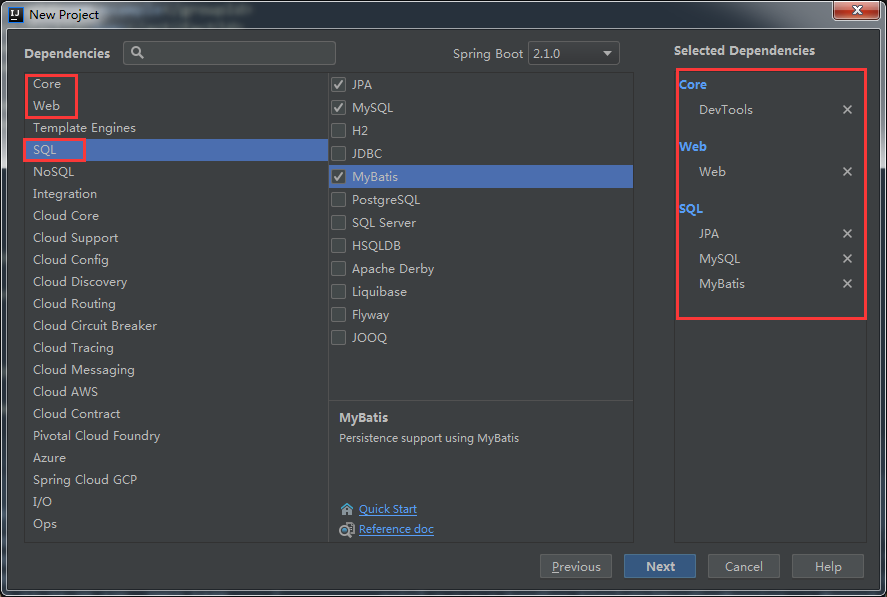
# Springboot整合mybatis

### 一、新建springboot项目



点出下一步

下一步



下一步，填写项目名称，点击完成

二、操作步骤：

1、新建实体类：在com.example.demo.entity包下新建User类

|  |
| --- |
| public class User {  private Long id;  private String name;  private int age;  private Date birthday;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public Date getBirthday() {  return birthday;  }  public void setBirthday(Date birthday) {  this.birthday = birthday;  }  } |

新建数据表：

|  |
| --- |
| CREATE TABLE USER(  id INT PRIMARY KEY AUTO\_INCREMENT,  NAME VARCHAR(30),  age INT,  birthday DATE  ) |

在resources下新建application.yml:

|  |
| --- |
| #默认使用配置  spring:  profiles:  active: dev  datasource:  driver-class-name: com.mysql.jdbc.Driver  url: jdbc:mysql://localhost:3306/springboot?useUnicode=true&characterEncoding=utf8&serverTimezone=UTC  username: root  password: root  # type: com.alibaba.druid.pool.DruidDataSource  #公共配置与profiles选择无关  mybatis:  typeAliasesPackage: com.example.demo.entity  mapperLocations: classpath:mapper/\*.xml |

