爱旅行项目—Redis 技术分析及运用

爱旅行项目的认证模块（Auth）中，基于系统的安全性考虑，需要设置 Token 并通过 Redis 进行 Token 有效期的维护管理。下面介绍 Redis 技术以及在本项目中的具体运用。

1.1 Redis 简介

目前互联网基本使用两种方式来进行数据的存储：关系型数据库和 NoSql 数据库。

1、 关系数据库：主要用于存储格式化的数据结构，比如 MySQL、Oracle、SqlServer 等都属于关系型数据库。

2、 NoSql（泛指：非关系型数据库）数据库，其存储的数据都是按照键值对（Key-Value）的结构进行存储。主要服务于特定背景的专用数据库，向外提供特定的 API，而不是提供通用的 SQL 接口，所以效率更加高效，简单理解就是 NOSQL 可以不写任何 SQL 语句就能实现数据的存储与查询的数据库。

Redis 是一个开源 K-V 的数据库，属于 NoSql 数据库，并且与 Memcached 一样，为了保证效率，数据都是缓存在内存中，并基于内存操作，性能较高。它所支持存储的 value 类型相对更多，包括 string、list、set、zset 和 hash，在内存中设计了各种数据类型，让业务能够高速原子的访问这些数据结构，并且不需要关心持久存储的问题，从架构上解决了关系型数据库存储需要走一些弯路的问题。所以作为缓存应用，与其类似的 Memcached，EHCache,OSCache 等缓存器相比，Redis 的出现，很大程度补偿了 Memcached 这类 key/value 存储的不足，在高并发查询时可以对关系数据库起到很好的性能补充作用。此外与 Memcached 的最大区别在于：Redis 会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了 master-slave(主从)同步，故：Redis 支持主从同步，数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。

1.2 Redis 安装与配置

1.2.1 安装环境

1> 服务器：CentOS 6.8 64 位

2> redis_version:3.2.8

1.2.2 安装步骤

1. 创建安装路径：/usr/local/redis
2. 下载地址：http://redis.googlecode.com/files/redis-3.2.8.tar.gz
3. 解压安装：
   - tar -zxvf redis-3.2.8.tar.gz
   - make PREFIX=/usr/local/redis install
   - mkdir /usr/local/redis/etc

   注：可能会出现的错误提示如下：

   >>提示 1：

   make[3]: gcc：命令未找到

   >>解决

   yum -y install gcc-c++

   >>提示 2：

   在包含自 adlist.c：34 的文件中:

   zmalloc.h:50:31: 错误：jemalloc/jemalloc.h：没有那个文件或目录

   zmalloc.h:55:2: 错误：#error "Newer version of jemalloc required"

   >>解决

   make 的时候加上 MALLOC=libc 参数

   make PREFIX=/usr/local/redis MALLOC=libc install
4. 配置：
   - 设置内存分配策略（可选，根据服务器的实际情况进行设置，可选值：0、1、2 ） /proc/sys/vm/overcommit_memory

     echo 1 > /proc/sys/vm/overcommit_memory

     0：表示内核将检查是否有足够的可用内存供应用进程使用；如果有足够的可用内存，内存申请允许；否则，内存申请失败，并把错误返回给应用进程。

     1：表示内核允许分配所有的物理内存，而不管当前的内存状态如何。

2：表示内核允许分配超过所有物理内存和交换空间总和的内存

值得注意的一点是，Redis 在 dump 数据的时候，会 fork 出一个子进程，理论上 child 进程所占用的内存和 parent 是一样的，比如 parent 占用的内存为 8G，此时也要同样分配 8G 的内存给 child,如果内存无法负担，往往会造成 Redis 服务器的 down 机或者 IO 负载过高，效率下降。所以较为优化的内存分配策略应该设置为 1（表示内核允许分配所有的物理内存，而不管当前的内存状态如何）。

