* 1. 六、测试分析
  2. 七、安装指南
  3. 八、附录A（缓存服务子系统接口）

数据类型:

缓存作为key-value的快速映射, key是字符串类型的. value支持两种数据类型, 分别是string, hash.

其中字符串类型就是字符序列, 不支持空字符序列, 空字符序列等效于null.

其中hash是索引-值得集合, 索引和值都是字符串类型的. 例如下面:

{'a': 'hello', 'b': 'value'}

可以看成是一个hash数据.

缓存服务子系统支持客户端通过两种协议来访问, 分别是HTTP和RMI. 针对HTTP协议, 提供了客户端JAR封装.

下面是对HTTP协议的接口描述, 客户端JAR和RMI具有等效的接口, 就不单独列出来了.

提前说明:

下面的接口描述中, 所给的参数都不能为空, 加上[]的除外

每个接口都有用户隔离的功能, 使用token参数进行区别, 就没有单独列出来了

例如对于用户mingming来说, 其完整的调用应该是/set?key=hello&value=world&token=mingming

没有单独提供remove数据的接口, 因为意识到用户很少有意识的删除数据, 故前期就不实现了.

但每个接口都添加有过期值的支持, 参数名为expire, value值是秒数

例如想要key在10秒后过期, 完整的调用应该是/set?key=hello&value=world&token=mingming&expire=10

[过期值有妙用]

关于标量string的接口:

/set: key, value

存储字符串的值

例如: /set?key=hello&value=world

/get: key => value

返回字符串的值

例如: /get?key=hello => world

关于hashtable的接口:

/hash/setAll: key, value

存储hashtable类型的值,

其中hashtable内部元素的值是由","和":"分割的字符串键值对序列,

字符串值中的",:\"需要用"\"转义

例如: /table/setAll?key=hello&value=k1:1,k2:2,k3:3

/hash/getAll: key => value

返回hashtable类型的值

例如: /table/getAll?key=hello => k1:1,k2:2,k3:3

/hash/set: key, index, value

设置hashtable集合内指定索引键的值

例如: /table/set?key=hello&index=k1&value=b

/hash/get: key, index

返回hashtable集合内指定索引建的值

例如: /table/get?key=hello&index=k1

/hash/remove:

删除hashtable集合内指定索引建的键值对

例如: /table/remove?key=hello&index=k1

/hash/size: key => size

返回hashtable的大小

例如: /hash/size?key=hello => 3

Both客户端JAR包和RMI远程接口我们都给了JAVA DOC

客户端JAR包:

用户可以使用JAVA语言来使用缓存服务. 需要引入下面的两个包:

cache-client.jar

groovy-all.x.x.x.jar (groovy依赖包)

使用时, 首先使用emem.cacheclient.CacheClient的构造方法创建一个实例, 如下:

import emem.cacheclient.CacheClient;

HTTPCacheClient cacheClient = new HTTPCacheClient(host, port, token);

然后使用的是与HTTP协议等效的方法, 例如set操作:

cacheClient.set(key, value)

cacheClient.set(key, value, expire)

其他的操作类似, 就不一一列出了.

另外, 为了便利JAVA接口的使用, 可以直接将一个完整的对象存储进缓存, 以后可以从缓存提取这个对象, 例如:

Person person = new Person(name, age)

cacheClient.set(key, person)

person = (Person)cacheClient.get(key)

下面是客户端JAR包的JAVA DOC文档摘要：

|  |  |
| --- | --- |
| 限定符和类型 | 方法和说明 |
| java.lang.String | [get](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "get(java.lang.String))(java.lang.String key)  返回绑定到制定键的字符串类型的值 |
| java.lang.String | [get](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "get(java.lang.String, int))(java.lang.String key, int expire) |
| java.io.Serializable | [getObject](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "getObject(java.lang.String))(java.lang.String key)  返回对象类型的数据, 该对象需要实现Serializable接口. |
| java.io.Serializable | [getObject](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "getObject(java.lang.String, int))(java.lang.String key, int expire) |
| java.lang.String | [hashGet](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashGet(java.lang.String, java.lang.String))(java.lang.String key, java.lang.String index)  返回hash类型的数据, 仅仅对其中的某个索引进行操作. |
| java.lang.String | [hashGet](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashGet(java.lang.String, java.lang.String, int))(java.lang.String key, java.lang.String index, int expire) |
| java.util.Map<java.lang.String,java.lang.String> | [hashGetAll](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashGetAll(java.lang.String))(java.lang.String key)  返回hash类型的完整数据 |
| java.util.Map<java.lang.String,java.lang.String> | [hashGetAll](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashGetAll(java.lang.String, int))(java.lang.String key, int expire) |
| void | [hashRemove](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashRemove(java.lang.String, java.lang.String))(java.lang.String key, java.lang.String index)  可以针对hash数据中的某个索引进行删除 |
| void | [hashRemove](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashRemove(java.lang.String, java.lang.String, int))(java.lang.String key, java.lang.String index, int expire) |
| void | [hashSet](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSet(java.lang.String, java.lang.String, java.lang.String))(java.lang.String key, java.lang.String index, java.lang.String value)  设置hash类型的数据, 仅仅对其中的某个索引进行操作. |
| void | [hashSet](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSet(java.lang.String, java.lang.String, java.lang.String, int))(java.lang.String key, java.lang.String index, java.lang.String value, int expire) |
| void | [hashSetAll](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSetAll(java.lang.String, java.util.Map))(java.lang.String key, java.util.Map<java.lang.String,java.lang.String> map)  设置hash类型的完整数据, 其中hash值用一个字符串到字符串的映射map表示 |
| void | [hashSetAll](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSetAll(java.lang.String, java.util.Map, int))(java.lang.String key, java.util.Map<java.lang.String,java.lang.String> map, int expire) |
| long | [hashSize](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSize(java.lang.String))(java.lang.String key)  返回的是hash数据中索引的个数 |
| long | [hashSize](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSize(java.lang.String, int))(java.lang.String key, int expire) |
| void | [set](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "set(java.lang.String, java.lang.String))(java.lang.String key, java.lang.String value)  存储单纯的字符串类型的键值对 |
| void | [set](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "set(java.lang.String, java.lang.String, int))(java.lang.String key, java.lang.String value, int expire) |
| void | [setObject](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "setObject(java.lang.String, java.io.Serializable))(java.lang.String key, java.io.Serializable obj)  存储对象类型的数据, 该对象需要实现Serializable接口. |
| void | [setObject](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "setObject(java.lang.String, java.io.Serializable, int))(java.lang.String key, java.io.Serializable obj, int expire) |

