https://blog.csdn.net/weixin\_39723544/article/details/80743074

**SpringBoot之整合Redis分析和实现-基于Spring Boot2.0.2版本**

2018年06月20日 10:50:13 [gyoomi](https://me.csdn.net/weixin_39723544) 阅读数：9881 标签： [spring boot](http://so.csdn.net/so/search/s.do?q=spring%20boot&t=blog)[redis](http://so.csdn.net/so/search/s.do?q=redis&t=blog)[jedis](http://so.csdn.net/so/search/s.do?q=jedis&t=blog)[2.0](http://so.csdn.net/so/search/s.do?q=2.0&t=blog) 更多

个人分类： [Spring Boot](https://blog.csdn.net/weixin_39723544/article/category/7240211)

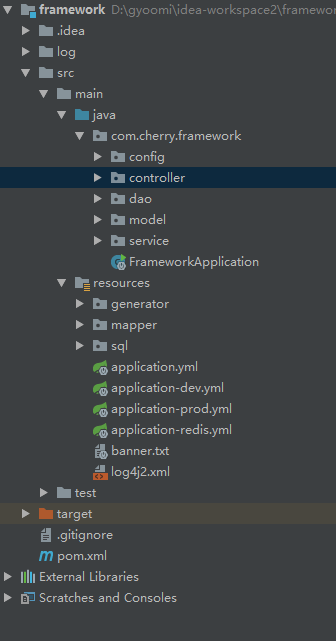
所属专栏： [Spring Boot系列](https://blog.csdn.net/column/details/27618.html)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/weixin\_39723544/article/details/80743074

### 背景介绍

* 公司最近的新项目在进行技术框架升级，基于的Spring Boot的版本是2.0.2，整合Redis数据库。网上基于2.X版本的整个Redis少之又少，中间踩了不少坑，特此把整合过程记录，以供小伙伴们参考。
* 本文的基于在于会搭建Spring Boot项目的基础上进行的，入门是小白的话，请自行学习相关基础知识，网上或相关书籍很多。
* 由于我本人对Maven比较熟悉，所以是以Maven进行的。Gradle类似，核心思想都是一样的，实现项目管理工具不同而已。

### 整合过程

* 创建Spring Boot项目（2.0.2版本）  
  利用idea提供的接口进行创建，创建后目录结构如下：(请忽略mybatis.log4j2等相关代码和文件)  
  
* pom依赖

<!-- Spring Boot Redis依赖 -->

<!-- 注意：1.5版本的依赖和2.0的依赖不一样，注意看哦 1.5我记得名字里面应该没有“data”, 2.0必须是“spring-boot-starter-data-redis” 这个才行-->

<dependency>

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

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

<!-- 1.5的版本默认采用的连接池技术是jedis 2.0以上版本默认连接池是lettuce, 在这里采用jedis，所以需要排除lettuce的jar -->

<exclusions>

<exclusion>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

</exclusion>

<exclusion>

<groupId>io.lettuce</groupId>

<artifactId>lettuce-core</artifactId>

</exclusion>

</exclusions>

</dependency>

<!-- 添加jedis客户端 -->

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

</dependency>

<!--spring2.0集成redis所需common-pool2-->

<!-- 必须加上，jedis依赖此 -->

<!-- spring boot 2.0 的操作手册有标注 大家可以去看看 地址是：https://docs.spring.io/spring-boot/docs/2.0.3.RELEASE/reference/htmlsingle/-->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-pool2</artifactId>