- 开启 Redis 端口（6379），修改防火墙配置文件
  - vi /etc/sysconfig/iptables
  - -A INPUT -m state --state NEW -m tcp -p tcp --dport 6379 -j ACCEPT
  - 重新加载规则：service iptables restart
- 指定 Redis 的 log 文件所在目录
  - 设置 Redis 配置文件 redis.conf 中参数 logfile
- 指定 RedisDB 的数据文件所在目录
  - 设置 Redis 配置文件 redis.conf 中参数 dir
- 设置 Redis 密码

  Redis 默认无密码，可设置密码，设置 redis 配置文件 redis.conf 中的参数 requirepass 的值即可，修改完毕后重启 Redis 服务。

5. 启动/停止 Redis：
   - 启动：redis-server /usr/local/redis/etc/redis.conf
   - 停止：kill -9 id

1.2.3 Redis 客户端

除了使用 Shell 命令行操作 Redis，在项目开发期间还可以安装 Redis 的图形化桌面客户端（RedisDesktopManager），如下图 1 所示：

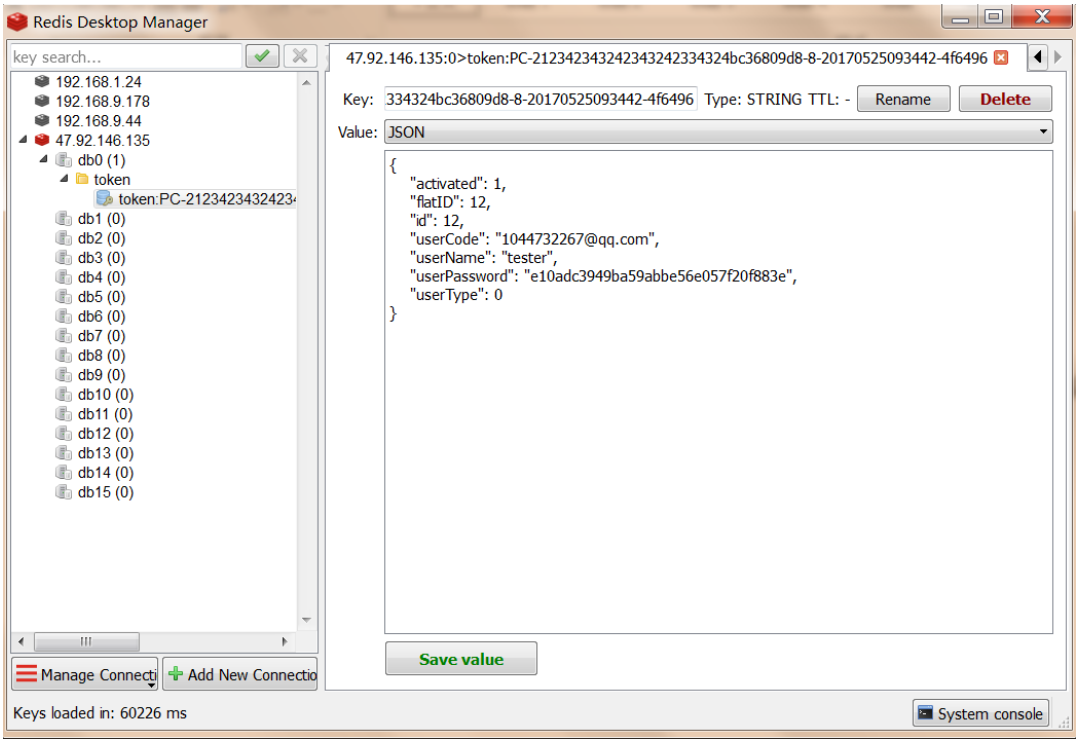

图 1：RedisDesktopManager

该客户端操作方便，图形化界面更加清晰明了，非常适用于开发阶段，关于其安装过程较为简单，不再赘述。

1.3 Redis 常用命令

下面介绍操作 Redis 的常用命令：

1. redis-cli：Shell 命令行下启动 Redis 客户端工具
2. set（set key value）：

   之前介绍过 Redis 是以 key-value 的格式来存储数据的，set 命令就是设置 key 对应的 value 值（string 类型），设置成功，返回 1；失败，返回 0。
3. get（get key）：通过 key 值获取其对应的 value 值，若不存在，则返回 nil。
4. exists（exists key）：判断指定的 key 是否存在，存在返回 1，不存在返回 0。
5. del(key)：删除一个指定 key。
6. quit：关闭连接。
7. info：查看 Redis 的相关信息。
8. flushdb：清空当前库。
9. flushall：清空所有数据库。

