springboot 整合mybatis并使用 Redis 作为缓存组件 demo

- Auth：HMStrange-TIAN　Time:2019年4月13日　e-mail:zhangqihao@hnu.edu.cn
- 开发环境：window10　IDEA2018
-
-
- 1、安装docker & redis
  - 如果不清楚docker是什么，请查看docker的文档和简介，这里给出docker的安装过程
  - 1.1 安装虚拟机（如果有远程服务器的，请略过此步骤）
    - 本文推荐VMvare，尽管vmvare比较臃肿，但是对于新手比较友好，配置很简单
    - 从官网下载VMvare，官网地址：https://www.vmware.com/cn.html
    - 从官网下载centos镜像文件，官网地址：https://www.centos.org/download/
    - 打开VMvare创建虚拟机，导入镜像系统
    - Vmvare会自动配置，根据提示输入账户和密码之后，等待自动配置即可
  - 1.2 打开虚拟机的terminal，输入ifconfig查看ip地址，如图：
    -

```
[root@localhost tian]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
        inet6 fe80::42:93ff:fe83:7a1  prefixlen 64  scopeid 0x20<link>
        ether 02:42:93:83:07:a1  txqueuelen 0  (Ethernet)
        RX packets 256  bytes 19046 (18.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 305  bytes 20181 (19.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.232.133  netmask 255.255.255.0  broadcast 192.168.232.255
        inet6 fe80::3e90:f40c:162d:b3a5  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:8b:01:d5  txqueuelen 1000  (Ethernet)
        RX packets 7422  bytes 591202 (577.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1626  bytes 167880 (163.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
```

  - 1.3 使用远程工具连接服务器，本文推荐使用Cygwin/SmartTTY/Putty/GitBash
    - 打开连接工具，使用ssh root@192.168.xx.xx，登陆服务器即可操作
  - 1.4 安装docker
    - 1.4.1 检查内核版本，必须是3.10及以上
      - uname -r
    - 1.4.2 安装docker
      - yum install docker
      - 输入 y 确认安装
    - 1.4.3 启动docker
      - systemctl start docker
      - 查看docker时候安装成功
        - docker -v

        - 若有提示如：Docker version 1.12.6, build 3e8e77d/1.12.6，则安装成功
    - 设置开机启动docker
        - systemctl enable docker
    - 如果想停止docker(慎重！！！)
        - systemctl stop docker
- 1.4.4 常见docker命令以及操作
    - a）镜像操作
        - 检索镜像
            - docker search **keyword**
            - 例如：docker search mysql
        - 拉取镜像
            - docker pull **iamges**
            - 例如：docker pull registry.docker-cn.com/library/mysql
        - 查看镜像列表
            - docker images
        - 删除镜像
            - docker rmi image(镜像)-id
    - b) 容器操作
        - 根据拉取的镜像启动容器(可以docker images查看已有的镜像，启动需要的镜像)
            - docker run --name mymysql -d mysql:latest
            - --name后面是容器的名字 -d 表示后台运行 latest是tag标签，表示最新版本
        - 查看运行中的容器、
            - docker ps
        - 停止运行中的容器
            - docker stop 容器的id
        - 查看所有的容器
            - docker ps -a
        - 启动容器
            - docker start 容器id
        - 删除一个容器
            - docker rm 容器id
        - 启动一个做了端口映射的容器
            - docker run -d -p 8080：8080
            - -d：后台运行 -p : 将主机的端口映射到容器的一个端口 主机端口：容器内容端口
        - 更多命令和操作请查看docker官网
- 1.5 使用docker 安装 redis
    - 1.5.1 搜索镜像