<version>2.5.0</version>

</dependency>

<!-- 将作为Redis对象序列化器 -->

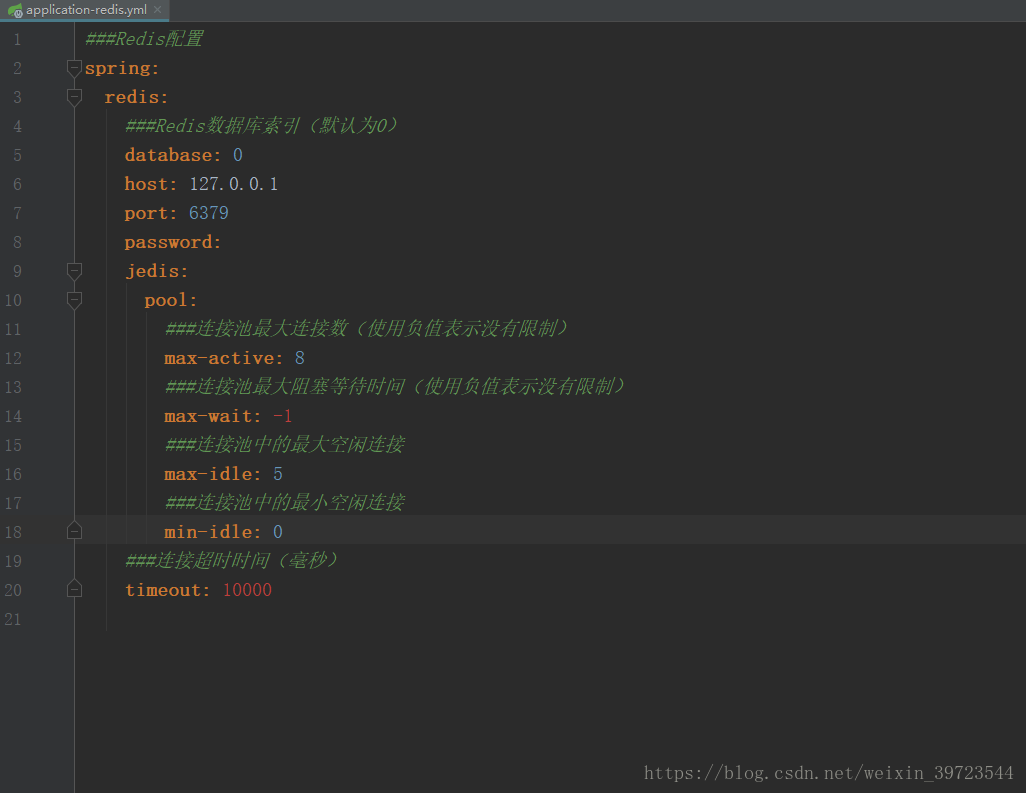
<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.47</version>

</dependency>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* yml相关Redis配置
  + 其实关于Redis的配置主要包括两方面，一时Redis的配置，一个是jedis pool连接池的配置
  + 具体配置如下  
    