在resources下新建mapper目录，在其下创建mapper文件：UserMapper.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >  <mapper namespace="com.example.demo.dao.UserDao" >  <resultMap id="BaseResultMap" type="com.example.demo.entity.User" >  <id column="id" property="id" jdbcType="INTEGER" />  <result column="name" property="name" jdbcType="VARCHAR" />  <result column="birthday" property="birthday" jdbcType="DATE" />  <result column="age" property="age" jdbcType="INTEGER" />  </resultMap>  <sql id="Base\_Column\_List" >  id, name, age,birthday  </sql>  <select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Long" >  select  <include refid="Base\_Column\_List" />  from user  where id = #{id,jdbcType=INTEGER}  </select>  <delete id="deleteByPrimaryKey" parameterType="java.lang.Long" >  delete from user  where id = #{id,jdbcType=INTEGER}  </delete>  <insert id="insert" parameterType="com.example.demo.entity.User" >  insert into user (id, name, birthday,  age)  values (#{id,jdbcType=INTEGER}, #{name,jdbcType=VARCHAR}, #{birthday,jdbcType=VARCHAR},  #{age,jdbcType=INTEGER})  </insert>  <insert id="insertSelective" parameterType="com.example.demo.entity.User" >  insert into user\_t  <trim prefix="(" suffix=")" suffixOverrides="," >  <if test="id != null" >  id,  </if>  <if test="name != null" >  user\_name,  </if>  <if test="birthday != null" >  password,  </if>  <if test="age != null" >  age,  </if>  </trim>  <trim prefix="values (" suffix=")" suffixOverrides="," >  <if test="id != null" >  #{id,jdbcType=INTEGER},  </if>  <if test="name != null" >  #{userName,jdbcType=VARCHAR},  </if>  <if test="birthday != null" >  #{password,jdbcType=VARCHAR},  </if>  <if test="age != null" >  #{age,jdbcType=INTEGER},  </if>  </trim>  </insert>  <update id="updateByPrimaryKeySelective" parameterType="com.example.demo.entity.User" >  update user\_t  <set >  <if test="name != null" >  user\_name = #{userName,jdbcType=VARCHAR},  </if>  <if test="birthday != null" >  password = #{password,jdbcType=VARCHAR},  </if>  <if test="age != null" >  age = #{age,jdbcType=INTEGER},  </if>  </set>  where id = #{id,jdbcType=INTEGER}  </update>  <update id="updateByPrimaryKey" parameterType="com.example.demo.entity.User" >  update user  set name = #{name,jdbcType=VARCHAR},  birthday = #{birthday,jdbcType=VARCHAR},  age = #{age,jdbcType=INTEGER}  where id = #{id,jdbcType=INTEGER}  </update>  </mapper> |

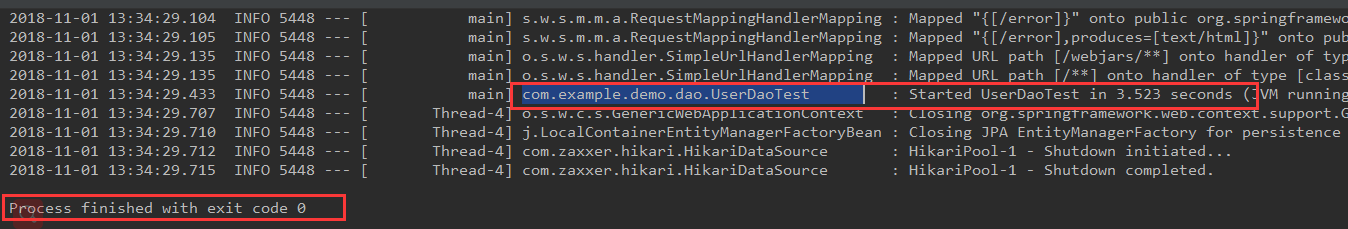
新建Dao包，创建UserDao

|  |
| --- |
| @Mapper  public interface UserDao {  int deleteByPrimaryKey(Long id);  int insert(User record);  int insertSelective(User record);  User selectByPrimaryKey(Long id);  int updateByPrimaryKeySelective(User record);  int updateByPrimaryKey(User record);  } |

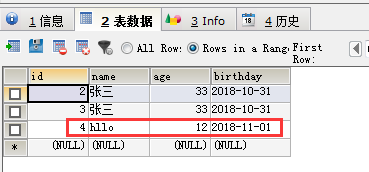
测试类UserDaoTest

|  |
| --- |
| @RunWith(SpringRunner.class)  @SpringBootTest  public class UserDaoTest {  @Resource  private UserDao userDao;  @Test  public void insertTest(){  User user = new User();  user.setName("hllo");  user.setAge(12);  user.setBirthday(new Date());  userDao.insert(user);  }  } |