1.4 Redis 项目中的运用

1.4.1 模块实现

Redis 在爱旅行项目中主要负责 Token 的维护管理，针对不同的客户端的 Token（PC、移动），维护不同的 Token 有效期（根据需求设计 PC 端的 Token 有效时间为 2 个小时，移动端的 Token 永不失效，只有当修改密码后，在 Redis 中更新该 Token 中用户修改的新密码即可）。那么用户在系统平台中的所有操作，若需要进行用户身份的验证，只需去 Redis 中跟 key 查询相对应的 value 值即可，便可获取到当前用户的相关信息。

1.4.2 Redis 的设计思路

Redis 的具体使用很简单，只需要在 spring 配置文件中配置三个对象即可，分别为：jedisPoolConfig、jedisPool、redisAPI，如下图 2 所示：

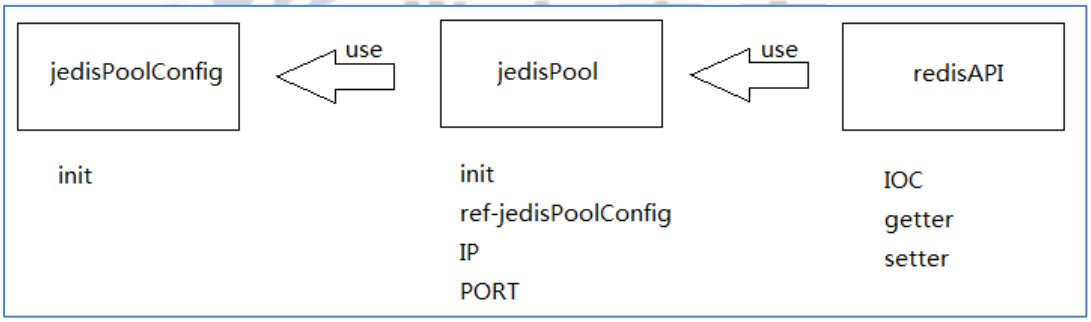

图 2

首先初始化 Redis 连接池配置对象（jedisPoolConfig），然后再将该对象通过 IoC 注入到连接池对象（jedisPool）中，当连接池对象初始化时引用连接池配置对象（jedisPoolConfig），同时通过配置的 IP、端口等进行连接的实例化。最后由自定义的 redisAPI 来使用 jedisPool 对象。redisAPI 是基于 jedisPool 编写的一个数据访问对象，相当于系统中的 dao 层，比如说对数据进行一些 set、get 等操作。

1.4.3 Redis 的相关配置

1. JedisPoolConfig：Redis 连接池配置

   首先进行 spring IoC 配置，在 spring 容器中注册一个 ID 为 jedisPoolConfig 的 bean，使用的类是 redis.clients.jedis.JedisPoolConfig 。

```
<bean id="jedisPoolConfig" class="redis.clients.jedis.JedisPoolConfig">
    <property  name="maxActive"  value="${redis.maxActive}" />
    <property  name="maxIdle"    value="${redis.maxIdle}" />
    <property  name="maxWait"  value="${redis.maxWait}" />
    <property  name="testOnBorrow"  value="true" />
</bean>
```

- maxActive：最大活动数。

  若设置 value 为-1，则表示不限制，若 pool 已经分配了 n（设置的 value）个的 jedis 实例，则此时 pool 的状态为 exhausted(耗尽)。

