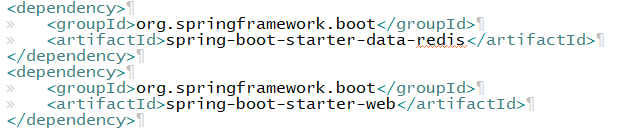
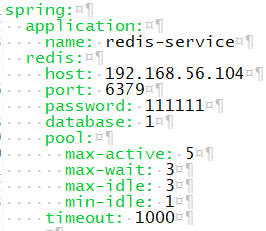
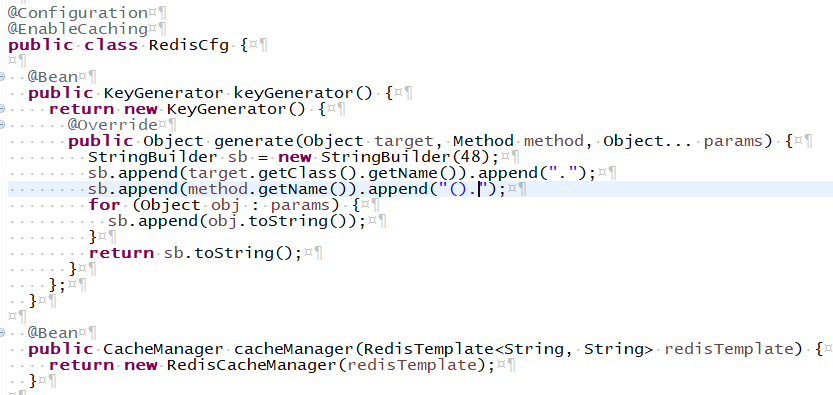
# Spring Boot使用Redis

