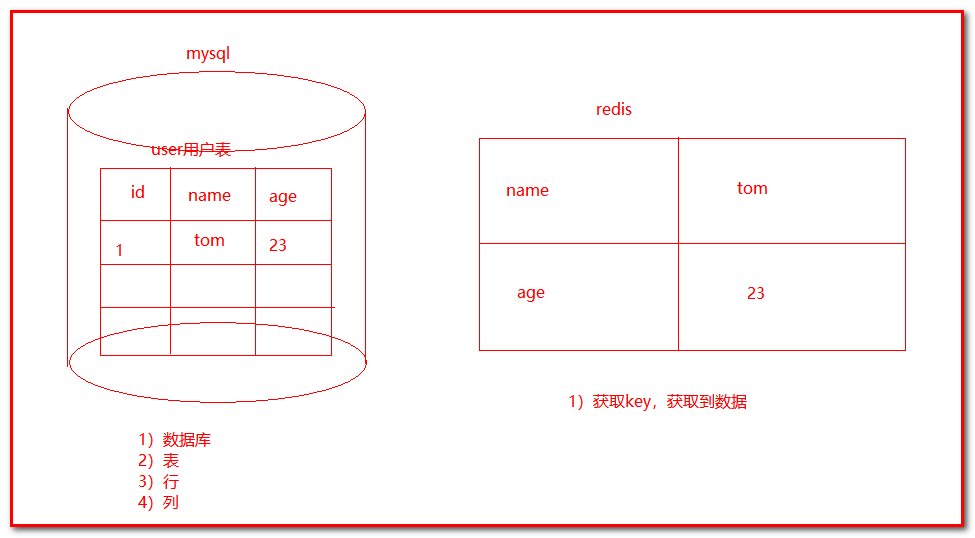
# Redis官方教程翻译和使用学习

Nosql数据库：不使用sql语句操作的数据库



为什么要学习redis？

答：开源Key-Value数据库，以前的数据库都是结构化数据（库——表——行——列），现在nosql的key-value数据库，查询的时候，只需要找到相应key，就获取值（key——value），整体的性能表现，远超以前额结构化数据库。

优势：支持多种编程语言。

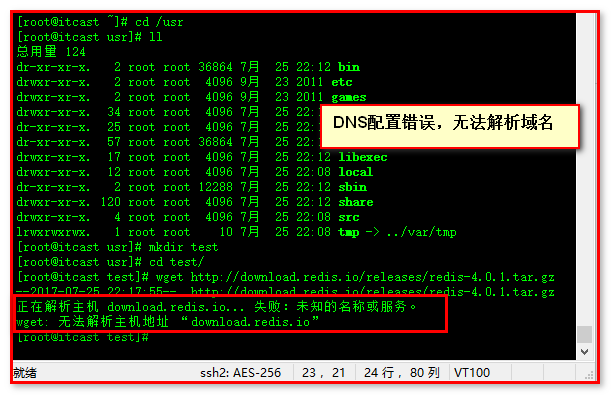
## 一、 Redis在Linux下的安装

安装路径：/usr/test/

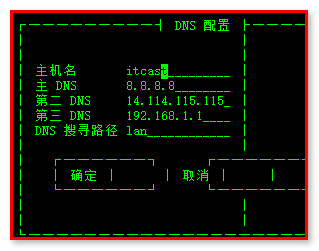
**下载：**

**wget** wget http://download.redis.io/releases/redis-4.0.1.tar.gz

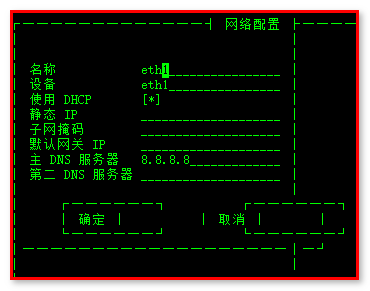
**如果：**



**修改DNS为：8.8.8.8**



**修改网络地址转换（nat）网卡dns：8.8.8.8**



**解压：**

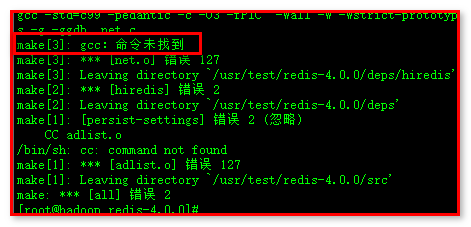
tar xzf redis-4.0.1.tar.gz

**编译：**

cd redis-4.0.1

make

**如果出现：**



**编译需要的C环境没有：yum install gcc-c++**

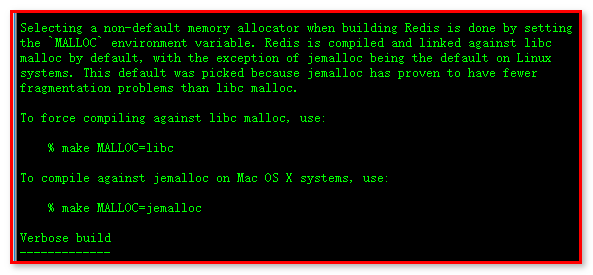
**Make编译**

**如果出现：**



**解决：make MALLOC=libc**

**参考Redis文件夹中的README.md文档：**



选择非默认的内存分配器的建设时，Redis是通过设置的` malloc `环境变量。redis是编译和链接之间的对抗malloc在默认情况下，在对Linux默认jemalloc例外系统。这是因为默认jemalloc证明有少碎裂的问题比libc malloc

