# 关于公共 API

源码文件：modules/accounts.js

API 功能解读：

```
367
368     router.map(shared, {    //用router匹配shared中的方法（shared是什么？？）
369         "post /open": "open",//打开账户(返回用户全部信息)
370         "get /getBalance": "getBalance",//获得余额（Balance和unconfirmedBalance）
371         "get /getPublicKey": "getPublickey",//返回公钥（仅返回PublicKey,不含Sec_Pk）
372         "post /generatePublicKey": "generatePublickey",//生成，设定并返回Publickey
373         "get /delegates": "getDelegates",//返回受托人的信息（并不知道返回了多少信息）
374         "get /delegates/fee": "getDelegatesFee",//返回受托人费用（似乎是一个定值1*constants.fixedPoints）
375         "put /delegates": "addDelegates",//添加受托人（代码较长，没看）
376         "get /username/get": "getUsername",//看方法似乎是通过username返回符合的人的所有信息（1050行）
377         "get /username/fee": "getUsernameFee",//返回值同getDelegatesFee
378         "put /username": "addUsername",//添加用户名，同时生成该用户的全部信息并入库
379         "get /": "getAccount"//根据用户地址查找用户信息，返回全部信息
380     });
381
```

Open 函数的返回结果：

```
547                 accountData = {
548                     address: account.address,
549                     unconfirmedBalance: account.u_balance,
550                     balance: account.balance,
551                     publicKey: account.publicKey,
552                     unconfirmedSignature: account.u_secondSignature,
553                     secondSignature: account.secondSignature,
554                     secondPublicKey: account.secondPublicKey,
555                     multisignatures: account.multisignatures,
556                     u_multisignatures: account.u_multisignatures
557                 };
558
```

Open 方法返回的结果可以看出一个用户对象在数据库内存放的所有信息。

# 关于 Hash 地址

源码文件：modules/accounts.js

```
455 Accounts.prototype.generateAddressByPublicKey = function (publicKey) {
456     var publicKeyHash = crypto.createHash('sha256').update(publicKey, 'hex').digest();
457     var temp = new Buffer(8);//该部分Buffer的长度存疑，因为public是一个64位长度的16进制数
458     for (var i = 0; i < 8; i++) {
459         temp[i] = publicKeyHash[7 - i];//把publicKeyHash反转存储
460     }
461     //结尾用'L'作为亿书区块链地址的标记,bignum部分作用存疑
462     var address = bignum.fromBuffer(temp).toString() + 'L';
463     if (!address) {
464         throw Error("wrong publicKey " + publicKey);
465     }
466     return address;
467 };
468
```

通过加密算法 SHA256 生成 Hash 地址，并加上'L'作为亿书区块链的专用地址标记。

关于 buffer：

1、 new Buffer(8)是创建一个长度为 8 的 buffer 实例，在 buffer 实例中，每一个子元素存储的方式是十六进制，例子如下：

Example:

```
const buf = Buffer.alloc(5, 'a');

// Prints: <Buffer 61 61 61 61 61>
console.log(buf);
```

# 关于交易时提供的信息和注册用户时提供的信息的不同

***注册用户和进行交易都是通过新建交易记录的方法来实现的；**

***两者最主要的区分属性：type**

注册用户时交易记录上的数据：

```
983                     try {
984                         var transaction = library.logic.transaction.create({
985                             type: TransactionTypes.USERNAME,
986                             username: body.username,
987                             sender: account,
988                             keypair: keypair,
989                             secondKeypair: secondKeypair
990                         });
```

进行交易时提供的数据：

```
763                         var transaction = library.logic.transaction.create({
764                             type: TransactionTypes.SEND,
765                             amount: body.amount,
766                             sender: account,
767                             recipientId: recipientId,
768                             recipientUsername: recipientUsername,
769                             keypair: keypair,
770                             requester: keypair,
771                             secondKeypair: secondKeypair
772                         });
```

!：进行交易的时候，查找发起交易的用户的方法有两种，一种是通过多重签名用户公钥的属性(MutlisigAccountPublickey)查找，另一种是通过用户上传的内容中的 secret 属性进行查找，前者的优先度最高。