执行结果：



数据库



### 三、使用druid数据源：

Druid参考：https://github.com/alibaba/druid/tree/master/druid-spring-boot-starter

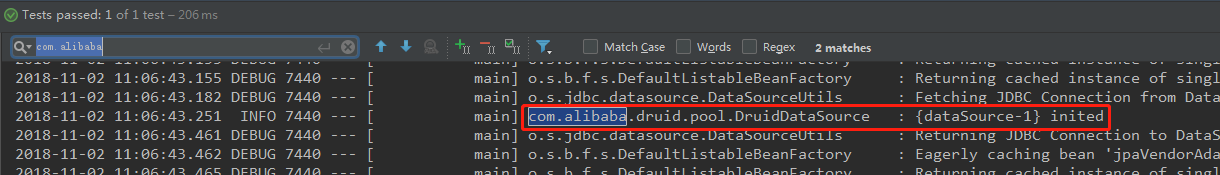
添加依赖：

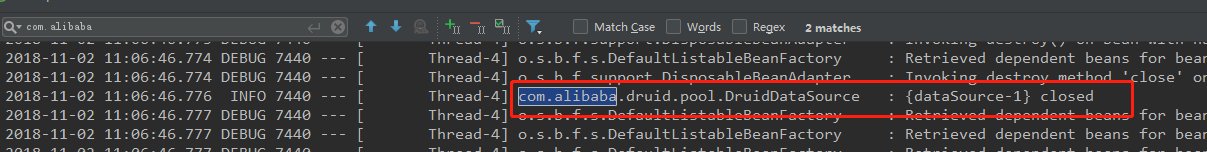
|  |
| --- |
| <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.1.12</version>  </dependency> |

修改配置文件application.yml

|  |
| --- |
| #默认使用配置  spring:  profiles:  active: dev  datasource:  driver-class-name: com.mysql.jdbc.Driver  url: jdbc:mysql://localhost:3306/springboot?useUnicode=true&characterEncoding=utf8&serverTimezone=UTC  username: root  password: root  # 使用druid数据源  type: com.alibaba.druid.pool.DruidDataSource  #公共配置与profiles选择无关  mybatis:  typeAliasesPackage: com.example.demo.entity  mapperLocations: classpath:mapper/\*.xml |

结果



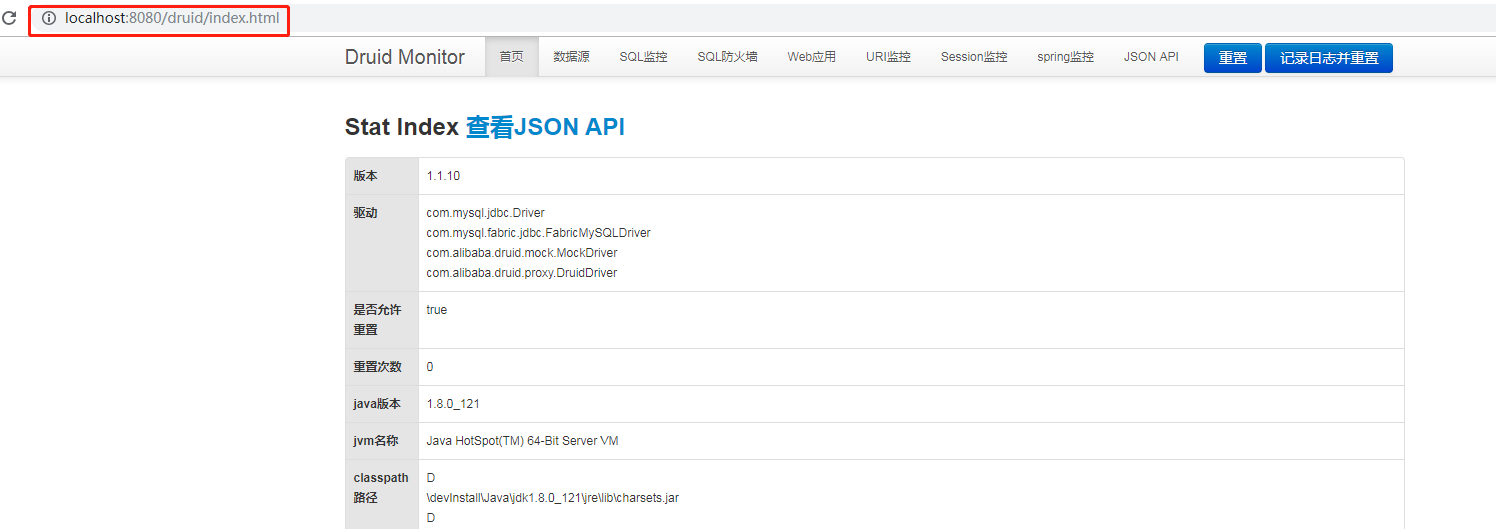


### 四、Druid 监控

将依赖改成：

|  |
| --- |
| <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid-spring-boot-starter</artifactId>  <version>1.1.10</version>  </dependency> |