-

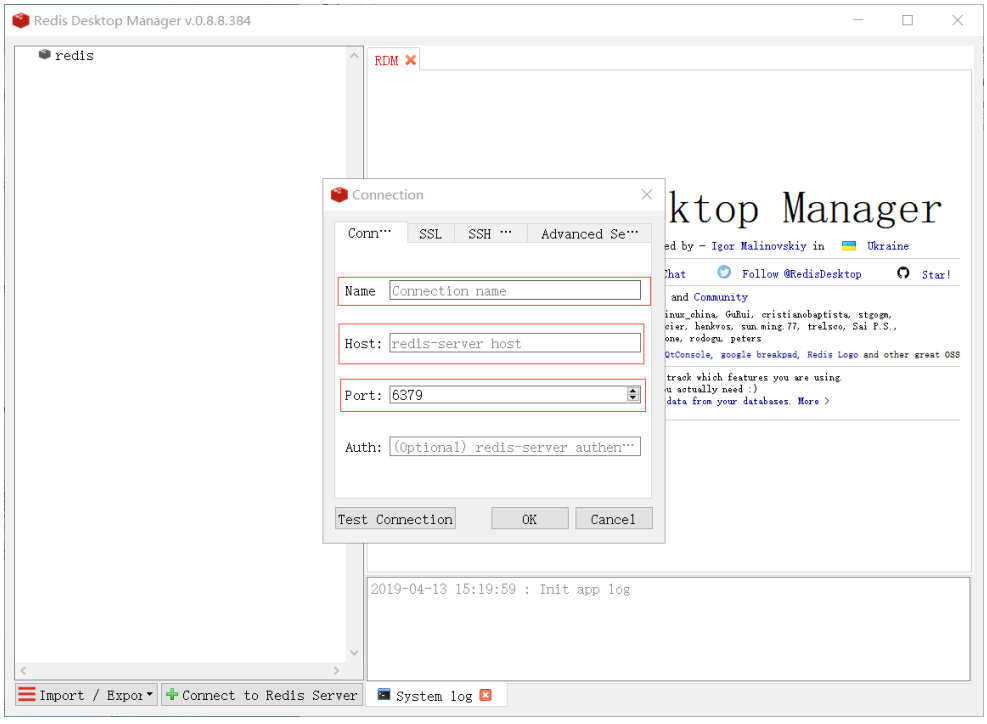
```
[root@localhost tian]# docker search redis
INDEX       NAME                                      DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
docker.io   docker.io/redis                           Redis is an open source key-value store th...   6756      [OK]
docker.io   docker.io/bitnami/redis                   Bitnami Redis Docker Image                      110                  [OK]
docker.io   docker.io/sameersbn/redis                                                                 76                   [OK]
docker.io   docker.io/grokzen/redis-cluster           Redis cluster 3.0, 3.2, 4.0 & 5.0               44
docker.io   docker.io/hypriot/rpi-redis               Raspberry Pi compatible redis image             34
docker.io   docker.io/kubeguide/redis-master          redis-master with "Hello World!"                28
docker.io   docker.io/rediscommander/redis-commander  Alpine image for redis-commander - Redis m...   22                   [OK]
docker.io   docker.io/redislabs/redis                 Clustered in-memory database engine compat...   19
docker.io   docker.io/redislabs/redisearch            Redis With the RedisSearch module pre-load...   15
docker.io   docker.io/arm32v7/redis                   Redis is an open source key-value store th...   14
docker.io   docker.io/oliver006/redis_exporter        Prometheus Exporter for Redis Metrics. Su...    10
docker.io   docker.io/webhippie/redis                 Docker images for Redis                         10                   [OK]
docker.io   docker.io/insready/redis-stat             Docker image for the real-time Redis monit...   7                    [OK]
docker.io   docker.io/s7anley/redis-sentinel-docker   Redis Sentinel                                  7                    [OK]
docker.io   docker.io/arm64v8/redis                   Redis is an open source key-value store th...   6
docker.io   docker.io/rtoma/logspout-redis-logstash   Logspout including Redis adapter for sendi...   5
docker.io   docker.io/centos/redis-32-centos7         Redis in-memory data structure store, used...   4
docker.io   docker.io/redislabs/redisgraph            A graph database module for Redis               4                    [OK]
docker.io   docker.io/circleci/redis                  CircleCI images for Redis                       2                    [OK]
docker.io   docker.io/frodenas/redis                  A Docker Image for Redis                        2                    [OK]
docker.io   docker.io/wodby/redis                     Redis container image with orchestration        2                    [OK]
docker.io   docker.io/tiredofit/redis                 Redis Server w/ Zabbix monitoring and S6 O...   1                    [OK]
docker.io   docker.io/cflondonservices/redis          Docker image for running redis                  0
docker.io   docker.io/iadvize/redis                                                                   0
docker.io   docker.io/xetamus/redis-resource         forked redis-resource                           0                    [OK]
[root@localhost tian]#
```

- 1.5.2 拉取镜像
  - docker pull docker.io/redis
- 1.5.3 查看镜像
  -

```
[root@localhost tian]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker.io/redis     latest              a55fbf438dfd        2 weeks ago         95 MB
```

- 1.5.4 运行镜像
  - docker run -d -p 6379:6379 --name myredis docker.io/redis
- 1.5.5 查看运行中的镜像
  - docker ps
- 此时，使用docker安装、运行镜像已经完成了