- maxIdle：最大空闲数。
- maxWait：表示当 borrow(引入/借调)一个 jedis 实例时，最大的等待时间，如果超过等待时间，则直接抛出 JedisConnectionException。
- testOnBorrow：在 borrow 一个 jedis 实例时，是否提前进行 validate 操作；若为 true，则得到的 jedis 实例均是可用的。

2. jedisPool

在 spring 容器中注册一个 ID 为 jedisPool 的 bean，使用的类是 redis.clients.jedis.JedisPool，再将配置好的 redis 的连接池配置对象（jedisPoolConfig）通过 IoC 注入到连接池对象中，连接池对象初始化时引用上面的连接池配置对象，同时通过配置的 IP 和端口等进行连接的实例化。

```
<bean id="jedisPool" class="redis.clients.jedis.JedisPool" destroy-method="destroy" >
    <constructor-arg ref="jedisPoolConfig"/>
    <constructor-arg value="${redis.host}"/>
    <constructor-arg value="${redis.port}"/>
    <constructor-arg value="${redis.timeout}"/>
    <constructor-arg value="${redis.pass}"/>
    <constructor-arg value="0"/>
</bean>
```

3. redisAPI

在 spring 容器中注册一个 ID 为 redisAPI 的 bean，这个 bean 使用的类是 org.slsale.common.RedisAPI，在实例化这个 bean 的时候要通过 jedisPool 进行初始化（相当于赋值）。

```
<bean id="redisAPI" class="cn.itrip.common.RedisAPI">
    <property name="jedisPool" ref="jedisPool"/>
</bean>
```

4. validationToken

该工具类主要是为了根据传入的 key（Token）去 Redis 中获取相应用户信息。

```xml
<bean id="validationToken" class="cn.itrip.common.ValidationToken">
    <property name="redisAPI" ref="redisAPI" />
</bean>
```

1.4.4 代码实现

根据分析，redisAPI 需要提供以下六个方法，以后根据项目需求，该 API 可以不断扩充。

1> boolean set(String key,String value)：set key value 到 redis 中

2> boolean set(String key,int seconds,String value)：set key、value、有效期到 redis 中

3> boolean get(String key)：通过 key 获取 value

4> boolean exist(String key)：判断某个 key 是否存在

5> delete(String key)：删除指定的 key

6> Long ttl(String key)：查询 key 的有效期。当 key 不存在时，返回 -2 ；当 key 存在但没有设置剩余有效时间时，返回 -1，否则，以秒为单位，返回 key 的剩余有效时间

作为业务实现，直接调用 redisAPI 的接口方法进行业务逻辑的处理即可。下面以工具类 ValidationToken.java 的代码实现为例进行介绍。

该工具类主要作用是调用 redisAPI 的 get 方法，通过传入的 key（Token）来获取相应 value 值，即当前用户信息（ItripUser），以验证当前用户是否登录状态。若获取到 value，即可获取 ItripUser 对象，否则返回空，则需要用户进行登录操作。代码片段如下：

```java
public class ValidationToken {
    private Logger logger = Logger.getLogger(ValidationToken.class);
    private RedisAPI redisAPI;
    public RedisAPI getRedisAPI() {
        return redisAPI;
    }
    public void setRedisAPI(RedisAPI redisAPI) {
        this.redisAPI = redisAPI;
    }
    public ItripUser getCurrentUser(String tokenString){//根据token从redis中获取用户信息
        ItripUser itripUser = null;
        if(null == tokenString || "".equals(tokenString)){
            return null;
        }
        try{
            String userInfoJson = redisAPI.get(tokenString);
            itripUser = JSONObject.parseObject(userInfoJson,ItripUser.class);
        }catch(Exception e){
            itripUser = null;
            logger.error("get userinfo from redis but is error : " + e.getMessage());
        }
        return itripUser;
    }
```

注意：所有的工具类中尽量不要使用注解，以降低耦合度，比如：RedisAPI 的注入。