重新启动服务，访问<http://localhost:8080/druid/index.html>



此时默认可以直接访问

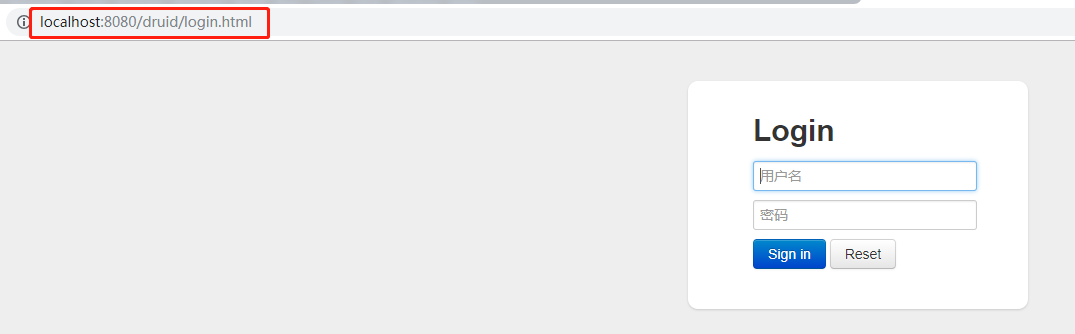
添加权限(application.yml形式)：

|  |
| --- |
| spring:  datasource:  druid:  web-stat-filter.enabled: true  web-stat-filter.url-pattern: /\*  web-stat-filter.exclusions: /druid/\*,\*.js,\*.gif,\*.jpg,\*.png,\*.css,\*.ico  # druid页面监控  stat-view-servlet.enabled: true  stat-view-servlet.url-pattern: /druid/\*  stat-view-servlet.login-username: druid  stat-view-servlet.login-password: druid123 |

application.properties形式文件

|  |
| --- |
| # druid 监控  spring.datasource.druid.web-stat-filter.enabled=true  spring.datasource.druid.web-stat-filter.url-pattern=/\*  spring.datasource.druid.web-stat-filter.exclusions=/druid/\*,\*.js,\*.gif,\*.jpg,\*.png,\*.css,\*.ico  # druid 监控页面  spring.datasource.druid.stat-view-servlet.enabled=true  spring.datasource.druid.stat-view-servlet.url-pattern=/druid/\*  spring.datasource.druid.stat-view-servlet.login-username=druid  spring.datasource.druid.stat-view-servlet.login-password=druid123 |

此时需要登录



# Springboot缓存和nosql

1. Redis 缓存

添加依赖

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-redis</artifactId>  </dependency> |

添加配置：在application.yml添加redis配置

|  |
| --- |
| spring:  # redis 配置  redis:  host: 192.168.1.171  port: 6379  password: redis  # 缓存  cache:  # 过期时间，单位毫秒  redis:  time-to-live: 60000ms  # 缓存类型（ehcache、redis）  type: redis |

使用上面mybatis做测试：

Service层：

DepartmentService接口

|  |
| --- |
| package com.example.demo.services;  import com.example.demo.entity.Department;  public interface DepartmentService {  Department save(Department department);  Department update(Department department);  Department getDepartmentById(Long id);  void delete(Long id);  } |