人话：linux 没有jemalloc这个内存分配器， 使用make命令同时可以指定内存分配器：malloc=libc



**编译后的二进制文件可以在src目录下使用：**

**运行redis:**

**src/redis-server**

**使用redis客户端连接redis:**

**src/redis-cli**

**测试使用：**

**redis> set foo bar**

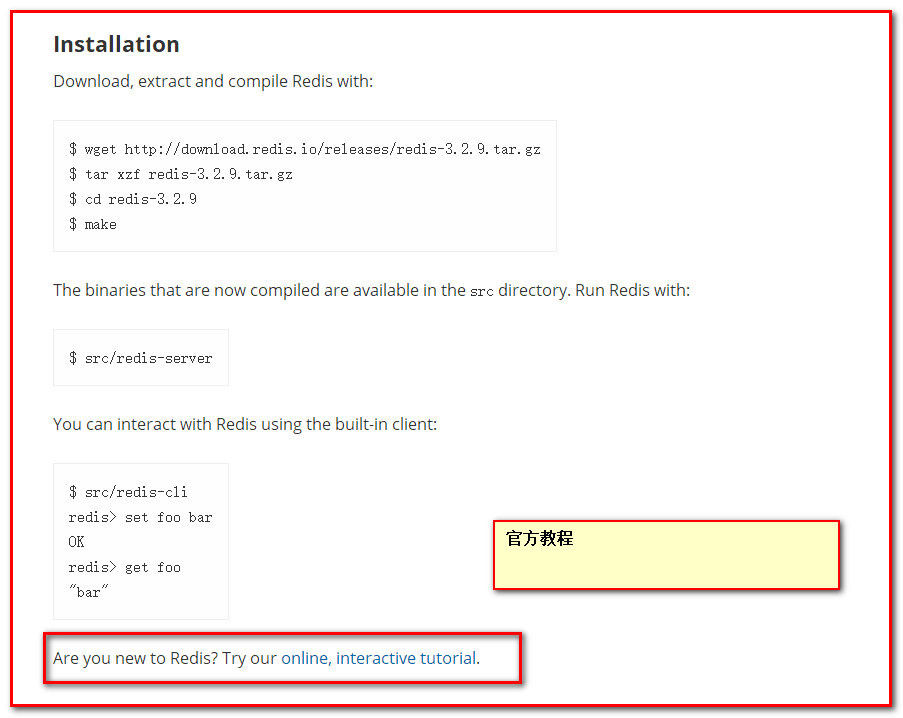
**OK**

**redis> get foo**

**"bar"**

## 二、 Redis快速入门

官方教学：

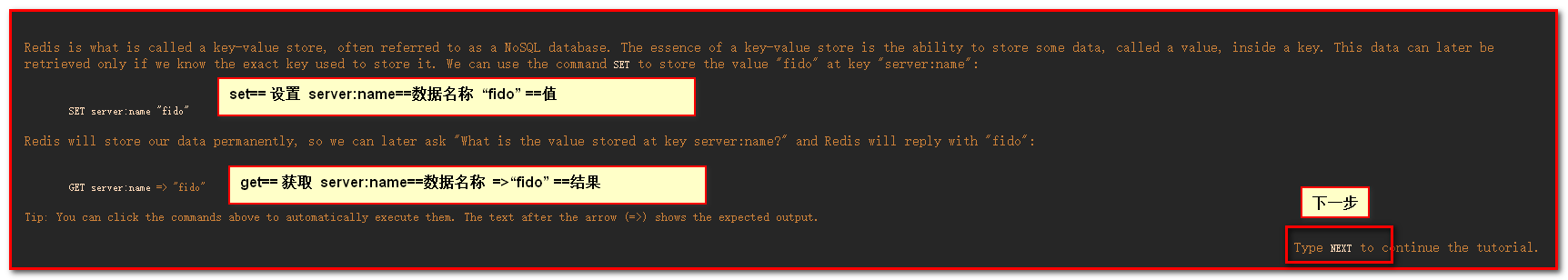


教学界面：



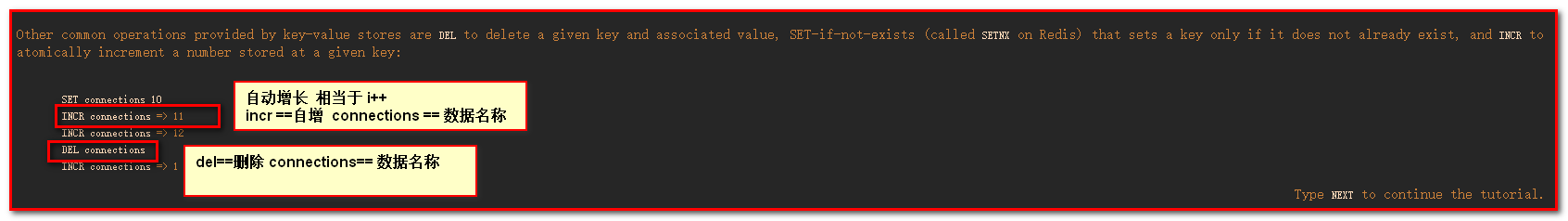
### 第一个教学：

设置简单的键值对数据，set命令 server:name 数据的名称（key） fido 数据具体的值

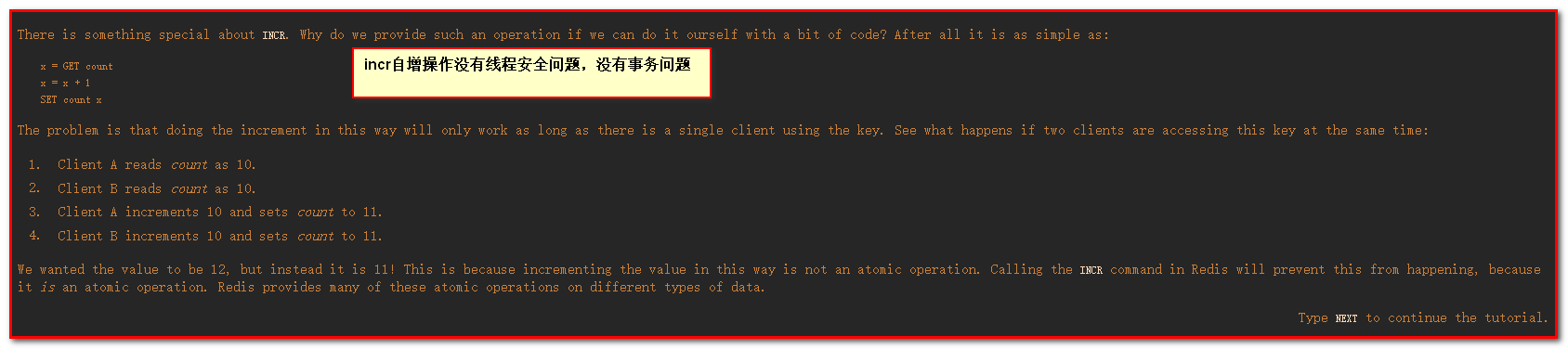


### 第二个教学：

Incr 相当于一个自增长的操作，每次执行加一。 Del 删除数据 ，数据删除了，再次查询：nil——表示空



### 第三个教学：



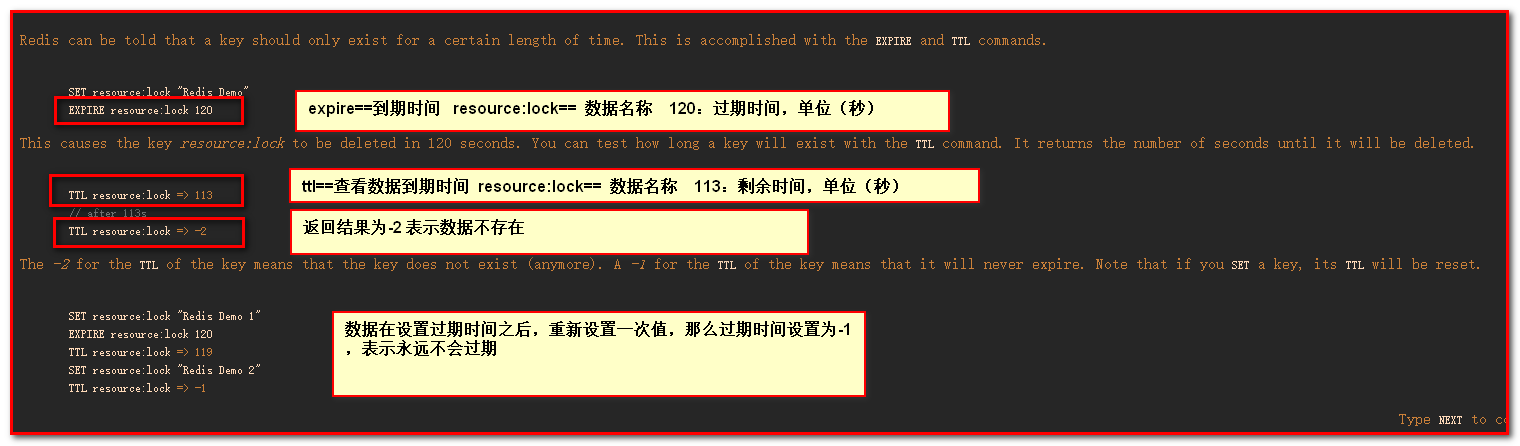
### 第四个教学：

Expire: 设置数据的到期时间 单位是秒 （相当于cookie的生存时间）

ttl ：获取当前数据还可以存活多少时间