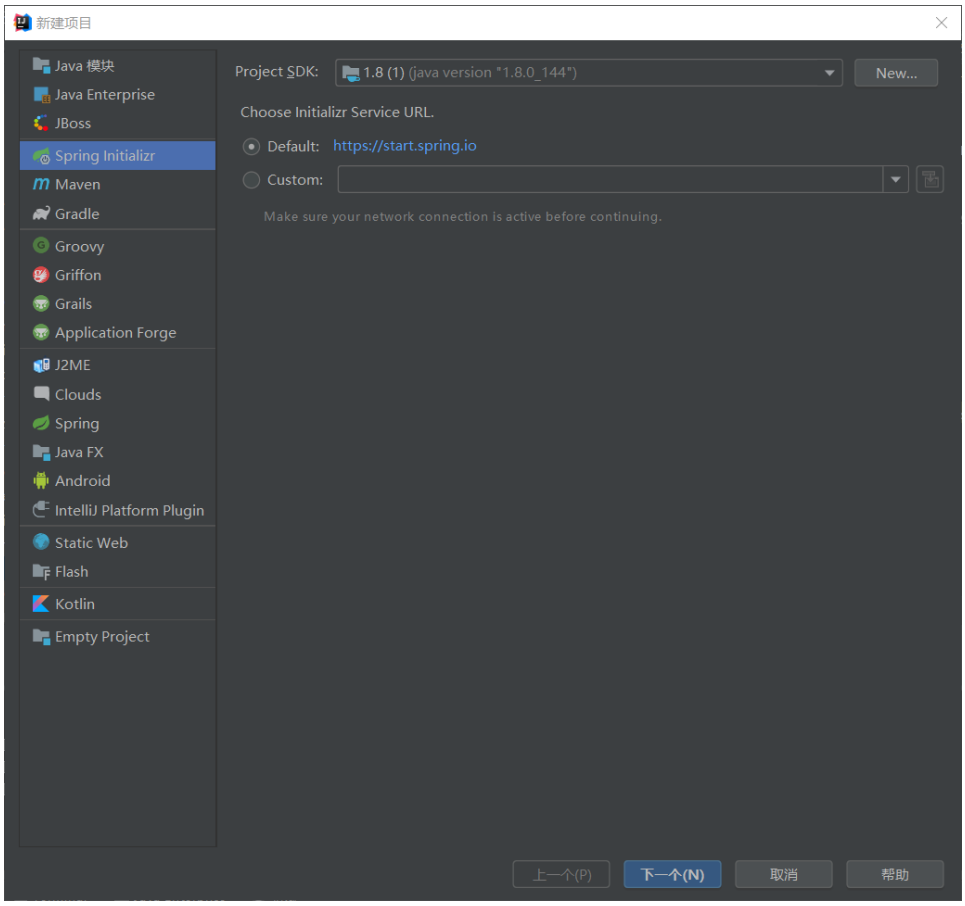
- 1.6 使用RedisDesktopManager连接Redis数据库
  - 下载地址：https://redisdesktop.com/download
  - 设置连接名、主机名字（就是我们前面输入ifconfig查看得到的ip）、端口号（暴露的那个端口号就是用那个端口号，默认为6379）



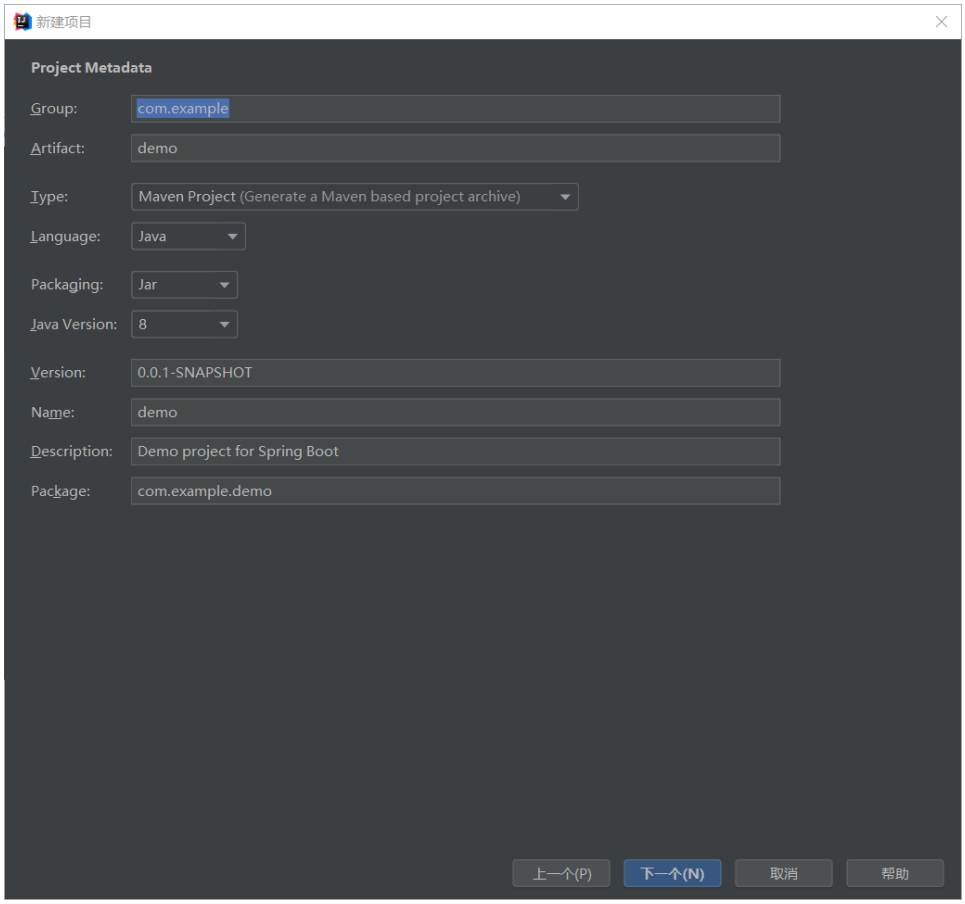
  - 点击Tes tConnection 显示 successful 点击 OK
  - 更多关于redis的操作命令请查看官网：