实现类

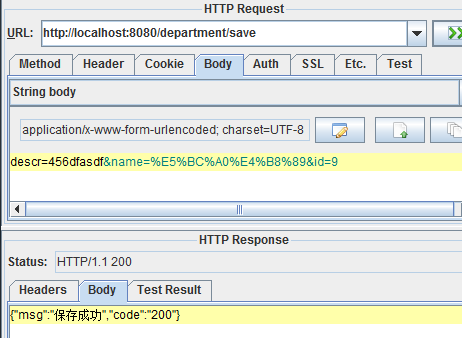
|  |
| --- |
| package com.example.demo.services.impl;  import com.example.demo.dao.DepartmentDao;  import com.example.demo.entity.Department;  import com.example.demo.services.DepartmentService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.cache.annotation.CacheConfig;  import org.springframework.cache.annotation.CacheEvict;  import org.springframework.cache.annotation.CachePut;  import org.springframework.cache.annotation.Cacheable;  import org.springframework.stereotype.Service;  import javax.annotation.Resource;  @CacheConfig(cacheNames = "department")  @Service  public class DepartmentServiceImpl implements DepartmentService {  @Resource  private DepartmentDao departmentMapper;  @CachePut(key = "#department.id")  public Department save(Department department) {  System.out.println("保存 id=" + department.getId() + " 的数据");  this.departmentMapper.insert(department);  return department;  }  @CachePut(key = "#department.id")  public Department update(Department department) {  System.out.println("修改 id=" + department.getId() + " 的数据");  this.departmentMapper.update(department);  return department;  }  @Cacheable(key = "#id")  public Department getDepartmentById(Long id) {  System.out.println("获取 id=" + id + " 的数据");  Department department = this.departmentMapper.getById(id);  return department;  }  @CacheEvict(key = "#id")  public void delete(Long id) {  System.out.println("删除 id=" + id + " 的数据");  this.departmentMapper.deleteById(id);  }  } |

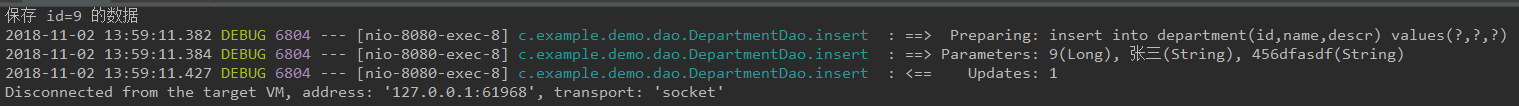
Controller层

|  |
| --- |
| package com.example.demo.controller;  import com.example.demo.entity.Department;  import com.example.demo.services.DepartmentService;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.ResponseBody;  import org.springframework.web.bind.annotation.RestController;  import javax.annotation.Resource;  import java.util.HashMap;  import java.util.Map;  @RestController  @RequestMapping("department")  @ResponseBody  public class DepartmentController {  @Resource  private DepartmentService departmentService;  @RequestMapping("save")  public Map<String,Object> save(Department department) {  this.departmentService.save(department);  Map<String,Object> map = new HashMap<String,Object>();  map.put("code", "200");  map.put("msg", "保存成功");  return map;  }  @RequestMapping("get/{id}")  public Map<String,Object> get(@PathVariable("id") Long id) {  Department department = this.departmentService.getDepartmentById(id);  Map<String,Object> map = new HashMap<String,Object>();  map.put("code", "200");  map.put("msg", "获取成功");  map.put("data", department);  return map;  }  @RequestMapping("update")  public Map<String,Object> update(Department department) {  this.departmentService.update(department);  Map<String,Object> map = new HashMap<String,Object>();  map.put("code", "200");  map.put("msg", "修改成功");  return map;  }  @RequestMapping("delete/{id}")  public Map<String,Object> delete(@PathVariable("id") Long id) {  this.departmentService.delete(id);  Map<String,Object> map = new HashMap<String,Object>();  map.put("code", "200");  map.put("msg", "删除成功");  return map;  }  } |