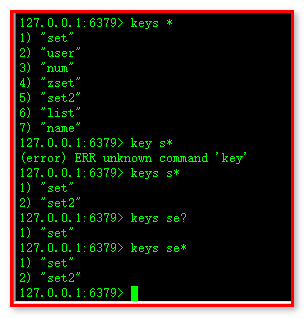
如果倒计时结束，数据就删除了，使用ttl返回的结果是-2

如果当前设置好倒计时之后，再次设置数据，那么，会清除倒计时的效果，ttl命令返回值：-1

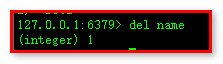


### keys的通用操作

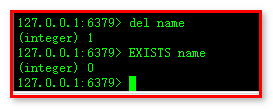
* keys pattern：获取所有与pattern匹配的key，返回所有与该key匹配的keys。\*表示任意一个或多个字符，?表示任意一个字符



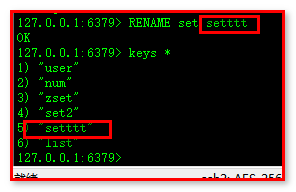
* del key1 key2…：删除指定的key



* exists key：判断该key是否存在，1代表存在，0代表不存在



* rename key newkey：为当前的key重命名



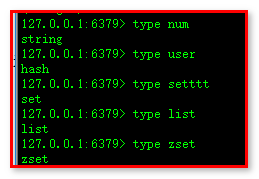
* expire key ：设置过期时间，单位：秒



* ttl key：获取该key所剩的超时时间，如果没有设置超时，返回-1。如果返回-2表示超时不存在。



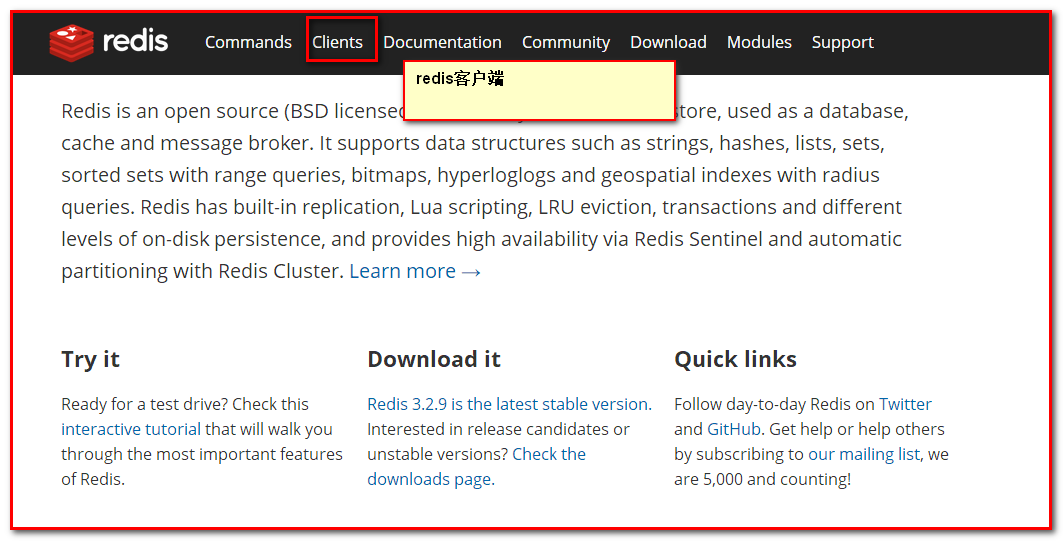
* type key：获取指定key的数据类型。该命令将以字符串的格式返回。 返回的字符串为string、list、set、hash和zset，如果key不存在返回none。