* Redis自定义配置
  + 关于Redis的配置方式有很多，我知道有1.自动配置；2.手动配置；3.传统的xml文件也是可以的。在这里我只讲第2中，第二种比较灵活，符合spring boot风格。
  + 新建config包
  + 新建RedisConfiguration类
* package com.cherry.framework.config;
* import com.fasterxml.jackson.annotation.JsonAutoDetect;
* import com.fasterxml.jackson.annotation.PropertyAccessor;
* import com.fasterxml.jackson.databind.ObjectMapper;
* import org.slf4j.Logger;
* import org.slf4j.LoggerFactory;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.beans.factory.annotation.Value;
* import org.springframework.boot.context.properties.ConfigurationProperties;
* import org.springframework.cache.Cache;
* import org.springframework.cache.CacheManager;
* import org.springframework.cache.annotation.CachingConfigurerSupport;
* import org.springframework.cache.annotation.EnableCaching;
* import org.springframework.cache.interceptor.CacheErrorHandler;
* import org.springframework.cache.interceptor.KeyGenerator;
* import org.springframework.context.annotation.Bean;
* import org.springframework.context.annotation.Configuration;
* import org.springframework.data.redis.cache.RedisCacheManager;
* import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;
* import org.springframework.data.redis.core.RedisTemplate;
* import org.springframework.data.redis.serializer.Jackson2JsonRedisSerializer;
* import org.springframework.data.redis.serializer.RedisSerializer;
* import org.springframework.data.redis.serializer.StringRedisSerializer;
* import redis.clients.jedis.JedisPool;
* import redis.clients.jedis.JedisPoolConfig;
* /\*\*
* \* Redis 配置类
* \*
* \* @author Leon
* \* @version 2018/6/17 17:46
* \*/
* @Configuration
* // 必须加，使配置生效
* @EnableCaching
* public class RedisConfiguration extends CachingConfigurerSupport {
* /\*\*
* \* Logger
* \*/
* private static final Logger lg = LoggerFactory.getLogger(RedisConfiguration.class);
* @Autowired
* private JedisConnectionFactory jedisConnectionFactory;
* @Bean
* @Override
* public KeyGenerator keyGenerator() {
* // 设置自动key的生成规则，配置spring boot的注解，进行方法级别的缓存
* // 使用：进行分割，可以很多显示出层级关系
* // 这里其实就是new了一个KeyGenerator对象，只是这是lambda表达式的写法，我感觉很好用，大家感兴趣可以去了解下
* return (target, method, params) -> {
* StringBuilder sb = new StringBuilder();
* sb.append(target.getClass().getName());
* sb.append(":");
* sb.append(method.getName());
* for (Object obj : params) {
* sb.append(":" + String.valueOf(obj));
* }
* String rsToUse = String.valueOf(sb);
* lg.info("自动生成Redis Key -> [{}]", rsToUse);
* return rsToUse;
* };
* }
* @Bean
* @Override
* public CacheManager cacheManager() {
* // 初始化缓存管理器，在这里我们可以缓存的整体过期时间什么的，我这里默认没有配置
* lg.info("初始化 -> [{}]", "CacheManager RedisCacheManager Start");
* RedisCacheManager.RedisCacheManagerBuilder builder = RedisCacheManager
* .RedisCacheManagerBuilder
* .fromConnectionFactory(jedisConnectionFactory);
* return builder.build();
* }
* @Bean
* public RedisTemplate<String, Object> redisTemplate(JedisConnectionFactory jedisConnectionFactory ) {
* //设置序列化
* Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson2JsonRedisSerializer(Object.class);
* ObjectMapper om = new ObjectMapper();
* om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
* om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON\_FINAL);
* jackson2JsonRedisSerializer.setObjectMapper(om);
* // 配置redisTemplate
* RedisTemplate<String, Object> redisTemplate = new RedisTemplate<String, Object>();
* redisTemplate.setConnectionFactory(jedisConnectionFactory);
* RedisSerializer stringSerializer = new StringRedisSerializer();
* redisTemplate.setKeySerializer(stringSerializer); // key序列化
* redisTemplate.setValueSerializer(jackson2JsonRedisSerializer); // value序列化
* redisTemplate.setHashKeySerializer(stringSerializer); // Hash key序列化
* redisTemplate.setHashValueSerializer(jackson2JsonRedisSerializer); // Hash value序列化
* redisTemplate.afterPropertiesSet();
* return redisTemplate;
* }
* @Override
* @Bean
* public CacheErrorHandler errorHandler() {
* // 异常处理，当Redis发生异常时，打印日志，但是程序正常走
* lg.info("初始化 -> [{}]", "Redis CacheErrorHandler");
* CacheErrorHandler cacheErrorHandler = new CacheErrorHandler() {
* @Override
* public void handleCacheGetError(RuntimeException e, Cache cache, Object key) {
* lg.error("Redis occur handleCacheGetError：key -> [{}]", key, e);
* }
* @Override
* public void handleCachePutError(RuntimeException e, Cache cache, Object key, Object value) {
* lg.error("Redis occur handleCachePutError：key -> [{}]；value -> [{}]", key, value, e);
* }
* @Override
* public void handleCacheEvictError(RuntimeException e, Cache cache, Object key) {
* lg.error("Redis occur handleCacheEvictError：key -> [{}]", key, e);
* }
* @Override
* public void handleCacheClearError(RuntimeException e, Cache cache) {
* lg.error("Redis occur handleCacheClearError：", e);
* }
* };
* return cacheErrorHandler;
* }
* /\*\*
* \* 此内部类就是把yml的配置数据，进行读取，创建JedisConnectionFactory和JedisPool，以供外部类初始化缓存管理器使用
* \* 不了解的同学可以去看@ConfigurationProperties和@Value的作用
* \*
* \*/
* @ConfigurationProperties
* class DataJedisProperties{
* @Value("${spring.redis.host}")
* private String host;
* @Value("${spring.redis.password}")
* private String password;
* @Value("${spring.redis.port}")
* private int port;
* @Value("${spring.redis.timeout}")
* private int timeout;
* @Value("${spring.redis.jedis.pool.max-idle}")
* private int maxIdle;
* @Value("${spring.redis.jedis.pool.max-wait}")
* private long maxWaitMillis;
* @Bean
* JedisConnectionFactory jedisConnectionFactory() {
* lg.info("Create JedisConnectionFactory successful");
* JedisConnectionFactory factory = new JedisConnectionFactory();
* factory.setHostName(host);
* factory.setPort(port);
* factory.setTimeout(timeout);
* factory.setPassword(password);
* return factory;
* }
* @Bean
* public JedisPool redisPoolFactory() {
* lg.info("JedisPool init successful，host -> [{}]；port -> [{}]", host, port);
* JedisPoolConfig jedisPoolConfig = new JedisPoolConfig();
* jedisPoolConfig.setMaxIdle(maxIdle);
* jedisPoolConfig.setMaxWaitMillis(maxWaitMillis);
* JedisPool jedisPool = new JedisPool(jedisPoolConfig, host, port, timeout, password);
* return jedisPool;
* }
* }
* }
  + 1
  + 2
  + 3
  + 4
  + 5
  + 6
  + 7
  + 8
  + 9
  + 10
  + 11
  + 12
  + 13
  + 14
  + 15
  + 16
  + 17
  + 18
  + 19
  + 20
  + 21
  + 22
  + 23
  + 24
  + 25
  + 26
  + 27
  + 28
  + 29
  + 30
  + 31
  + 32
  + 33
  + 34
  + 35
  + 36
  + 37
  + 38
  + 39
  + 40
  + 41
  + 42
  + 43
  + 44
  + 45
  + 46
  + 47
  + 48
  + 49
  + 50
  + 51
  + 52
  + 53
  + 54
  + 55
  + 56
  + 57
  + 58
  + 59
  + 60
  + 61
  + 62
  + 63
  + 64
  + 65
  + 66
  + 67
  + 68
  + 69
  + 70
  + 71
  + 72
  + 73
  + 74
  + 75
  + 76
  + 77
  + 78
  + 79
  + 80
  + 81
  + 82
  + 83
  + 84
  + 85
  + 86
  + 87
  + 88
  + 89
  + 90
  + 91
  + 92
  + 93
  + 94
  + 95
  + 96
  + 97
  + 98
  + 99
  + 100
  + 101
  + 102
  + 103
  + 104
  + 105
  + 106
  + 107
  + 108
  + 109
  + 110
  + 111
  + 112
  + 113
  + 114
  + 115
  + 116
  + 117
  + 118
  + 119
  + 120
  + 121
  + 122
  + 123
  + 124
  + 125
  + 126
  + 127
  + 128
  + 129
  + 130
  + 131
  + 132
  + 133
  + 134
  + 135
  + 136
  + 137
  + 138
  + 139
  + 140
  + 141
  + 142
  + 143
  + 144
  + 145
  + 146
  + 147
  + 148
  + 149
  + 150
  + 151
  + 152
  + 153
  + 154
  + 155
  + 156
  + 157
  + 158
  + 159
  + 160
  + 161
  + 162
  + 163
  + 164
  + 165
  + 166
  + 167
  + 168
  + 169
  + 170
* 整合测试
  + 在UserService的实现类中（业务层）进行缓存测试，注入RedisTemplate或StringRedisTemplate都可以