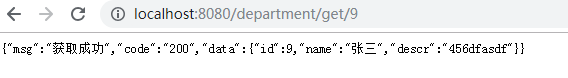
启动类添加注解：@EnableCaching

第一次访问：保存数据



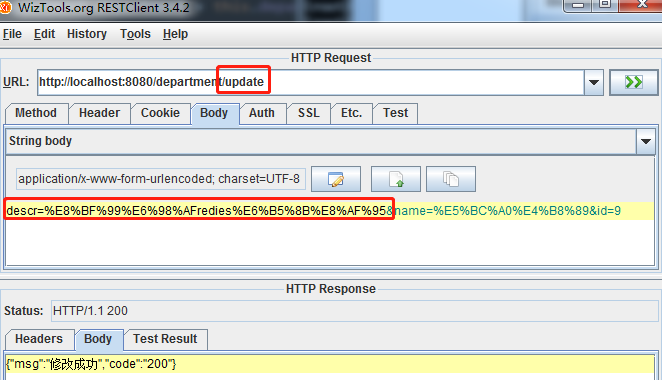


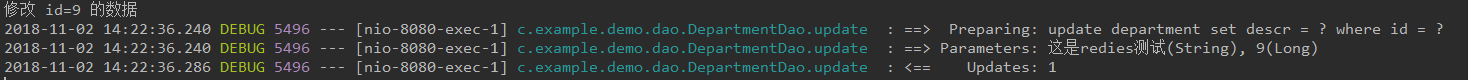
获取



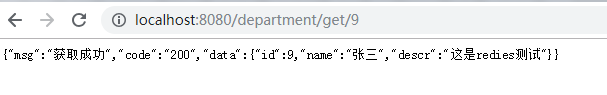
没有打印日志（说明从缓冲中获取数据）

更新

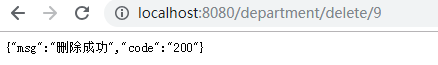


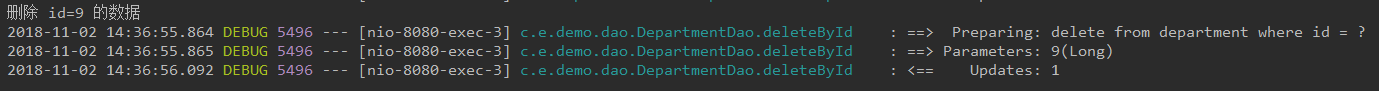


再次获取

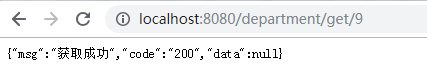
没有打日志(获取内容为修改后的，说明修改时更新了缓存及数据库)

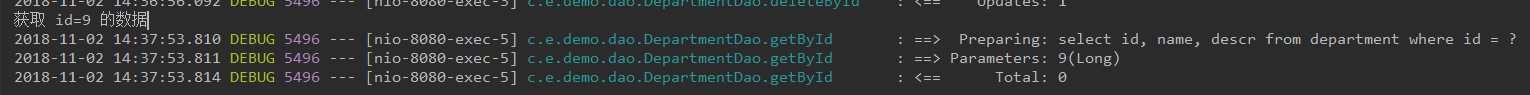
删除





第三次获取





获取数据为空，并查询了数据库，说明删除了缓存及数据库记录

1. nosql数据库
2. 整合redis（要有redis数据库）

添加依赖

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-redis</artifactId>  </dependency>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-pool2</artifactId>  <version>2.6.0</version>  </dependency> |

配置连接：application.properties形式

|  |
| --- |
| spring.redis.host=192.168.1.171  spring.redis.port=6379  spring.redis.password=redis |

application.yml形式：

|  |
| --- |
| spring:  redis:  host: 192.168.1.171  port: 6379  password: redis |

编码：

|  |
| --- |
| package com.example.demo.nosql;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.data.redis.core.StringRedisTemplate;  import org.springframework.stereotype.Component;  @Component  public class RedisDao {  @Autowired  private StringRedisTemplate stringRedisTemplate;  public void set(String key, String value) {  this.stringRedisTemplate.opsForValue().set(key, value);  }  public String get(String key) {  return this.stringRedisTemplate.opsForValue().get(key);  }  public void delete(String key) {  this.stringRedisTemplate.delete(key);  }  } |

测试类：

|  |
| --- |
| package com.example.demo.nosql;  import org.junit.Test;  import org.junit.runner.RunWith;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.boot.test.context.SpringBootTest;  import org.springframework.test.context.junit4.SpringRunner;  @RunWith(SpringRunner.class)  @SpringBootTest  public class RedisDaoTest {  @Autowired  private RedisDao redisDao;  @Test  public void testSet() {  String key = "name";  String value = "zhangsan";  this.redisDao.set(key, value);  }  @Test  public void testGet() {  String key = "name";  String value = this.redisDao.get(key);  System.out.println(value);  }  @Test  public void testDelete() {  String key = "name";  this.redisDao.delete(key);  }  } |