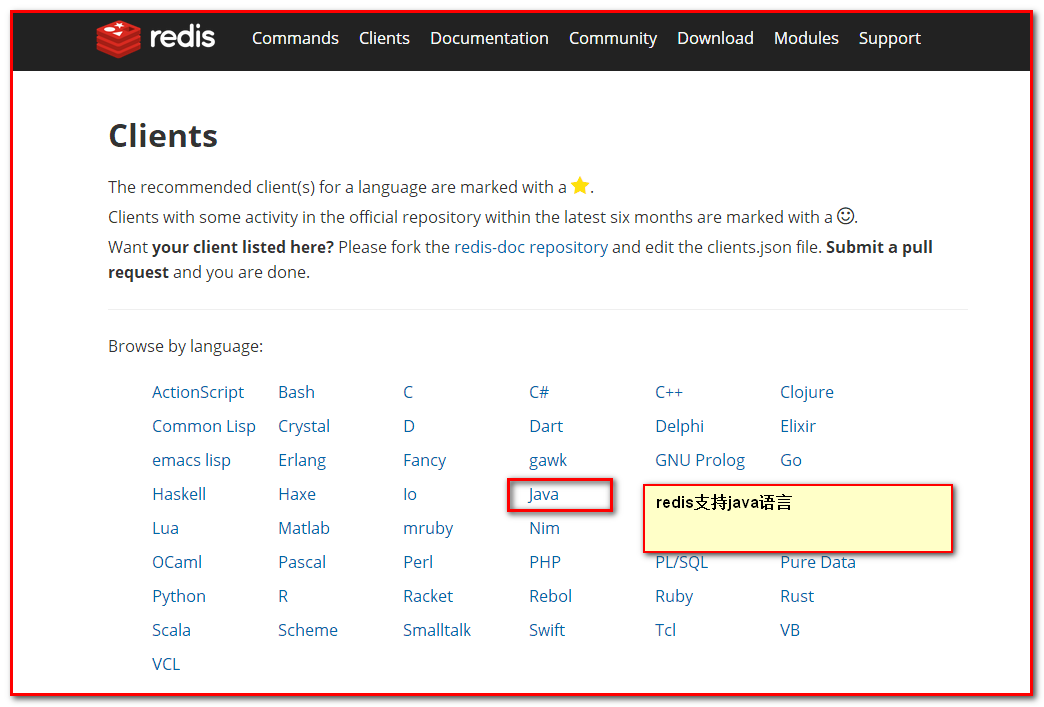


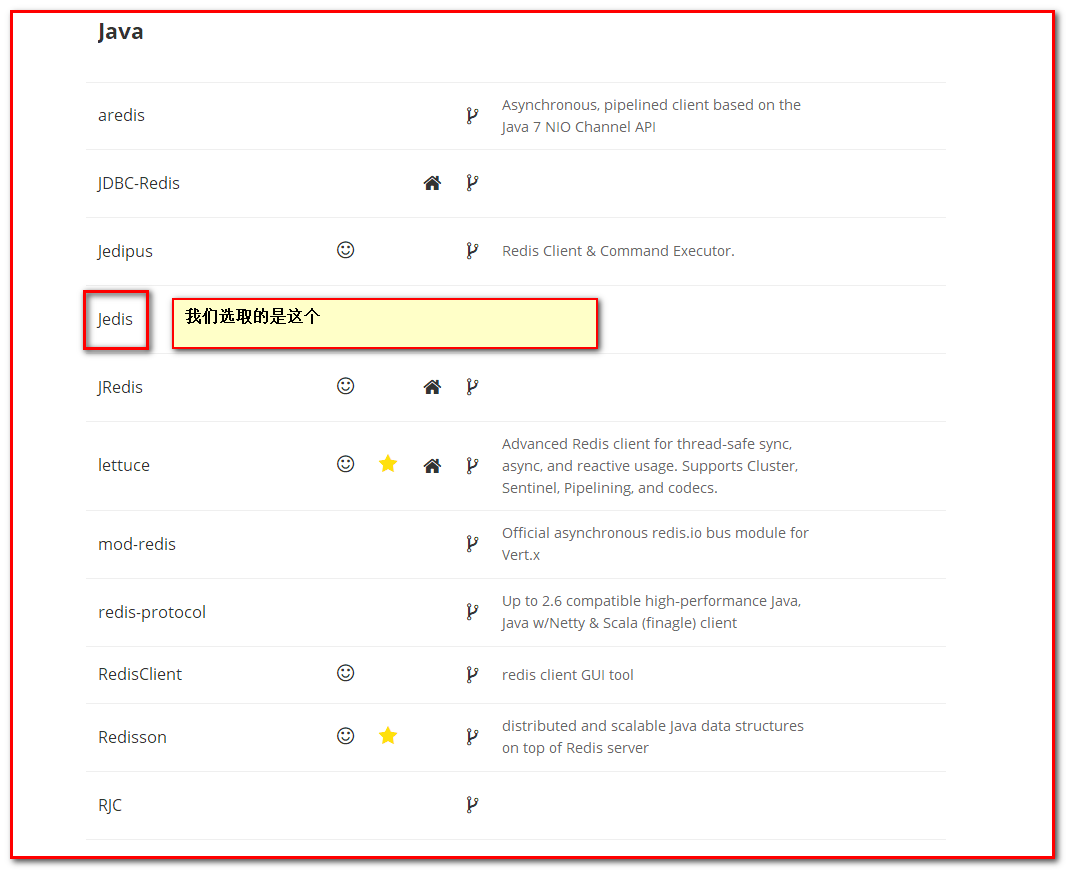
## 三、 使用jedis操作redis（使用java技术操作redis）

### Jedis介绍和下载

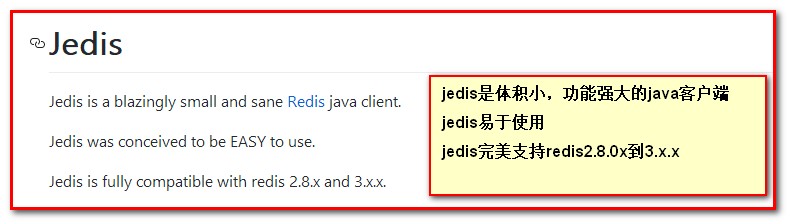
学会了简单使用redis之后，我们需要和以前学习mysql一样学习使用java的方式操作redis数据库