package com.cherry.framework.service.impl;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.cherry.framework.dao.UserEntityMapper;

import com.cherry.framework.model.UserEntity;

import com.cherry.framework.service.UserService;

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

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

import org.springframework.data.redis.core.RedisTemplate;

import org.springframework.data.redis.core.StringRedisTemplate;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

/\*\*

\* User ServiceImpl

\*

\* @author Leon

\* @version 2018/6/14 17:12

\*/

@Service

public class UserServiceImpl implements UserService {

@Autowired

UserEntityMapper userEntityMapper;

@Autowired

RedisTemplate redisTemplate;

@Autowired

StringRedisTemplate stringRedisTemplate;

/\*\*

\* 新增

\*

\* @param userEntity

\* @return

\*/

@Override

@Transactional

public int save(UserEntity userEntity) {

userEntityMapper.insert(userEntity);

return userEntity.getUserId();

}

/\*\*

\* 查询所有

\*

\* @return

\*/

@Override

public PageInfo<UserEntity> findAllUserList(int pageNum, int pageSize) {

PageHelper.startPage(pageNum, pageSize);

List<UserEntity> list = userEntityMapper.selectAll();

PageInfo<UserEntity> pageInfo = new PageInfo<>(list);

// 具体使用

redisTemplate.opsForList().leftPush("user:list", JSON.toJSONString(list));

stringRedisTemplate.opsForValue().set("user:name", "张三");

return pageInfo;

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 使用postman访问对象controller的url

/\*\*

\* 列表查询

\*

\* @return

\*/

@RequestMapping(value = "/user/list")

public PageInfo<UserEntity> findUserList(int pageNum, int pageSize) {

PageInfo<UserEntity> pageInfo = userService.findAllUserList(pageNum, pageSize);

return pageInfo;

