Redis

一、概念

redis 是一款高性能的 NOSQL 系列的非关系型数据库

1. 什么是 NOSQL

NoSQL(NoSQL = Not Only SQL)，意即“不仅仅是 SQL”，是一项全新的数据库理念，泛指非关系型的数据库。

随着互联网 web2.0 网站的兴起，传统的关系数据库在应付 web2.0 网站，特别是超大规模和高并发的 SNS 类型的 web2.0 纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。NoSQL 数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题。

① NOSQL 和关系型数据库比较

- 优点：
  - 成本：nosql 数据库简单易部署，基本都是开源软件，不需要像使用 oracle 那样花费大量成本购买使用，相比关系型数据库价格便宜。
  - 查询速度：nosql 数据库将数据存储于缓存之中，关系型数据库将数据存储在硬盘中，自然查询速度远不及 nosql 数据库。
  - 存储数据的格式：nosql 的存储格式是 key,value 形式、文档形式、图片形式等等，所以可以存储基础类型以及对象或者是集合等各种格式，而数据库则只支持基础类型。
  - 扩展性：关系型数据库有类似 join 这样的多表查询机制的限制导致扩展很艰难。

- 缺点：
  - 维护的工具和资料有限，因为 nosql 是属于新的技术，不能和关系型数据库 10 几年的技术同日而语。
  - 不提供对 sql 的支持，如果不支持 sql 这样的工业标准，将产生一定用户的学习和使用成本。
  - 不提供关系型数据库对事务的处理。

② 非关系型数据库的优势：

性能 NOSQL 是基于键值对的，可以想象成表中的主键和值的对应关系，而且不需要经过 SQL 层的解析，所以性能非常高。

可扩展性同样也是因为基于键值对，数据之间没有耦合性，所以非常容易水平扩展。

③ 关系型数据库的优势：

复杂查询可以用 SQL 语句方便的在一个表以及多个表之间做非常复杂的数据查询。

事务支持使得对于安全性能很高的数据访问要求得以实现。对于这两类数据库，对方的优势就是自己的弱势，反之亦然。

④ 总结

关系型数据库与 NoSQL 数据库并非对立而是互补的关系，即通常情况下使用关系型数据库，在适合使用 NoSQL 的时候使用 NoSQL 数据库，

让 NoSQL 数据库对关系型数据库的不足进行弥补。

一般会将数据存储在关系型数据库中，在 nosql 数据库中备份存储关系型数据库的数据

2. 主流的 NOSQL 产品

- 键值(Key-Value)存储数据库
  - 相关产品： Tokyo Cabinet/Tyrant、Redis、Voldemort、Berkeley DB
  - 典型应用： 内容缓存，主要用于处理大量数据的高访问负载。
  - 数据模型： 一系列键值对
  - 优势： 快速查询
  - 劣势： 存储的数据缺少结构化
- 列存储数据库
  - 相关产品：Cassandra, HBase, Riak
  - 典型应用：分布式的文件系统
  - 数据模型：以列簇式存储，将同一列数据存在一起
  - 优势：查找速度快，可扩展性强，更容易进行分布式扩展

- 劣势：功能相对局限

- 文档型数据库
  - 相关产品：CouchDB、MongoDB
  - 典型应用：Web 应用（与 Key-Value 类似，Value 是结构化的）
  - 数据模型： 一系列键值对
  - 优势：数据结构要求不严格
  - 劣势： 查询性能不高，而且缺乏统一的查询语法
- 图形(Graph)数据库
  - 相关数据库：Neo4J、InfoGrid、Infinite Graph
  - 典型应用：社交网络
  - 数据模型：图结构
  - 优势：利用图结构相关算法。
  - 劣势：需要对整个图做计算才能得出结果，不容易做分布式的集群方案。

3. 什么是 Redis

Redis 是用 C 语言开发的一个开源的高性能键值对（key-value）数据库，官方提供测试数据，50 个并发执行 100000 个请求,读的速度是 110000 次/s,写的速度是 81000 次/s ，且 Redis 通过提供多种键值数据类型来适应不同场景下的存储需求，目前为止 Redis 支持的键值数据类型如下：

- 字符串类型 string
- 哈希类型 hash
- 列表类型 list
- 集合类型 set
- 有序集合类型 sortedset

① redis 的应用场景

- 缓存（数据查询、短连接、新闻内容、商品内容等等）
- 聊天室的在线好友列表
- 任务队列。（秒杀、抢购、12306 等等）
- 应用排行榜
- 网站访问统计
- 数据过期处理（可以精确到毫秒）
- 分布式集群架构中的 session 分离

二、下载安装

1. Mac 系统安装 redis

- 下载 redis :