进入github获取资源：



Jedis可以为我们做什么？

以下所有redis支持的功能：

排序

连接处理

操作任何类型的值的命令

字符串值操作命令

命令操作的哈希值

在列表上操作的命令

在集合上操作的命令

在排序集上操作的命令

处理( transaction的名词复数 )

流水线操作

发布/订阅

坚持控制命令

远程服务器控制命令

连接池

Sharding（MD5，murmurhash）

切分关键标签

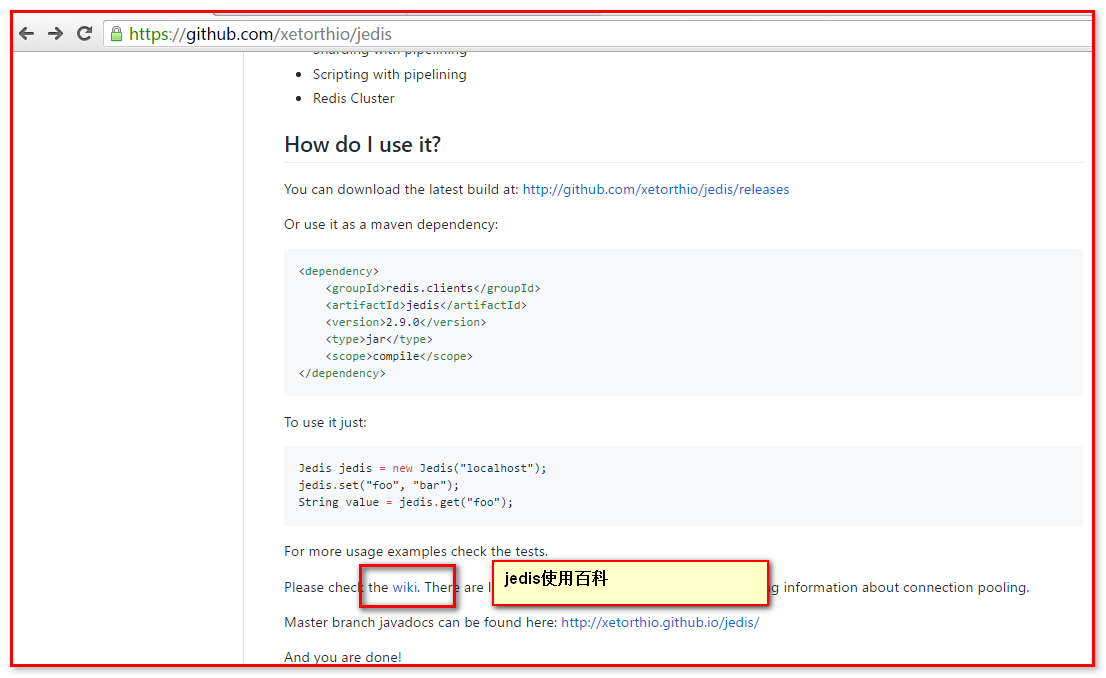
分片的流水线

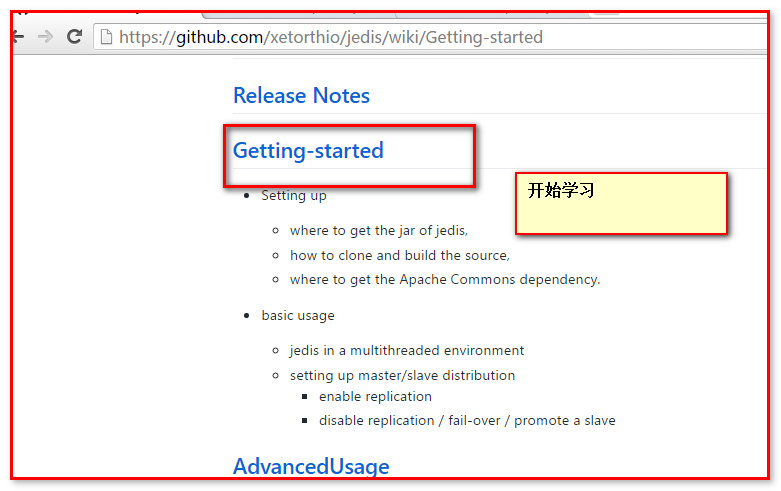
脚本的流水线

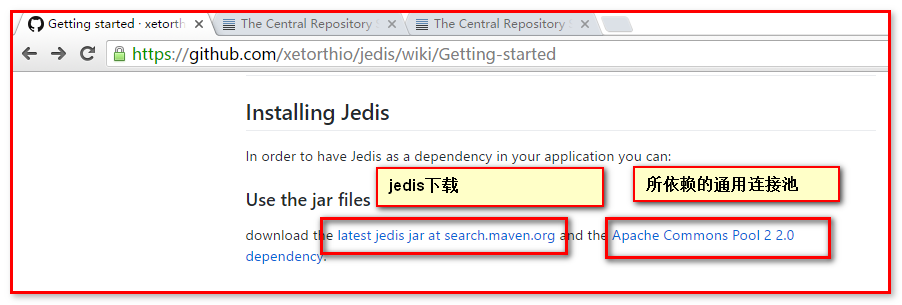
Redis集群

如何使用jedis?