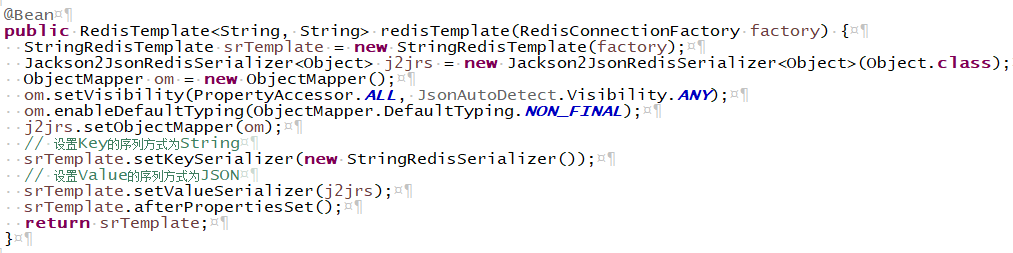
## 相关配置





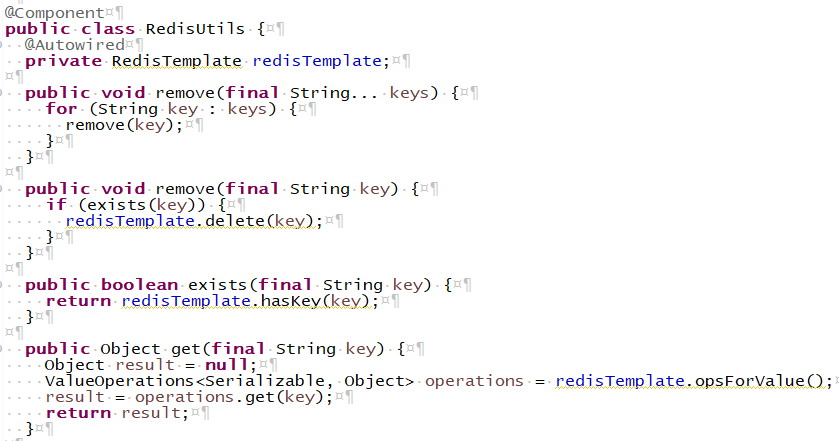
Redis配置类

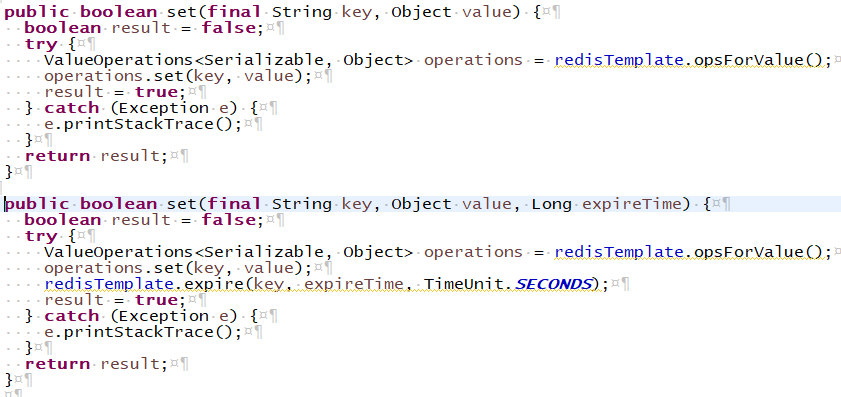




在这里我们将Key设置为String，Value设置为JSON格式。

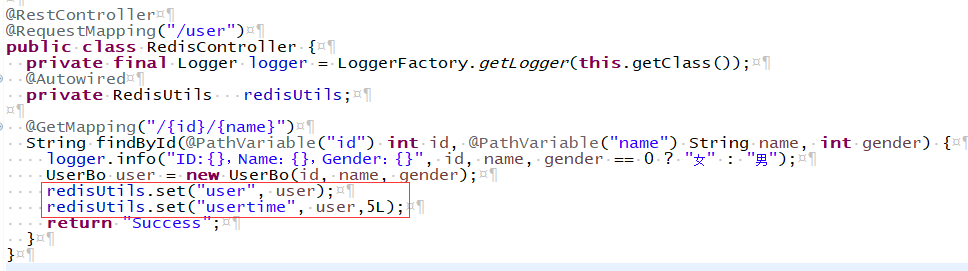
Redis工具类





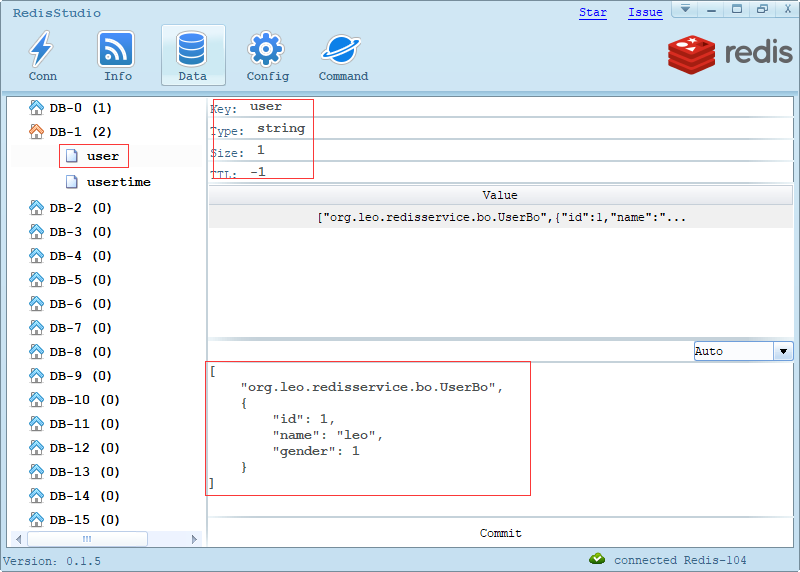
工具类主要是给开发人员手工操作Redis用，可以根据实际工作进行扩展，这里就写了常用的几种操作。