网址：https://redis.io/download，下载 stable 版本，稳定版本。

- 解压：

tar zxvf redis-5.0.4.tar.gz。

- 将解压后文件夹放到/usr/local

mv redis-5.0.4 /usr/local/

- 切换到切换到相应目录

cd /usr/local/redis-5.0.4/

- 进入 src 目录下面编译 redis

sudo make

编译成功后，会有一个提示： It's a good idea to run 'make test'

（这里可能会有“invalid active developer path (/Library/Developer/CommandLineTools), missing xcrun at:/Library/Developer/CommandLineTools/usr/bin/xcrun”问题，解决方法很简单，就是在命令行中执行 xcode-select --install，然后下载，安装完成后，再重新编译即可）

- 编译 test

sudo make test

成功后提示： All tests passed without errors

- 安装

sudo make install

成功后提示： It's a good idea to run 'make test'！

- 运行

redis-server

三、命令操作

1. redis 的数据结构：

redis 存储的是：key,value 格式的数据，其中 key 都是字符串，value 有 5 种不同的数据结构

value 的数据结构：

- 字符串类型 string
- 哈希类型 hash ： map 格式
- 列表类型 list ： linkedlist 格式。支持重复元素
- 集合类型 set ： 不允许重复元素
- 有序集合类型 sortedset：不允许重复元素，且元素有顺序

2. 字符串类型 string

- 存储： set key value

  127.0.0.1:6379> set username zhangsan

  OK
- 获取： get key

  127.0.0.1:6379> get username

  "zhangsan"
- 删除： del key

  127.0.0.1:6379> del age

  (integer) 1

3. 哈希类型 hash

- 存储： hset key field value

  127.0.0.1:6379> hset myhash username lisi

  (integer) 1

  127.0.0.1:6379> hset myhash password 123

  (integer) 1
- 获取：
  - ➢ hget key field: 获取指定的 field 对应的值

127.0.0.1:6379> hget myhash username

"lisi"

  - ➢ hgetall key：获取所有的 field 和 value

127.0.0.1:6379> hgetall myhash

1) "username"

2) "lisi"

3) "password"

4) "123"

- 删除： hdel key field

127.0.0.1:6379> hdel myhash username

(integer) 1

4. 列表类型 list:

可以添加一个元素到列表的头部（左边）或者尾部（右边）

- 添加：
  - lpush key value: 将元素加入列表左表
  - rpush key value：将元素加入列表右边

127.0.0.1:6379> lpush myList a

(integer) 1

127.0.0.1:6379> lpush myList b

(integer) 2

127.0.0.1:6379> rpush myList c

(integer) 3

- 获取：
  - range key start end ：范围获取

127.0.0.1:6379> lrange myList 0 -1

1) "b"

2) "a"

3) "c"

- 删除：
  - lpop key： 删除列表最左边的元素，并将元素返回
  - rpop key： 删除列表最右边的元素，并将元素返回

5. 集合类型 set

不允许重复元素

- 存储：sadd key value

127.0.0.1:6379> sadd myset a

(integer) 1

127.0.0.1:6379> sadd myset a

(integer) 0

- 获取：smembers key:获取 set 集合中所有元素

127.0.0.1:6379> smembers myset

1) "a"

- 删除：srem key value:删除 set 集合中的某个元素

127.0.0.1:6379> srem myset a

(integer) 1

6. 有序集合类型 sortedset

不允许重复元素，且元素有顺序.每个元素都会关联一个 double 类型的分数。redis 正是通过分数来为集合中的成员进行从小到大的排序。

- 存储：zadd key score value

127.0.0.1:6379> zadd mysort 60 zhangsan

(integer) 1

127.0.0.1:6379> zadd mysort 50 lisi

(integer) 1

127.0.0.1:6379> zadd mysort 80 wangwu

(integer) 1

- 获取：zrange key start end [withscores]

127.0.0.1:6379> zrange mysort 0 -1

1) "lisi"

2) "zhangsan"

3) "wangwu"

127.0.0.1:6379> zrange mysort 0 -1 withscores

1) "zhangsan"

2) "60"

3) "wangwu"

4) "80"

5) "lisi"

6) "500"

- 删除：zrem key value

127.0.0.1:6379> zrem mysort lisi

(integer) 1

7. 通用命令

- keys * : 查询所有的键
- type key ： 获取键对应的 value 的类型
- del key：删除指定的 key value

四、持久化

redis 是一个内存数据库，当 redis 服务器重启，获取电脑重启，数据会丢失，我们可以将 redis 内存中的数据持久化保存到硬盘的文件中。

1. redis 持久化机制：

- RDB

默认方式，不需要进行配置，默认就使用这种机制

在一定的间隔时间中，检测 key 的变化情况，然后持久化数据

- AOF

日志记录的方式，可以记录每一条命令的操作。可以每一次命令操作后，持久化数据

五、Java 客户端 Jedis

Jedis: 一款 java 操作 redis 数据库的工具.