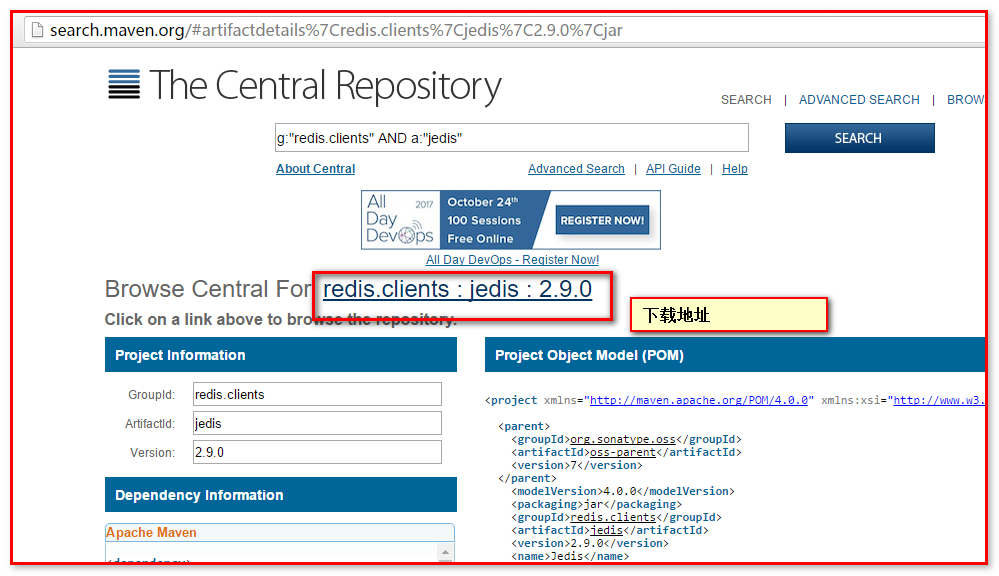
1 下载资源





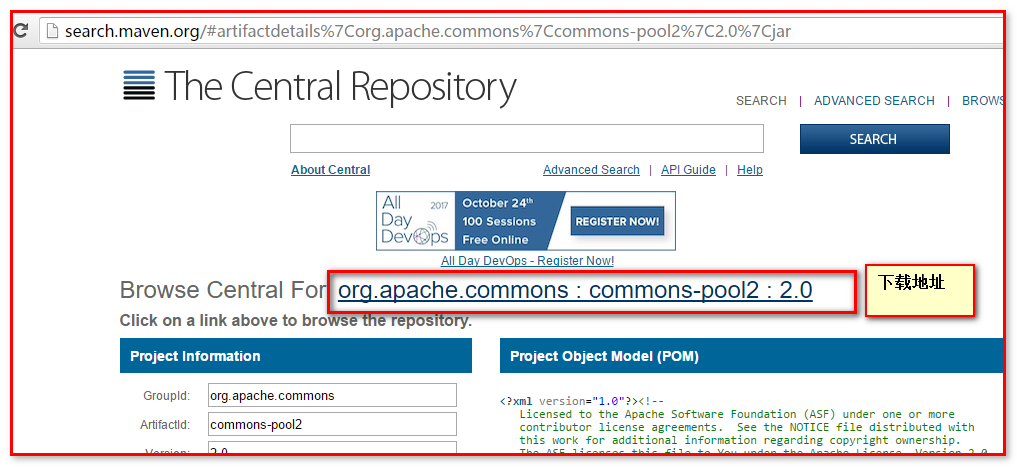






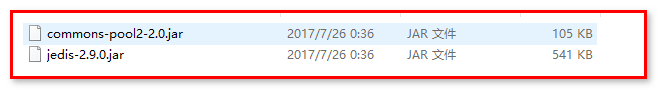


另一个jar下载：



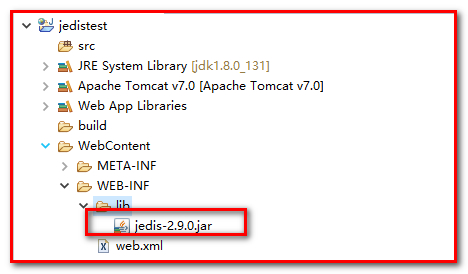


下载后：



### jedis快速入门：

1 创建工程导入jar包



2 创建测试类

|  |
| --- |
| **package** cn.itcast.test;  **import** org.junit.Test;  **import** redis.clients.jedis.Jedis;  **public** **class** TestUtils {  @Test  **public** **void** test1() {    Jedis jedis = **new** Jedis("localhost");  jedis.set("foo", "bar");  String value = jedis.get("foo");  }  } |

结果：没有本地的redis，所以发生异常

|  |
| --- |
| redis.clients.jedis.exceptions.JedisConnectionException: java.net.ConnectException: Connection refused: connect  at redis.clients.jedis.Connection.connect(Connection.java:207)  at redis.clients.jedis.BinaryClient.connect(BinaryClient.java:93)  at redis.clients.jedis.Connection.sendCommand(Connection.java:126)  at redis.clients.jedis.BinaryClient.set(BinaryClient.java:110)  at redis.clients.jedis.Client.set(Client.java:47)  at redis.clients.jedis.Jedis.set(Jedis.java:120)  at cn.itcast.test.TestUtils.test1(TestUtils.java:12)  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  at java.lang.reflect.Method.invoke(Method.java:498)  at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)  at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)  at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)  at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)  at org.junit.runners.ParentRunner.runLeaf(ParentRunner.java:325)  at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:78)  at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:57)  at org.junit.runners.ParentRunner$3.run(ParentRunner.java:290)  at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:71)  at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:288)  at org.junit.runners.ParentRunner.access$000(ParentRunner.java:58)  at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:268)  at org.junit.runners.ParentRunner.run(ParentRunner.java:363)  at org.eclipse.jdt.internal.junit4.runner.JUnit4TestReference.run(JUnit4TestReference.java:86)  at org.eclipse.jdt.internal.junit.runner.TestExecution.run(TestExecution.java:38)  at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:459)  at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:678)  at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.run(RemoteTestRunner.java:382)  at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.main(RemoteTestRunner.java:192)  Caused by: java.net.ConnectException: Connection refused: connect  at java.net.DualStackPlainSocketImpl.waitForConnect(Native Method)  at java.net.DualStackPlainSocketImpl.socketConnect(DualStackPlainSocketImpl.java:85)  at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)  at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)  at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)  at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:172)  at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)  at java.net.Socket.connect(Socket.java:589)  at redis.clients.jedis.Connection.connect(Connection.java:184)  ... 29 more |