}

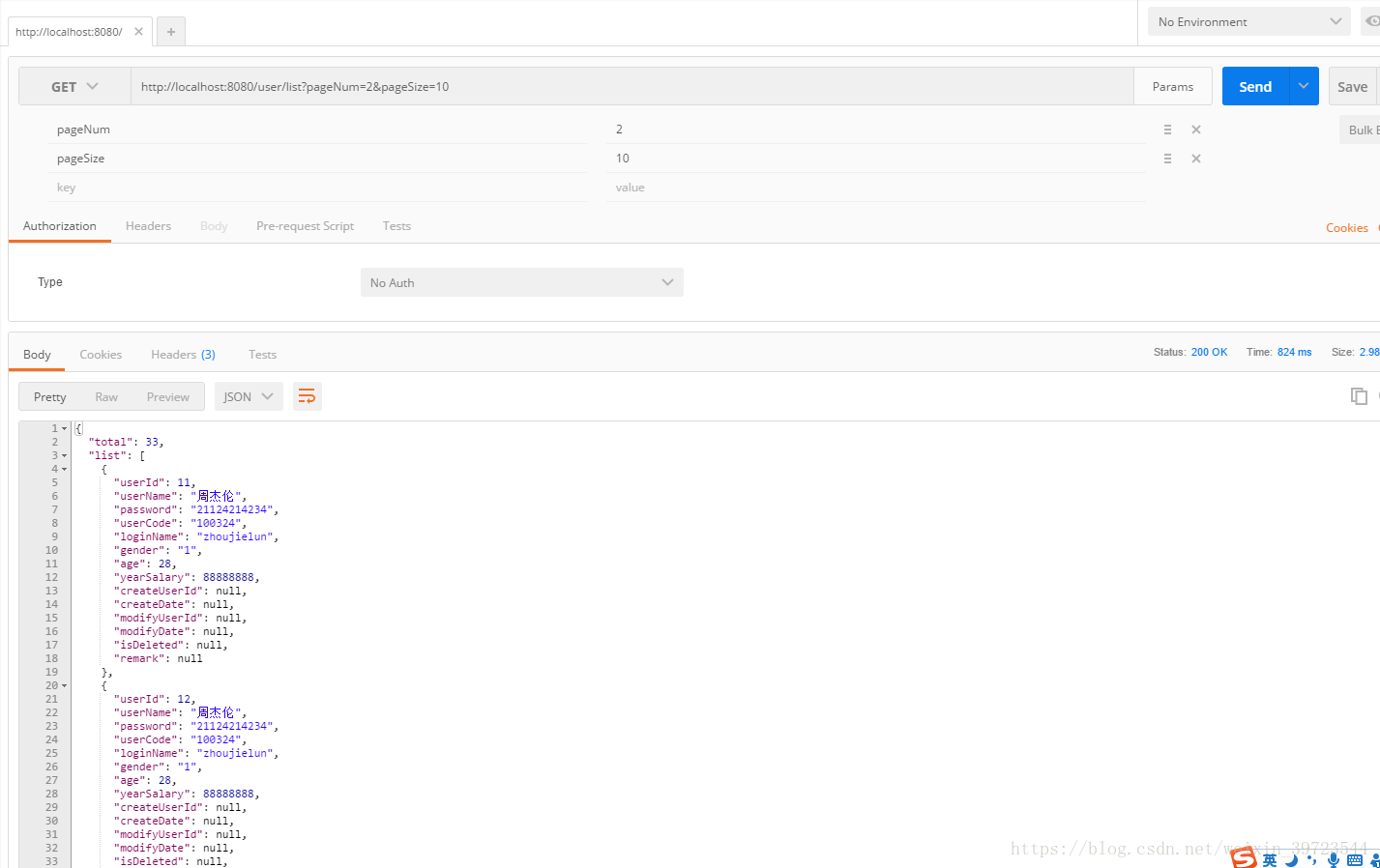
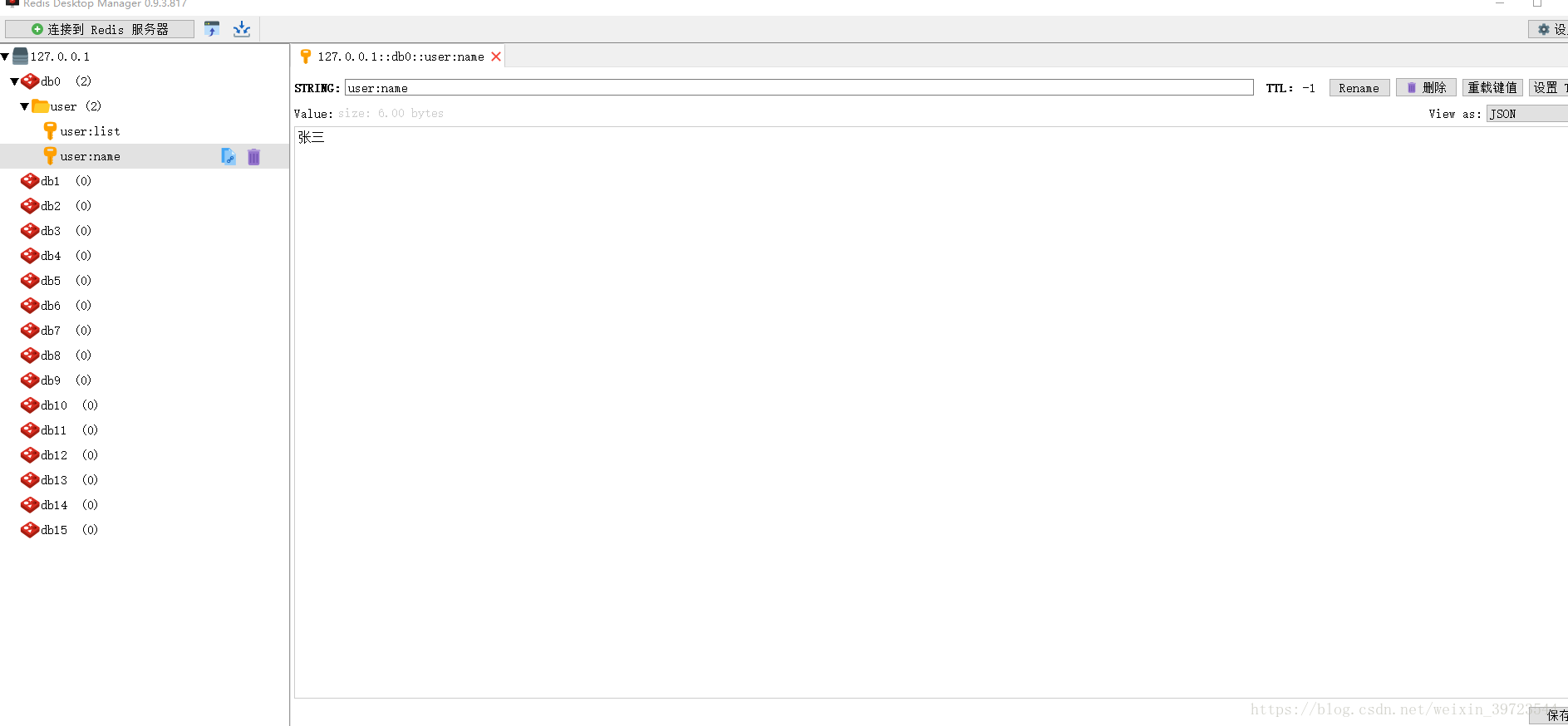
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 端口配置如下