- http://www.redis.cn/

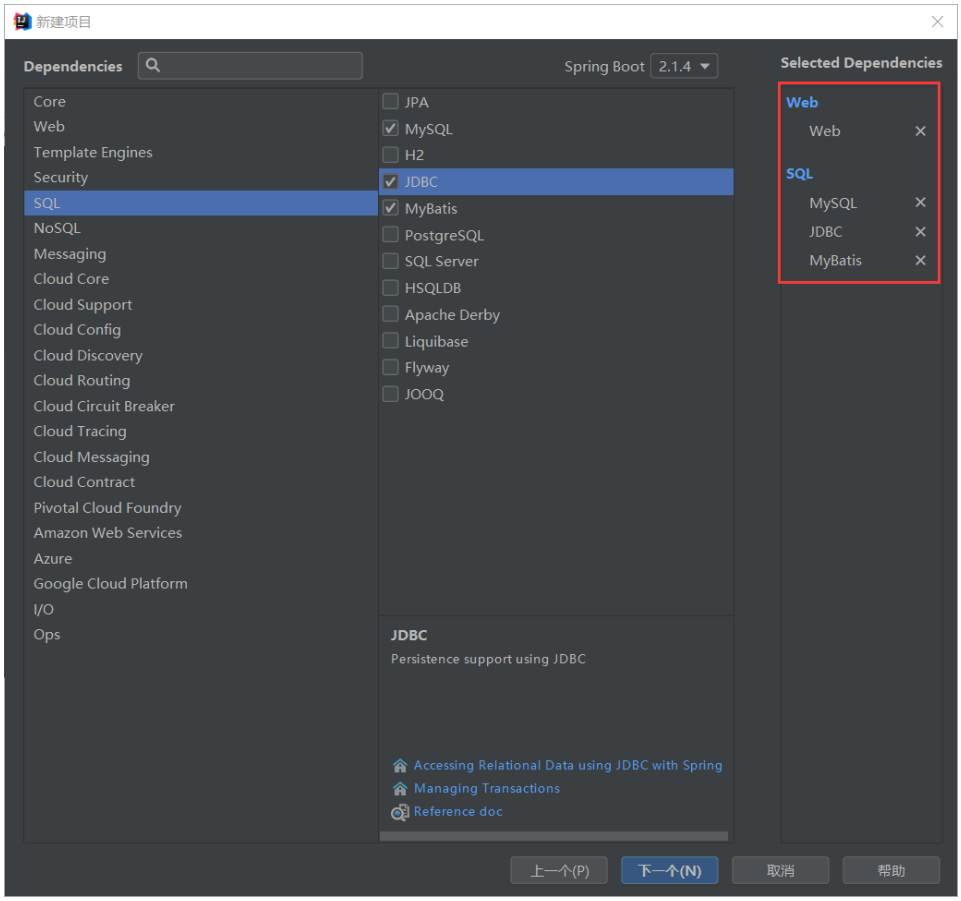
- 2、springboot整合mybatis
  - 2.1、打开IDEA，使用springboot Initializr 快速创建向导
    - 点击下一步



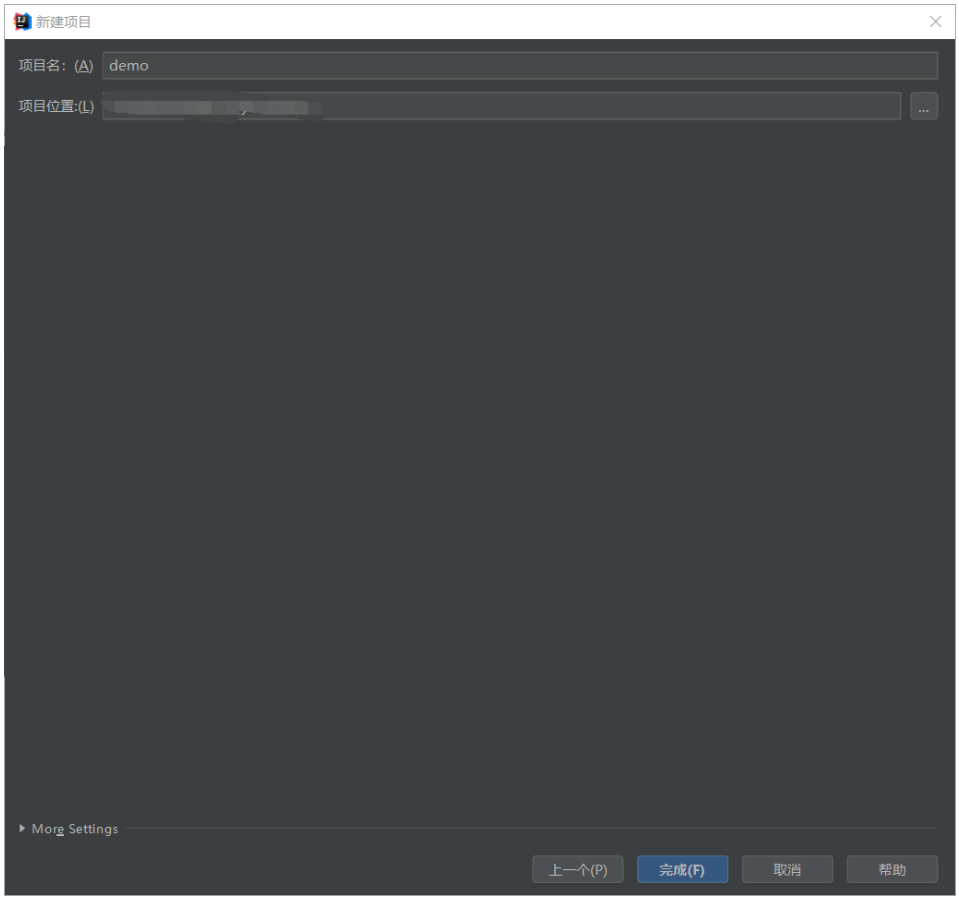
    - 输入相应的Group、Artifact(不会的请先学习IDEA)