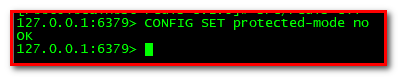
再次测试：结果为：redis目前处于受保护模式，不允许非本地客户端链接，我们可以通过给关闭保护模式解决

|  |
| --- |
| redis.clients.jedis.exceptions.JedisDataException: DENIED Redis is running in protected mode because protected mode is enabled, no bind address was specified, no authentication password is requested to clients. In this mode connections are only accepted from the loopback interface. If you want to connect from external computers to Redis you may adopt one of the following solutions: 1) Just disable protected mode sending the command 'CONFIG SET protected-mode no' from the loopback interface by connecting to Redis from the same host the server is running, however MAKE SURE Redis is not publicly accessible from internet if you do so. Use CONFIG REWRITE to make this change permanent. 2) Alternatively you can just disable the protected mode by editing the Redis configuration file, and setting the protected mode option to 'no', and then restarting the server. 3) If you started the server manually just for testing, restart it with the '--protected-mode no' option. 4) Setup a bind address or an authentication password. NOTE: You only need to do one of the above things in order for the server to start accepting connections from the outside.  at redis.clients.jedis.Protocol.processError(Protocol.java:127)  at redis.clients.jedis.Protocol.process(Protocol.java:161)  at redis.clients.jedis.Protocol.read(Protocol.java:215)  at redis.clients.jedis.Connection.readProtocolWithCheckingBroken(Connection.java:340)  at redis.clients.jedis.Connection.getStatusCodeReply(Connection.java:239)  at redis.clients.jedis.Jedis.set(Jedis.java:121)  at cn.itcast.test.TestUtils.test3(TestUtils.java:33)  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  at java.lang.reflect.Method.invoke(Method.java:498)  at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)  at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)  at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)  at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)  at org.junit.runners.ParentRunner.runLeaf(ParentRunner.java:325)  at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:78)  at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:57)  at org.junit.runners.ParentRunner$3.run(ParentRunner.java:290)  at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:71)  at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:288)  at org.junit.runners.ParentRunner.access$000(ParentRunner.java:58)  at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:268)  at org.junit.runners.ParentRunner.run(ParentRunner.java:363)  at org.eclipse.jdt.internal.junit4.runner.JUnit4TestReference.run(JUnit4TestReference.java:86)  at org.eclipse.jdt.internal.junit.runner.TestExecution.run(TestExecution.java:38)  at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:459)  at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:678)  at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.run(RemoteTestRunner.java:382)  at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.main(RemoteTestRunner.java:192) |

错误信息内容的中文翻译：

|  |
| --- |
| 1. 发送禁用保护模式的命令：“CONFIG SET protected-mode no”       2）或者你可以仅仅通过编辑redis配置文件禁用保护模式，并设置保护模式选择“否”，然后重新启动服务器。      重新启动redis，并加载配置文件：    效果：还是获取不到链接，原因是redis绑定默认访问端口为127.0.0.1：    重启redis，测试：    3）如果您只是为了测试而手动启动服务器，则用“--protected-mode no”选项重新启动它。  命令：src/redis-server --protected-mode no    测试：    4）设置绑定地址或身份验证密码。  不改变保护模式：    修改绑定地址IP：    重启redis，并加载配置文件：    测试：    注意：您只需要做上面的其中一件事情，以便服务器开始接受来自外部的连接。 |

关闭redis保护模式：CONFIG SET protected-mode no



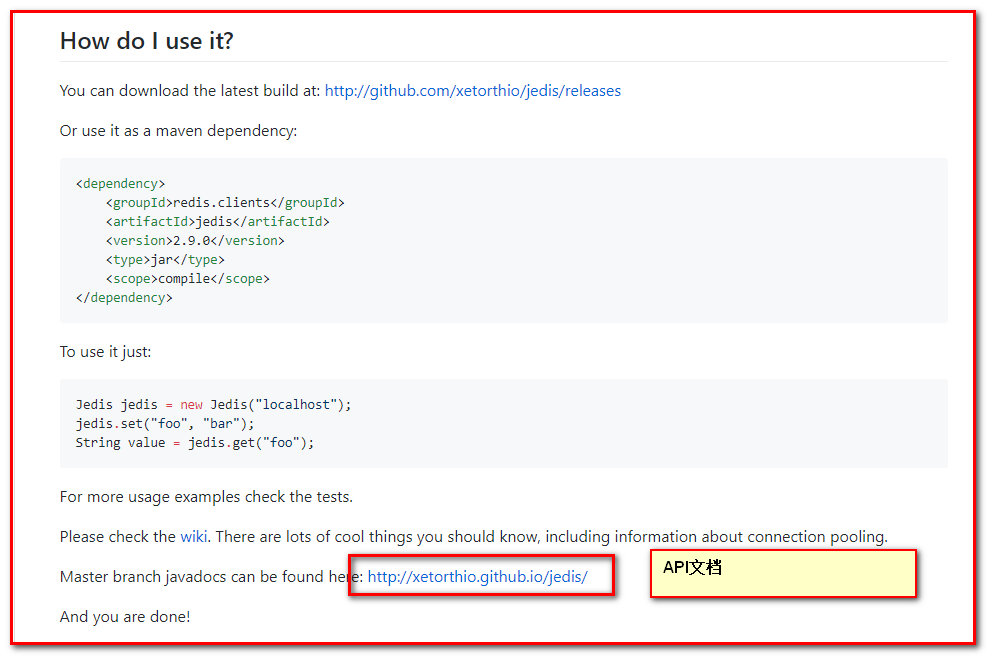
再试测试：

|  |
| --- |
| @Test  **public** **void** test5() **throws** Exception{  Jedis jedis = **new** Jedis("192.168.56.101", 6379);  jedis.set("hhh", "哈哈");  System.***out***.println(jedis.get("hhh"));  } |