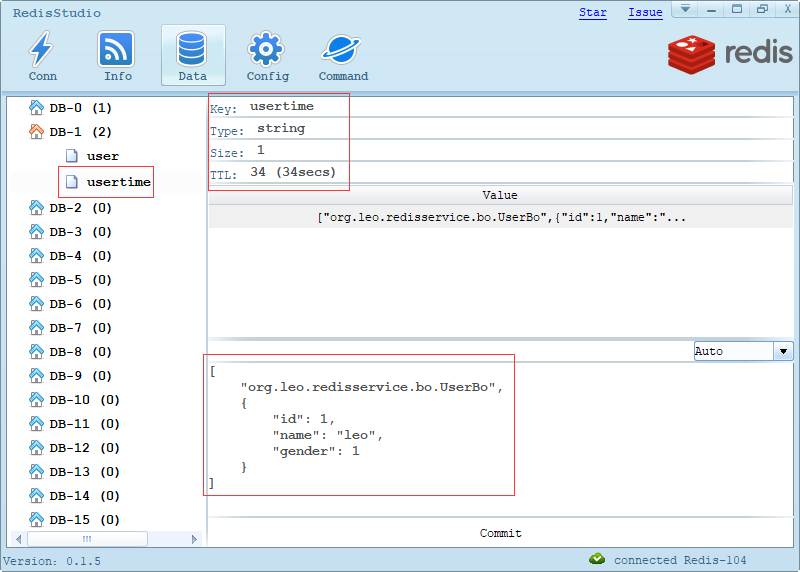
## 手工操作示例



在这里我们向RedisSet了两条数据，一条没有设置有效时间，一条设置了（图上写的是5，真正例子中是60）。

查询一下Redis结果如下：



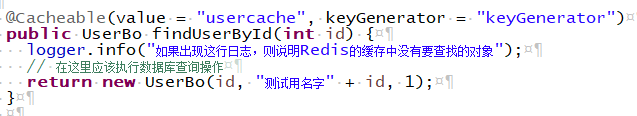


注意看二者TTL的不同。

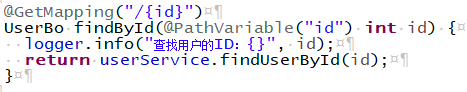
## 自动缓存

在Application类上增加@EnableCaching这个注解。

创建一个UserService，增加方法如下：



Controller增加如下方法：



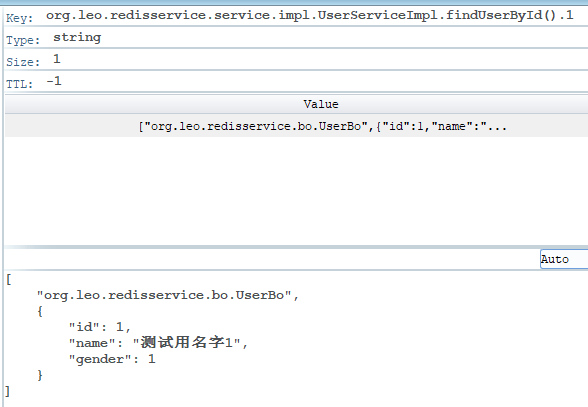
访问页面两次，后台日志如下：



可以看出，第一次因为Redis里没有ID为1的User，执行了Service——在实际工程中，Service应执行Dao或Mapper，从DB取数据。

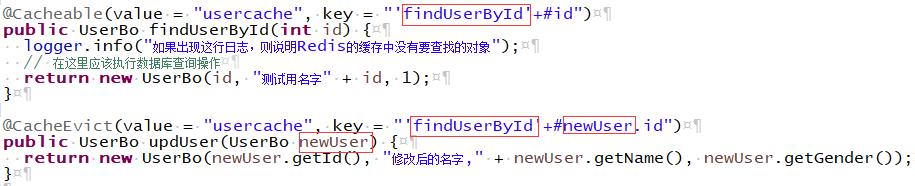
第二次，因为取得数据已经放入Redis了，则没有执行Service。

Redis里的数据是这样的：



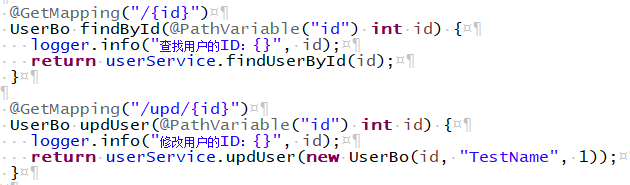
Key是由RedisCfg的keyGenerator生成的，查看代码可以知道，其实就是将类名、方法名、参数拼起来。但是在实际工作中有一个问题：当我们对某条数据进行修改、删除的时候，这条数据也应该从Redis里删除，这时候就需要你在代码里指定Key，可是如果在修改、删除的方法中使用keyGenerator生成的Key与获取对象那个方法生成的的Key是不一致的，这个很好理解吧。所以我们要改一下Key的生成方式。

修改Service：



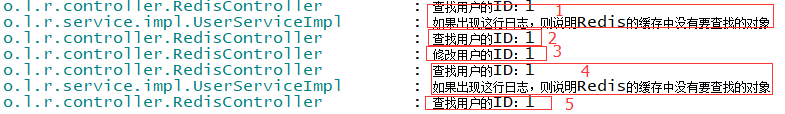
注意看红框，要一一对应。本例就写了update方法，delete方法的注解写法相同。

Controller：



按顺序访问页面：1获取对象>2获取对象>3修改对象>4获取对象>5获取对象。

后台日志如下：



Redis里的数据如下：