- 选择相应的模块，如右下方红框所示



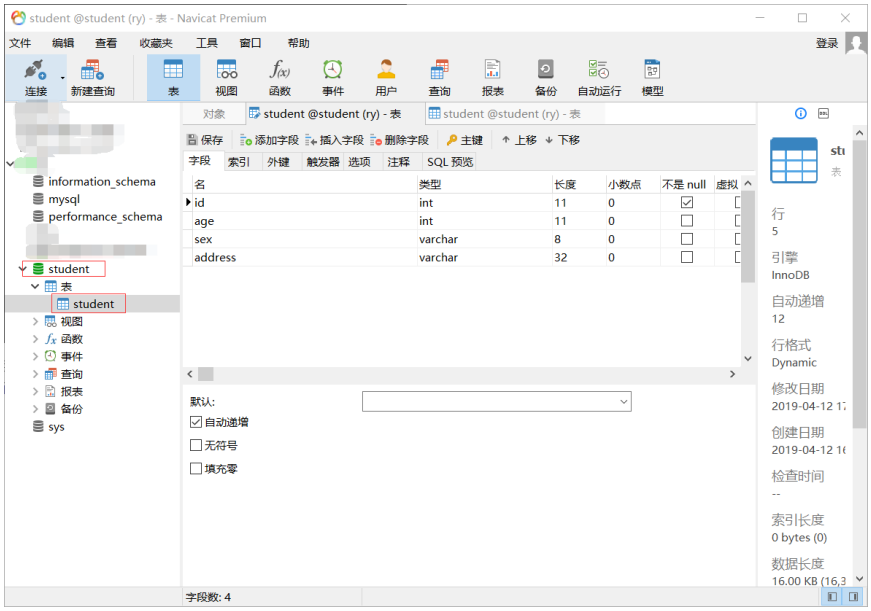
- 输入项目name和项目address



- 2.2、创建完成后，可以看到pom文件中引入了相应的starter
  -

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-cache</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>2.0.1</version>
</dependency>