结果输出：哈哈

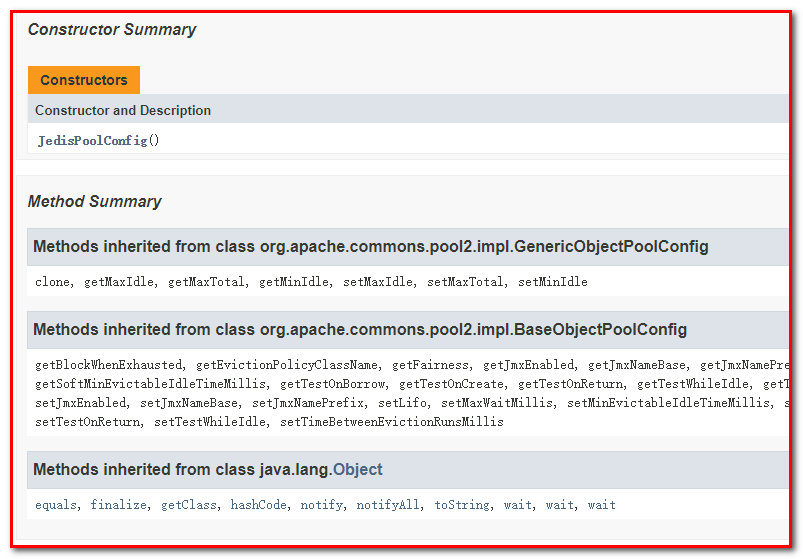
### 核心对象学习：（Jedis JedisPoolConfig）

文档位置：

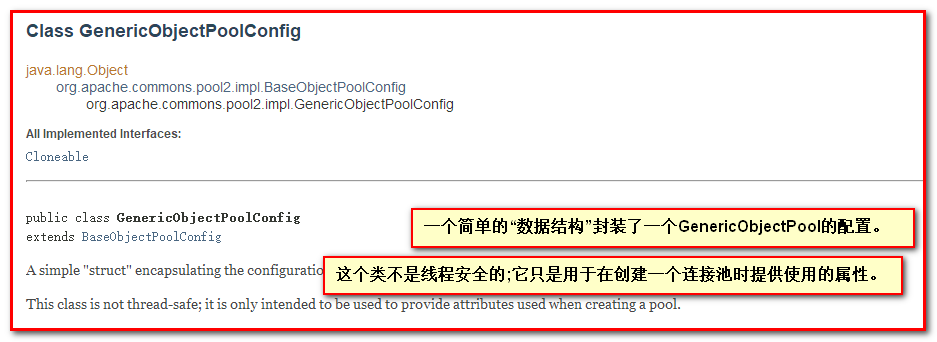


#### JedisPoolConfig：redis连接池配置对象





父类：GenericObjectPoolConfig





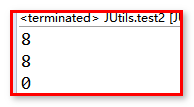
查看默认数据：

System.***out***.println(jedisPoolConfig.getMaxIdle());

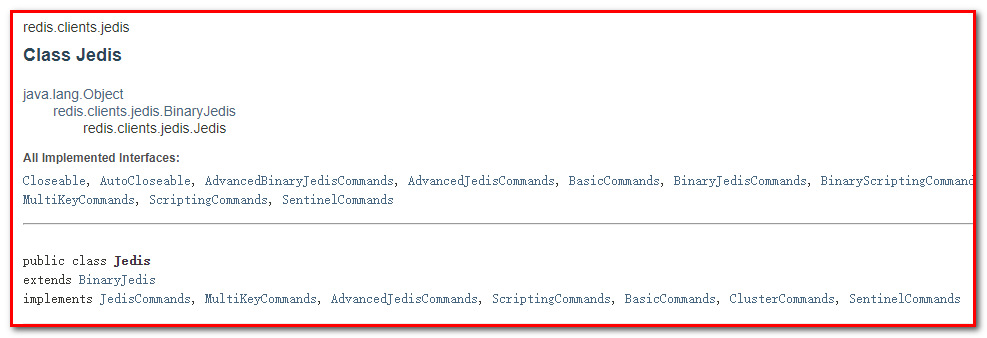
System.***out***.println(jedisPoolConfig.getMaxTotal());

System.***out***.println(jedisPoolConfig.getMinIdle());

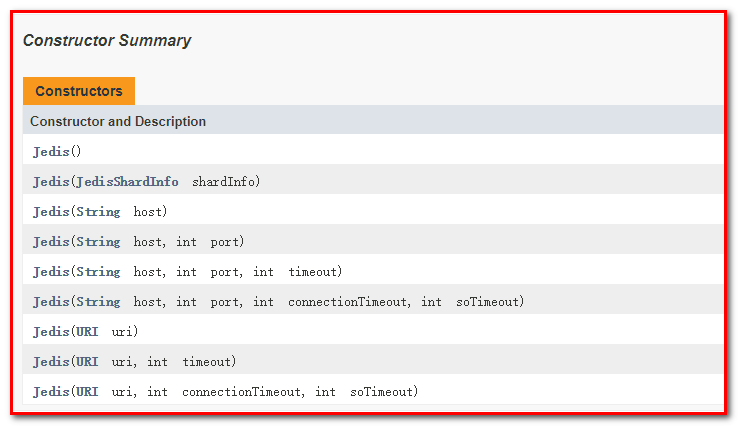
结果：



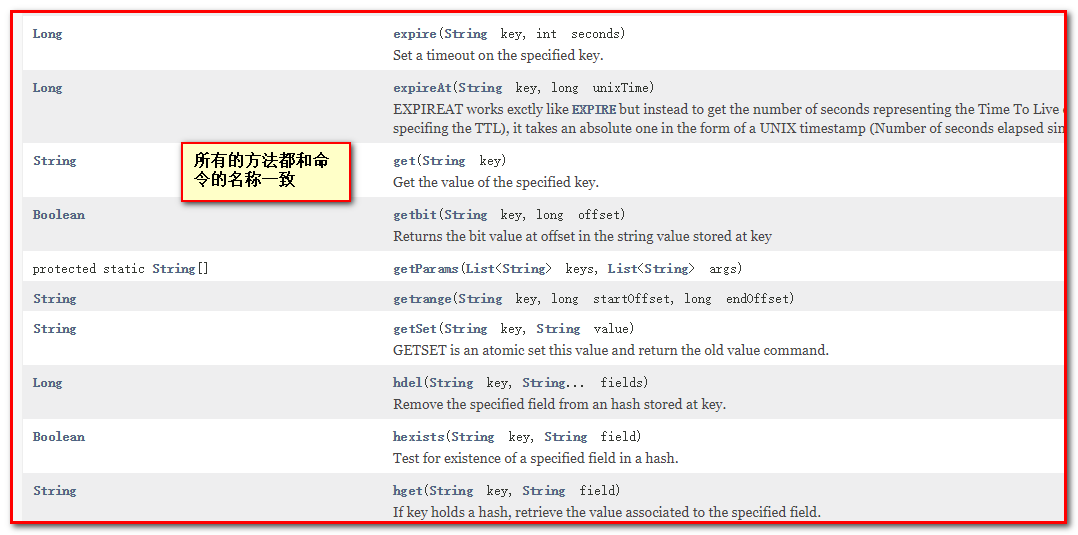
#### Jedis ：redis操作对象



构造函数：



方法：



测试代码：

|  |
| --- |
| @Test  **public** **void** test5() {  JedisPool pool = **new** JedisPool(**new** JedisPoolConfig(), "192.168.2.146",6379);  **try** (Jedis jedis = pool.getResource()) {  jedis.set("name", "zhangsan");  jedis.expire("name", 20);  }  pool.destroy();  } |

结果：