注意，要添加commons-pool2依赖，

否则报org/apache/commons/pool2/impl/GenericObjectPoolConfig类找不到。

还有redis连接超时时间不能设置为0，否则连接不上，报连接超时错误

# 连接超时时间（毫秒）

spring.redis.timeout=0（不能为0，可以设置5000ms）

2、整合 MongoDB（要提前装好MongoDB数据库）

添加依赖：

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-mongodb</artifactId>  </dependency> |

连接配置：application.properties形式

|  |
| --- |
| spring.data.mongodb.host=192.168.2.31  spring.data.mongodb.port=27017  spring.data.mongodb.database=test |

application.yml形式：

|  |
| --- |
| spring:  data:  mongodb:  host: 127.0.0.1  port: 27017  database: taobao |

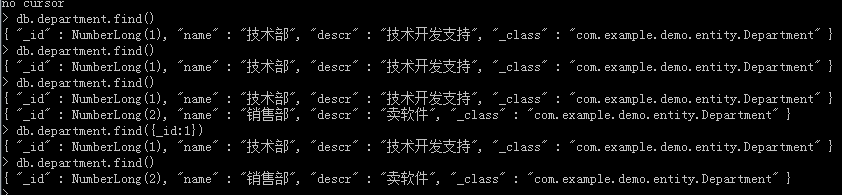
MongoTemplate 模板：

|  |
| --- |
| package com.example.demo.nosql;  import com.example.demo.entity.Department;  import org.springframework.beans.factory.annotation.Autowired;  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.Component;  import java.util.List;  @Component  public class MongodbDao {  @Autowired  private MongoTemplate mongoTemplate;  public void insert(Department department) {  this.mongoTemplate.insert(department);  }  public void deleteById(int id) {  Criteria criteria = Criteria.where("id").is(id);  Query query = new Query(criteria);  this.mongoTemplate.remove(query, Department.class);  }  public void update(Department department) {  Criteria criteria = Criteria.where("id").is(department.getId());  Query query = new Query(criteria);  Update update = new Update();  update.set("descr", department.getDescr());  this.mongoTemplate.updateMulti(query, update, Department.class);  }  public Department getById(int id) {  Criteria criteria = Criteria.where("id").is(id);  Query query = new Query(criteria);  return this.mongoTemplate.findOne(query, Department.class);  }  public List<Department> getAll() {  List<Department> userList = this.mongoTemplate.findAll(Department.class);  return userList;  }  } |

测试类：

|  |
| --- |
| package com.example.demo.nosql;  import com.example.demo.entity.Department;  import org.junit.Test;  import org.junit.runner.RunWith;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.boot.test.context.SpringBootTest;  import org.springframework.test.context.junit4.SpringRunner;  @RunWith(SpringRunner.class)  @SpringBootTest  public class MongodbDaoTest {  @Autowired  private MongodbDao mongodbDao;  @Test  public void testSet() {  Department dept = new Department();  dept.setId(2L);  dept.setName("销售部");  dept.setDescr("卖软件");  this.mongodbDao.insert(dept);  }  @Test  public void testGet() {  Department dept = this.mongodbDao.getById(1);  System.out.println(dept);  }  @Test  public void testDelete() {  this.mongodbDao.deleteById(1);  }  } |

测试结果



使用 Repository：

|  |
| --- |
| package com.example.demo.nosql;  import com.example.demo.entity.Department;  import org.springframework.data.mongodb.repository.MongoRepository;  public interface DepartmentRepository extends MongoRepository<Department, Integer> {  } |

测试类：

|  |
| --- |
| package com.example.demo.nosql;  import com.example.demo.entity.Department;  import org.junit.Test;  import org.junit.runner.RunWith;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.boot.test.context.SpringBootTest;  import org.springframework.test.context.junit4.SpringRunner;  import java.util.List;  @RunWith(SpringRunner.class)  @SpringBootTest  public class DepartmentRepositoryTest {  @Autowired  private DepartmentRepository departmentRepository;  @Test  public void insertTest(){  Department dept = new Department();  dept.setId(3L);  dept.setName("销售部");  dept.setDescr("卖软件");  departmentRepository.save(dept);  }  @Test  public void getAllTest(){  List<Department> departmentList = departmentRepository.findAll();  for (Department department: departmentList) {  System.out.println(department);  }  }  @Test  public void delTest(){  departmentRepository.deleteById(2);  }  } |