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

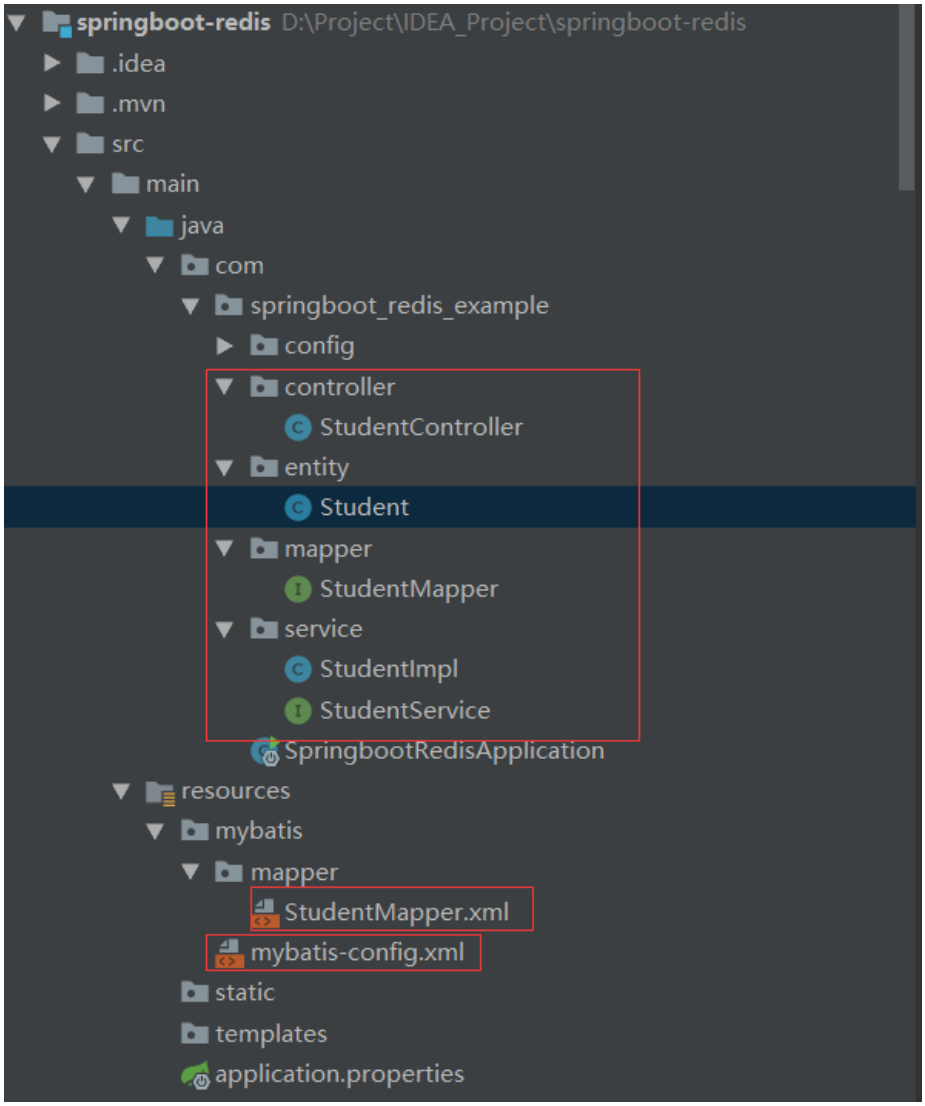
- 2.3、在IDEA中配置mysql数据库

- 2.3.1配置mysql
  - 新建数据库student，新建表student
    -


  - 注：关于如何安装mysql、navicat以及如何使用请自行百度
- 2.3.2在项目的目录结构中找到application.properties或者新建一个application.yml(关于yml的语法请自行百度)
  - url的配置规则请百度，输入自己数据库的用户名和密码

```
# mysql 配置
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/student?useUn
spring.datasource.username=
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

- 2.4、编码：新建entity实体类、service、service的实现类、以及mapper接口，然后在resource目录下建立对应的mapper以及mabatis的配置文件
  - 2.4.1项目目录结构如下：
  -



- 2.4.2 entity代码如下
  -

```java
/**
 * @Author: zhangqihao
 * @Date: 2019/4/11 20:47
 * @Version 1.0
 */
public class Student implements Serializable {
    //学生ID
    private Integer id;
    //学生年龄
    private int age;
    //学生性别
    private String sex;
    //学生住址
    private String address;

    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    public int getAge() { return age; }

    public void setAge(int age) { this.age = age; }

    public String getSex() { return sex; }

    public void setSex(String sex) { this.sex = sex; }

    public String getAddress() { return address; }

    public void setAddress(String address) { this.address = address; }
}
```

- 2.4.3 Service代码如下
  -

```java
import com.springboot_redis_example.entity.Student;



/**
 * @Author: zhangqihao
 * @Date: 2019/4/11 20:52
 * @Version 1.0
 */
public interface StudentService {

    /*
     *查询数据
     */
    public Student selectStudentById(Integer id);
    //写入数据
    public int insertStudent(Student student);
    //更新数据
    public int updateStudent(Student student);
    //删除数据
    public int deleteStudent(Integer id);
}
```

- 2.4.4 Service实现方法如下
  - 注意：在实现方法上加 @Service注解

```java
/**
 * @Author: zhangqihao
 * @Date: 2019/4/11 21:18
 * @Version 1.0
 */
@Service("StudentService")
public class StudentImpl implements StudentService{

    @Autowired
    private StudentMapper studentMapper;

    /**...*/
    @Override
    public Student selectStudentById(Integer id) { return studentMapper.selectStudentById(id); }

    /**...*/
    @Override
    public int insertStudent(Student student) { return studentMapper.insertStudent(student); }

    /**...*/
    @Override
    public int updateStudent(Student student) { return studentMapper.updateStudent(student); }

    /**...*/
    @Override
    public int deleteStudent(Integer id) { return studentMapper.deleteStudentById(id); }
}
```

- 2.4.5 mapper如下
  - 注意:在接口上方加@mapper注解

```java
/**...*/
@Mapper
public interface StudentMapper {
    /**...*/
    public Student selectStudentById(Integer id);

    /**
     * 新增用户管理
     *
     */
    public int insertStudent(Student student);

    /**
     * 修改用户管理
     *
     */
    public int updateStudent(Student student);

    /**
     * 删除用户管理
     *
     */
    public int deleteStudentById(Integer id);
}
```

- 2.4.6 Controller如下
  - 注意：加@RestConroller注解

```java
/**
 * @Author: zhangqihao
 * @Date: 2019/4/11 22:44
 * @Version 1.0
 */
@RestController
public class StudentController {

    @Autowired
    private StudentService studentService;

    @GetMapping("/student/{id}")
    public Student getStudentById(@PathVariable("id") Integer id) { return studentService.selectStudentById(id); }

