

## 前言:

上一篇关于 **Spring Cache** 的文章我们较为系统的介绍了 **Spring Cache**, 而 **Redis** 在实际项目工程中也有较为广泛的应用。

这篇文章就作为 **Spring Cache** 的一个简要补充, 或者叫做番外篇吧, 主要介绍一下 **Redis** 在 **winows** 下的安装, 以及 **Springboot** 集成 **Redis** 和一些较为简单的使用。

## Redis 介绍:

不废话, 先看官网对于 **Redis** 是怎样介绍的:

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

**Redis** 是一个开源(BSD 许可), 内存中的数据结构存储, 用作数据库、缓存和消息代理。它支持字符串、散列、列表、集合、带范围查询的排序集合、位图、超 **loglogs**、带半径查询和流的地理空间索引等数据结构。**Redis** 具有内置的复制、**Lua** 脚本、**LRU** 退出、事务和不同级别的磁盘持久性, 并通过 **Redis Sentinel** 和 **Redis** 集群自动分区提供高可用性。

大体来说呢, **Redis** 是一个开源的, 免费的, 高性能的 **Key-Value** 数据库。

**Redis** 与其他 **key - value** 缓存产品有以下三个特点:

1. **Redis** 支持数据的持久化, 可以将内存中的数据保存在磁盘中, 重启的时候可以再次加载进行使用。
2. **Redis** 不仅仅支持简单的 **key-value** 类型的数据, 同时还提供 **list**, **set**, **zset**, **hash** 等数据结构的存储。
3. **Redis** 支持数据的备份, 即 **master-slave** 模式的数据备份。

## 环境准备:

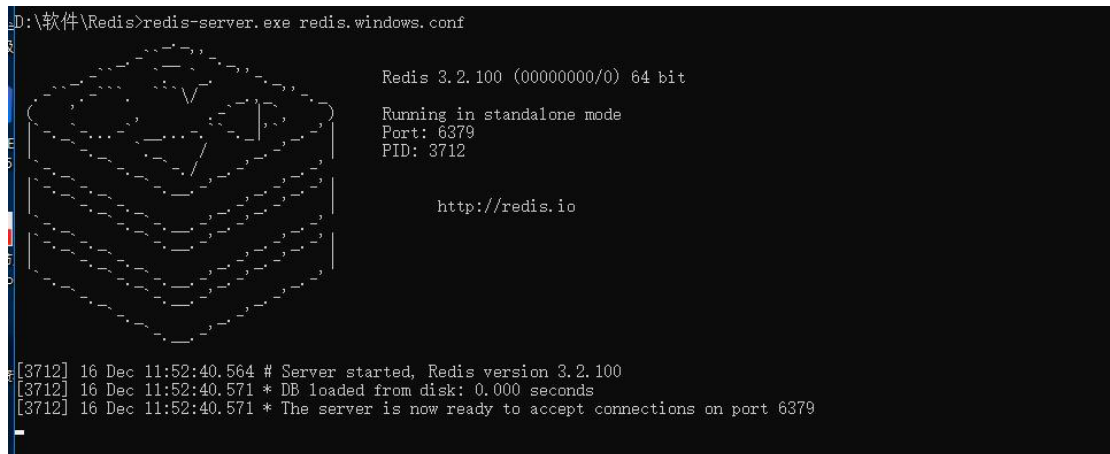
## 安装 Redis:

Windows 版下载地址 : <https://github.com/MicrosoftArchive/redis/releases>

下载之后解压到你解压的目录。

然后在该目录打开 CMD 命令行窗口。

然后执行命令: `redis-server.exe redis.windows.conf`



```
D:\软件\Redis>redis-server.exe redis.windows.conf

Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 3712

http://redis.io

[3712] 16 Dec 11:52:40.564 # Server started, Redis version 3.2.100
[3712] 16 Dec 11:52:40.571 * DB loaded from disk: 0.000 seconds
[3712] 16 Dec 11:52:40.571 * The server is now ready to accept connections on port 6379
```

出现这个我们服务端就成功启动了。

什么? 想要解锁更多姿势? , 请移步菜鸟教程或者官方文档。

## SpringBoot 集成 Redis

在 pom.xml 中添加我们集成 Redis 所需要的依赖。

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

然后在我们 Springboot 配置文件 application.properties 中配置 Redis 相关信息。

```
spring.redis.host=127.0.0.1  
//如果是本地的话，就是这个，如果是远程的话，请填远程 redis 服务器 IP  
spring.redis.port=6379  
//Redis 默认端口
```

```
spring.redis.host=127.0.0.1  
spring.redis.port=6379
```

当然配置 Redis 肯定不止这两个参数可以填，如果想要配置 `timeout` 或者其他的参数呢，按住 `Ctrl`，就会进入到 Redis 配置类了。大概长这个样子：

```
@ConfigurationProperties(  
    prefix = "spring.redis"  
)  
  
public class RedisProperties {  
    private int database = 0;  
    private String url;  
    private String host = "localhost";  
    private String password;  
    private int port = 6379;  
    private boolean ssl;  
    private Duration timeout;  
    private RedisProperties.Sentinel sentinel;  
    private RedisProperties.Cluster cluster;  
    private final RedisProperties.Jedis jedis = new RedisProperties.Jedis();  
    private final RedisProperties.Lettuce lettuce = new RedisProperties.Lettuce();  
}
```

看到这个 `prefix = "spring.redis"`  
我想天才的你肯定明白了什么，是的，就是这样。去探索吧。

当配置了 Redis 的基本参数之后，Spring 在启动的时候会自动帮我们创建好相应的 `RedisTemplate` 模板，到时候我们只需要注入就可以直接使用了。

## 代码实战:

```
@Autowired
private RedisTemplate<String, String> redisTemplate;

@RequestMapping("save")
public String redisSave(@RequestParam String key, @RequestParam String value) {

    redisTemplate.opsForValue().set(key, value);

    return key;
}

@RequestMapping("get")
public String redisGet(@RequestParam String key) {

    String value = (String) redisTemplate.opsForValue().get(key);

    return value;
}
```

## 总结:

本篇文档呢，只是简要概述了一下 Redis 的安装，Springboot 集成以及简单的使用，同时也附上了一个简单的实战。不过，Redis 的用法肯定不止 set 和 get 这两个，比如 Redis 分布式等等，还有很多更为高级的用法，不过因为我个人的能力有限，不敢贸然写 Redis 更为高级的文档，若日后深入学习，必将此文档补上。

代码环境:Springboot2.0.7.RELEASE

IDE:IDEA2018

版本:1.0

时间:2018-12-16 日

作者:韩数(代号)

邮箱:1758504262@qq.com

Github : <https://github.com/hanshuaikang>

参考资料:

Redis 官方网站: <https://redis.io/>

菜鸟教程 : <http://www.runoob.com/redis/redis-tutorial.html>

小马哥微服务实战视频 Springboot 部分

最后，给一个星星吧，我已经好几天没吃星星了。