server:

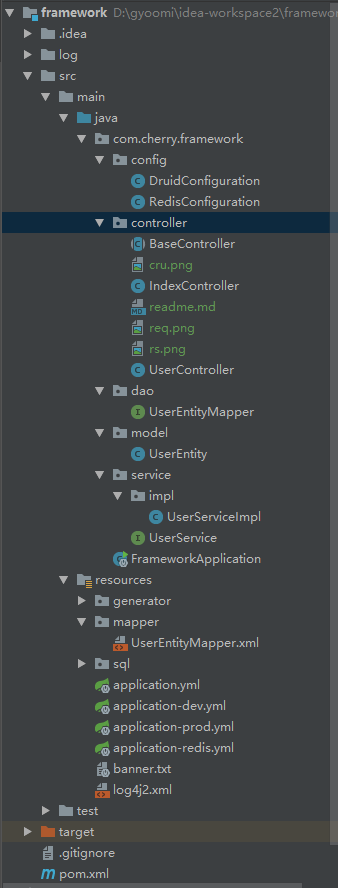
port: 8080

servlet:

path: /

* 1
* 2
* 3
* 4
* 访问url  
  
* 通过Redis Desktop Manager 进行查看  
  

### 总结

* 2.0整合在很多方面和1.5版本不一样，如果有问题参考官方英文文档，可以大大提高我们效率，（毕竟东西刚出，又没人翻译，只能看原版文档）
* 如果有说的不清楚的地方，在下面留言，我每天都会看博客，一起交流学习
* 最后晒出最后的总体结构图  
  

Github地址：[Spring Boot 2.X 整合Redis](https://github.com/gyoomi/framework)

https://github.com/gyoomi/framework