    @PostMapping("/student/add")
    public int insertStudent(Student student) { return studentService.insertStudent(student); }
}
```

- 2.4.7 mapper映射文件如下
  - 关于映射文件的语法，请查看官方文档，此处给出mybatis的中文文档：
    - http://www.mybatis.org/mybatis-3/zh/sqlmap-xml.html
  - resource/mybatis/mapper/StudentMapper.xml(此文件的路径)

```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.springboot_redis_example.mapper.StudentMapper">
    <!--BUG一，没有指定别名       -->
    <resultMap type="com.springboot_redis_example.entity.Student" id="StudentResult">
        <result property="id"     column="id"     /><!-- property：主键在pojo(简单javabean对象)中的属性名  column：主键在数据库中的列名 -->
        <result property="age"     column="age"     />
        <result property="sex"     column="sex"     />
        <result property="address"     column="address"     />
    </resultMap>

    <sql id="selectStudentVo">
        select * from student
    </sql>

    <select id="selectStudentById" parameterType="Integer" resultMap="StudentResult">
        <include refid="selectStudentVo"/>
        where id=#{id}
    </select>

    <insert id="insertStudent" parameterType="com.springboot_redis_example.entity.Student">
        insert into student (id,age,sex,address)
        values (#{id},#{age},#{sex},#{address})
    </insert>

    <update id="updateStudent" parameterType="com.springboot_redis_example.entity.Student">
        update student...
    </update>

    <delete id="deleteStudentById" parameterType="Integer"...>

</mapper>
```

- 2.4.8 mybatias配置文件（这里没有作任何配置，但是这个文件一定要有）
  - resource/mybatis/mybatis-config.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
<!--<settings>-->
    <!--<setting name="logImpl" value="STDOUT_LOGGING"/>-->
<!--</settings>-->
</configuration>
```

- 2.4.9 在application.properties配置mybatis
  - 这两个配置是核心，其余配置可参考官方文档

```
#mybatis 配置
mybatis.config-location=classpath:mybatis/mybatis-config.xml
mybatis.mapper-locations=classpath:mybatis/mapper/*.xml
```

- 2.4.10 在student表中插入相关数据
  - 如果不知道怎么插入数据，那么.....请百度.....

- 2.5 打开浏览器进行测试
  - 结果如下：
  - 此处用的google测试，也可以使用其他接口测试工具


```
{
    "id": 1,
    "age": 20,
    "sex": "男",
    "address": "湖南省长沙市"
}
```


- 3、springboot整合redis
  - 3.1 在pom文件中引入redis 的坐标

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

  - 3.2 在application.properties或者是application.yml中配置redis
    - host就是你的服务器的ip

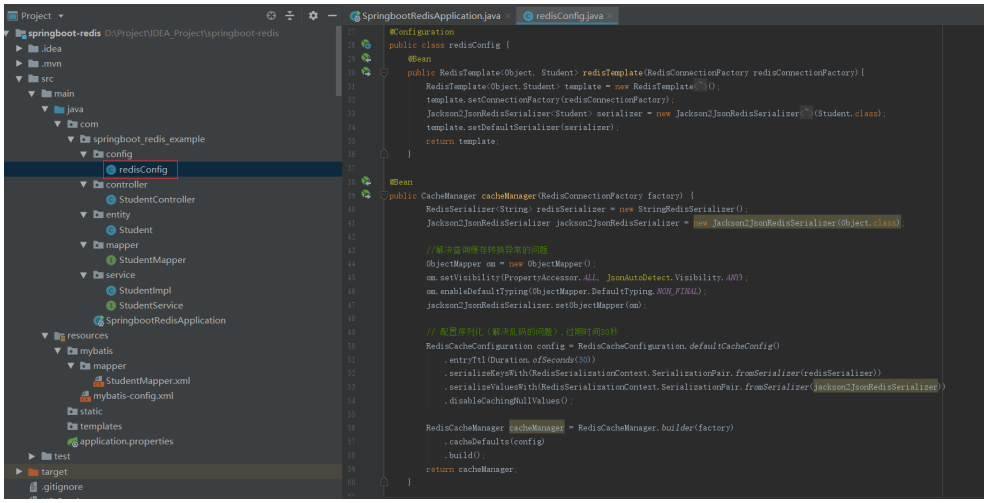
```
# redis配置
spring.redis.host=
spring.redis.port=6379
```