1. 使用步骤：

下载 jedis 的 jar 包

使用

```
//1. 获取连接
Jedis jedis = new Jedis("localhost",6379);
//2. 操作
jedis.set("username","zhangsan");
//3. 关闭连接
```

```
jedis.close();
```

2. Jedis 操作各种 redis 中的数据结构

① 字符串类型 string

set

get

```
 //1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造，默认值 "localhost",6379 端口
//2. 操作
//存储
jedis.set("username","zhangsan");
//获取
String username = jedis.get("username");
System.out.println(username);

//可以使用 setex()方法存储可以指定过期时间的 key value
jedis.setex("activecode",20,"hehe");//将 activecode：hehe 键值对存入 redis，并且 20 秒后自动删除该键值对

//3. 关闭连接
jedis.close();
```

② 哈希类型 hash ： map 格式

hset

hget

hgetAll

```
//1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造，默认值 "localhost",6379 端口
//2. 操作
// 存储 hash
jedis.hset("user","name","lisi");
```

```
jedis.hset("user","age","23");
jedis.hset("user","gender","female");

// 获取 hash
String name = jedis.hget("user", "name");
System.out.println(name);

 // 获取 hash 的所有 map 中的数据
Map<String, String> user = jedis.hgetAll("user");

// keyset
Set<String> keySet = user.keySet();
for (String key : keySet) {
      //获取 value
      String value = user.get(key);
      System.out.println(key + ":" + value);
}

//3. 关闭连接
jedis.close();
```

③ 列表类型 list

linkedlist 格式。支持重复元素

lpush / rpush

lpop / rpop

lrange start end : 范围获取

```
//1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造，默认值 "localhost",6379 端口
//2. 操作
// list 存储
jedis.lpush("mylist","a","b","c");//从左边存
jedis.rpush("mylist","a","b","c");//从右边存

// list 范围获取
```

```
List<String> mylist = jedis.lrange("mylist", 0, -1);
System.out.println(mylist);

// list 弹出
String element1 = jedis.lpop("mylist");//c
System.out.println(element1);

String element2 = jedis.rpop("mylist");//c
System.out.println(element2);

// list 范围获取
List<String> mylist2 = jedis.lrange("mylist", 0, -1);
System.out.println(mylist2);

//3. 关闭连接
jedis.close();
```

④ 集合类型 set

不允许重复元素

sadd

smembers:获取所有元素

```
//1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造，默认值 "localhost",6379 端口
//2. 操作
// set 存储
jedis.sadd("myset","java","php","c++");

// set 获取
Set<String> myset = jedis.smembers("myset");
System.out.println(myset);

//3. 关闭连接
jedis.close();
```

⑤ 有序集合类型 sortedset

不允许重复元素，且元素有顺序

zadd

zrange

```
//1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造，默认值 "localhost",6379 端口
//2. 操作
// sortedset 存储
jedis.zadd("mysortedset",3,"亚瑟");
jedis.zadd("mysortedset",30,"后裔");
jedis.zadd("mysortedset",55,"孙悟空");

// sortedset 获取
Set<String> mysortedset = jedis.zrange("mysortedset", 0, -1);

System.out.println(mysortedset);
 //3. 关闭连接
jedis.close();
```