RMI远程接口:

缓存服务子系统支持客户端使用RMI协议进行缓存操作. 假设变量host和port保存RMI服务器的主机和端口. 客户端需要的动作如下:

1. 首先引入两个远程接口, CacheClientFactory和CacheClient(在包emem.common.rmi下, 已打包为cache-rmi.jar).

2. 通过名字查找获取CacheClientFactory

CacheClientFactory cacheClientFactory = Naming.lookup("rmi://$host:$port/CacheClientFactory");

3. 传入token获取CacheClient:

CacheClient cacheClient = cacheClientFactory.getCacheClient(token);

4. 使用cacheClient进行操作即可, 方法与客户端JAR包的一致.

* 1. **5. 下面是RMI接口的JAVA doc表格摘要：**

|  |  |
| --- | --- |
| 限定符和类型 | 方法和说明 |
| java.lang.String | [get](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "get(java.lang.String))(java.lang.String key)  返回绑定到制定键的字符串类型的值 |
| java.lang.String | [get](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "get(java.lang.String, int))(java.lang.String key, int expire) |
| java.io.Serializable | [getObject](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "getObject(java.lang.String))(java.lang.String key)  返回对象类型的数据, 该对象需要实现Serializable接口. |
| java.io.Serializable | [getObject](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "getObject(java.lang.String, int))(java.lang.String key, int expire) |
| java.lang.String | [hashGet](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashGet(java.lang.String, java.lang.String))(java.lang.String key, java.lang.String index)  返回hash类型的数据, 仅仅对其中的某个索引进行操作. |
| java.lang.String | [hashGet](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashGet(java.lang.String, java.lang.String, int))(java.lang.String key, java.lang.String index, int expire) |
| java.util.Map<java.lang.String,java.lang.String> | [hashGetAll](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashGetAll(java.lang.String))(java.lang.String key)  返回hash类型的完整数据 |
| java.util.Map<java.lang.String,java.lang.String> | [hashGetAll](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashGetAll(java.lang.String, int))(java.lang.String key, int expire) |
| void | [hashRemove](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashRemove(java.lang.String, java.lang.String))(java.lang.String key, java.lang.String index)  可以针对hash数据中的某个索引进行删除 |
| void | [hashRemove](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashRemove(java.lang.String, java.lang.String, int))(java.lang.String key, java.lang.String index, int expire) |
| void | [hashSet](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSet(java.lang.String, java.lang.String, java.lang.String))(java.lang.String key, java.lang.String index, java.lang.String value)  设置hash类型的数据, 仅仅对其中的某个索引进行操作. |
| void | [hashSet](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSet(java.lang.String, java.lang.String, java.lang.String, int))(java.lang.String key, java.lang.String index, java.lang.String value, int expire) |
| void | [hashSetAll](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSetAll(java.lang.String, java.util.Map))(java.lang.String key, java.util.Map<java.lang.String,java.lang.String> map)  设置hash类型的完整数据, 其中hash值用一个字符串到字符串的映射map表示 |
| void | [hashSetAll](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSetAll(java.lang.String, java.util.Map, int))(java.lang.String key, java.util.Map<java.lang.String,java.lang.String> map, int expire) |
| long | [hashSize](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSize(java.lang.String))(java.lang.String key)  返回的是hash数据中索引的个数 |
| long | [hashSize](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "hashSize(java.lang.String, int))(java.lang.String key, int expire) |
| void | [set](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "set(java.lang.String, java.lang.String))(java.lang.String key, java.lang.String value)  存储单纯的字符串类型的键值对 |
| void | [set](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "set(java.lang.String, java.lang.String, int))(java.lang.String key, java.lang.String value, int expire) |
| void | [setObject](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "setObject(java.lang.String, java.io.Serializable))(java.lang.String key, java.io.Serializable obj)  存储对象类型的数据, 该对象需要实现Serializable接口. |
| void | [setObject](file:///home/hello/devs/workspace/CloudCache/CacheServer/Writing Docs/JAVADOC/emem/common/rmi/CacheClient.html" \l "setObject(java.lang.String, java.io.Serializable, int))(java.lang.String key, java.io.Serializable obj, int expire) |