  - 3.3 在springboot的启动类开启缓存注解

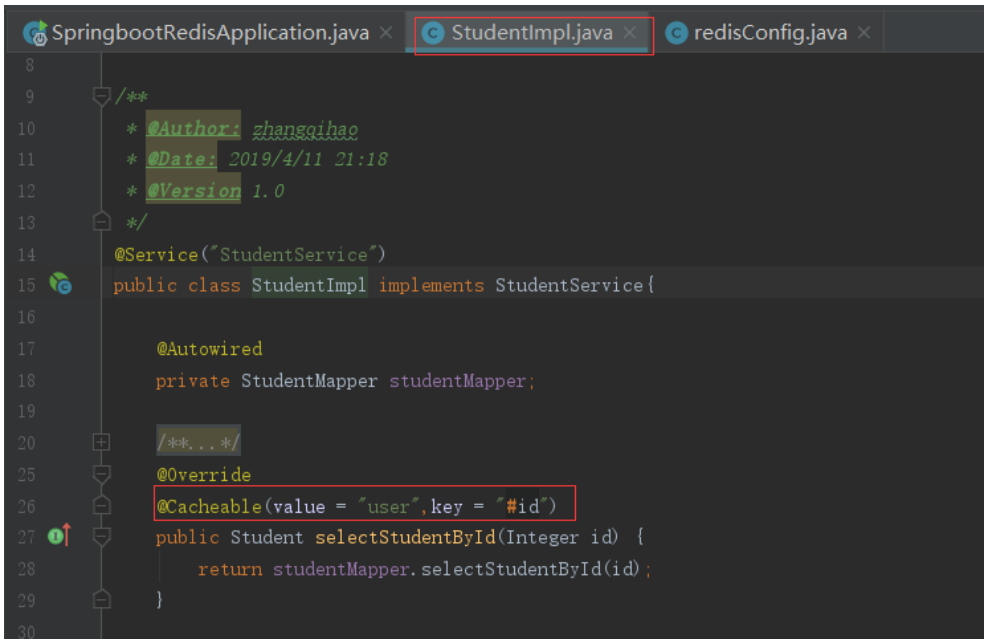
SpringbootRedisApplication.java

```
package com.springboot_redis_example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cache.annotation.EnableCaching;

@EnableCaching
@SpringBootApplication
public class SpringbootRedisApplication {
    public static void main(String[] args) { SpringApplication.run(SpringbootRedisApplication.class, args); }

}
```

  - 3.4 新建redisConfig类配置redis
    - 不要忘记加@Configuration，两个bean都是为了改变序列化的机制



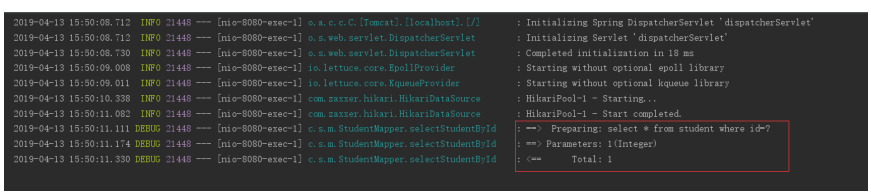
- 3.5 在service的实现类上开启注解
  -

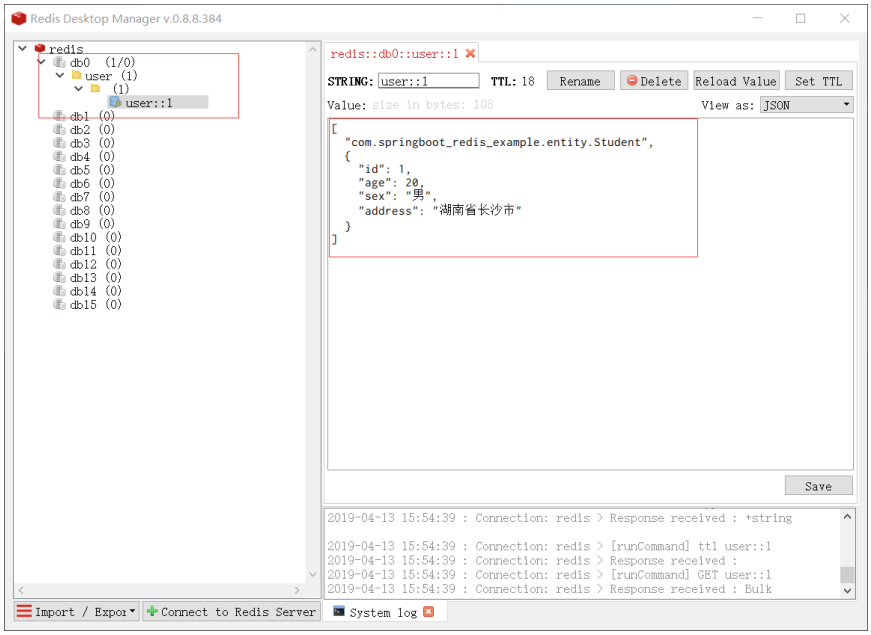


- 3.6 测试结果，
  - 3.6.1 先开启日志打印
    - 红框内是mapper的相对路径

```
#打印sql语句日志
logging.level.com.springboot_redis_example.mapper=debug
```

  - 3.6.2 第一次在浏览器请求会发现，控制台打印了sql语句
    - 发起请求，在浏览器地址栏输入：
      - http://127.0.0.1:8080/student/1
    - 查看控制台
      - 此时，student对象已被缓存到了redis中



- 3.6.2 第二次从浏览器发起请求，发现控制台没有打印sql日志，说明缓存成功，使用RedisDesktopManager查看数据库
  - 

  -

- 4、测试结束