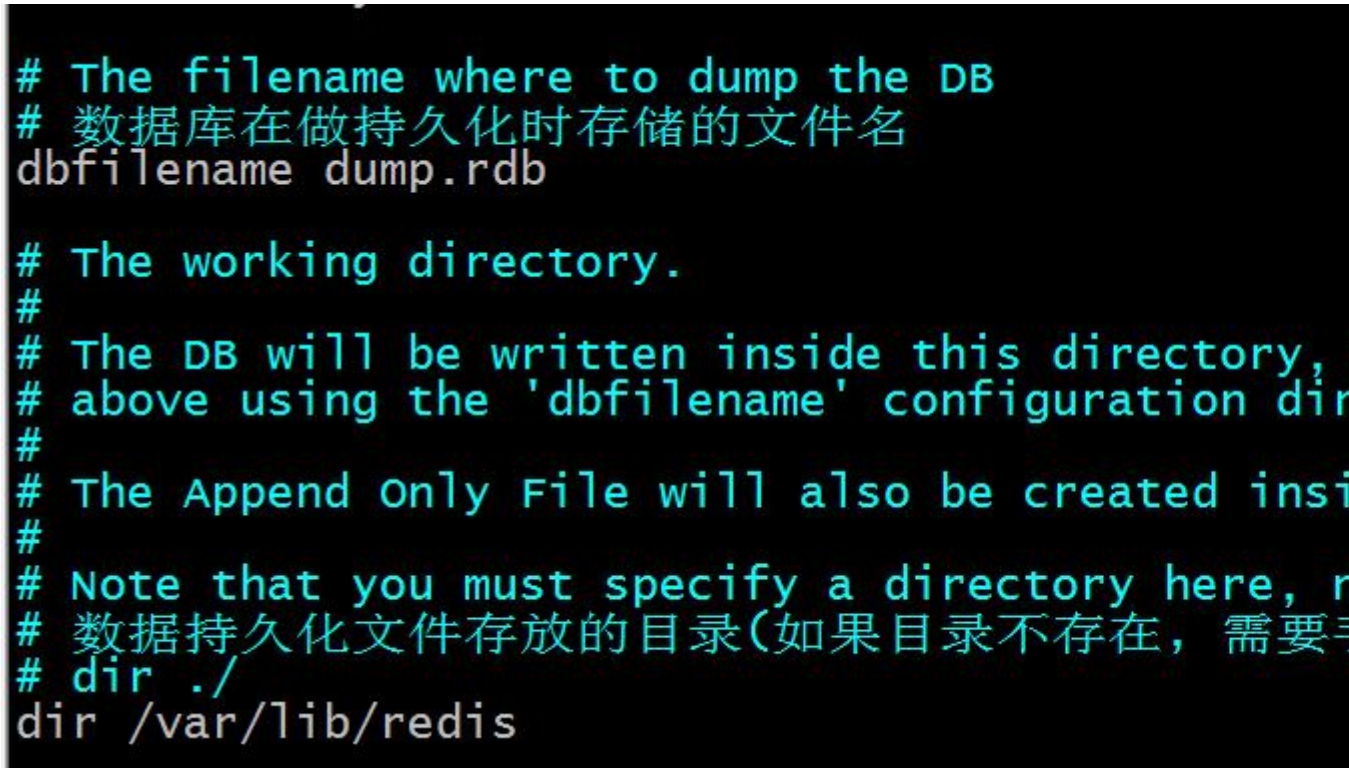
六、 jedis 连接池： JedisPool

```
//0.创建一个配置对象
JedisPoolConfig config = new JedisPoolConfig();
config.setMaxTotal(50);
config.setMaxIdle(10);

//1.创建 Jedis 连接池对象
JedisPool jedisPool = new JedisPool(config,"localhost",6379);

//2.获取连接
Jedis jedis = jedisPool.getResource();
//3. 使用
jedis.set("hehe","heihei");
```

```
//4. 关闭 归还到连接池中
jedis.close();
```

redis 保存异常: Background saving error

进入配置文件(make redis 目录), 查看持久化文件 dump.rdb 存放目录



修改持久化目录的权限

sudo chmod 777 /var/lib/redis/

修改文件权限:

$ chmod g+w dump.rdb

关闭 redis

redis-cli

shutdown nosave

启动 redis

redis-server /etc/redis/redis.conf

若问题解决, 就不用进行下一步操作了

修改配置文件的权限

sudo chmod 777 /etc/redis/redis.conf

七、SpringBoot 整合 Redis

添加 redis 的起步依赖

```
<!-- 配置使用 redis 启动器 -->
 <dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-data-redis</artifactId>
 </dependency>
```

配置 redis 的连接信息

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

注入 RedisTemplate 测试 redis 操作

```
@RunWith(SpringRunner.class)
 @SpringBootTest(classes = SpringbootJpaApplication.class)
 public class RedisTest {
     @Autowired
     private UserRepository userRepository;

     @Autowired
     private RedisTemplate<String, String> redisTemplate;

     @Test
     public void test() throws JsonProcessingException {
         //从 redis 缓存中获得指定的数据
         String userListData = redisTemplate.boundValueOps("user.findAll").get();
         //如果 redis 中没有数据的话
         if(null==userListData){
             //查询数据库获得数据
             List<User> all = userRepository.findAll();
             //转换成 json 格式字符串
             ObjectMapper om = new ObjectMapper();
             userListData = om.writeValueAsString(all);
             //将数据存储到 redis 中，下次在查询直接从 redis 中获得数据，不用在查询数
据库
```

```
                redisTemplate.boundValueOps("user.findAll").set(userListData);
                System.out.println("==========从数据库获得数据==========");
            }else{
                System.out.println("==========从redis缓存中获得数据==========");
            }
            System.out.println(userListData);
        }
}
```

八、案例

需求：

1. 提供 index.html 页面，页面中有一个省份下拉列表
2. 当页面加载完成后 发送 ajax 请求，加载所有省份