Getting started

Installing Jedis

In order to have Jedis as a dependency in your application you can:

Use the jar files

download the latest jedis jar at search.maven.org and the Apache Commons Pool 2 2.0 dependency.

build from source

This gives you the most recent version.

Clone the github project.

That is very easy, on the commandline you just need to: `git clone git://github.com/xetorthio/jedis.git`

build

Before you package it using maven, you have to pass the tests. To run the tests and package, run `make package`.

Configure a Maven dependency

Jedis is also distributed as a Maven Dependency through Sonatype. To configure that just add the following XML snippet to your pom.xml file.

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.6.0</version>
    <type>jar</type>
    <scope>compile</scope>
</dependency>
```

Basic usage example 基本用法示例

在多线程环境中使用Jedis

using Jedis in a multithreaded environment

不应该在不同的线程中使用同一个实例，因为这会出现奇怪的错误

You shouldn't use the same instance from different threads because you'll have strange errors. And sometimes creating lots of Jedis instances is not good enough because it means lots of sockets and connections, which leads to strange errors as well. `A single Jedis instance is not threadsafe!` To avoid these problems, you should use JedisPool, which is a threadsafe pool of network connections. You can use the pool to reliably create several Jedis instances, given you return the Jedis instance to the pool when done. This way you can overcome those strange errors and achieve great performance.

有时候，创建很多Jedis实例也不好。因为这意味着存在大量的会话和连接，这也会导致奇怪的错误。

这种方式可以克服这些奇怪的问题，并实现卓越的性能！

单一Jedis实例不是线程安全的！为了避免这些问题，应该使用JedisPool，它是线程安全的网络连接线程池。
你可以使用线程池来可靠地创建多个Jedis实例，当完成后会将Jedis实例返回给这个线程池。

To use it, init a pool:

要使用它，先初始化一个连接池：

```
JedisPool pool = new JedisPool(new JedisPoolConfig(), "localhost");
```

You can store the pool somewhere statically, it is thread-safe.

PoolConfig包括许多有用的Redis方面连接池的默认值

JedisPoolConfig includes a number of helpful Redis-specific connection pooling defaults. For example, Jedis with JedisPoolConfig will close a connection after 300 seconds if it has not been returned.

socket会话读取超时时间设置

You use it by:

Jedis实现了*Closable*接口，所以*Jedis*实例在最后一条语句之后会被自动关闭。

```
/// Jedis implements Closable. Hence, the jedis instance will be auto-closed a
try (Jedis jedis = pool.getResource()) {
  /// ... do stuff here ... for example
  jedis.set("foo", "bar");
  String foobar = jedis.get("foo");
  jedis.zadd("sose", 0, "car"); jedis.zadd("sose", 0, "bike");
  Set<String> sose = jedis.zrange("sose", 0, -1);
}
/// ... when closing your application:  当关闭应用时：
pool.destroy();
```

Setting up master/slave distribution 设置"主/从"分布

不建议通过客户端程序实现！

enable replication 1. 启用复制

Redis is primarily built for master/slave distribution. This means that write requests have to be explicitly addressed to the master (a redis server), which replicates

"一主多从-读写分离"架构：这意味着，写入请求必须显示地被发送到主库(Master)，它会复制变更到从库s(Slaves)。然后，读请求可以(但不一定必须)被发送到从库，这样就减轻了主库的压力。

changes to slaves (which are also redis servers). Read requests then can be (but must not necessarily) addressed to the slaves, which alleviates the master.

You use the master as shown above. In order to enable replication, there are two ways to tell a slave it will be "slaveOf" a given master:

在Redis服务器的配置文件中指定

- Specify it in the respective section in the Redis Config file of the redis server
- on a given jedis instance (see above), call the slaveOf method and pass IP (or "localhost") and port as argument:

```
jedis.slaveOf("localhost", 6379);  //  if the master is on the same PC which r
jedis.slaveOf("192.168.1.35", 6379);
```

Note: since Redis 2.6 slaves are read only by default, so write requests to them will result in an error.

If you change that setting they will behave like normal redis servers and accept write requests without errors, but the changes won't be replicated, and hence those changes are at risk to be silently overwritten, if you mix up your jedis instances.

2. 禁用复制/提升Slave为Master
disable replication / upon failing master, promote a slave

In case your master goes down, you may want to promote a slave to be the new master. You should first (try to) disable replication of the offline master first, then, in case you have several slaves, enable replication of the remaining slaves to the new master:

自动故障转移：当主库出现故障时，可能希望提升从库为新的主库。

```
slave1jedis.slaveofNoOne();
slave2jedis.slaveOf("192.168.1.36", 6379);
```