[1. 注册 1](#_Toc488763818)

[2. 登录(login) 2](#_Toc488763819)

[3. 修改 2](#_Toc488763820)

[4. 找回密码 3](#_Toc488763821)

[5. 上传头像 4](#_Toc488763822)

[6. 获得验证码 4](#_Toc488763823)

# 注册

**Url：** http://domain/user

**输入参数**

Values:{

“recInfo”:{

name:{value:”1234”},

account:{value:”asdf”},

Password:{value:”asdf”},

},

“method”:”0”,

}

**步骤：**

1. 检查用户状态（NO\_SESS/NOT\_LOGIN/LOGIN)
2. 检查输入参数中part的格式和值
3. 检查输入参数是否正确
4. 如果有unique的字段，预先检查；如果有外键，还需要检查docStatus（记录是否成功创建）

4.1 如果有成功创建记录，返回重复错误

4.2 如果记录状态是PENDING，则删除记录以及可能存在的外键记录（user，sugar，**User\_friend\_group**都插入记录，但是没有更新user的状态为DONE），然后重新开始创建过程。

5. 将当前account插入usedAccount，并设置lastAccountUpdateDate

1. 产生sugar
2. 用sugar加密password，替换原始的password
3. 对server产生的字段进行检查（dev only，pro not use）
4. 将用户信息加入users，置docStatus=PENDING。
5. 将sugar加入userSugar
6. 更新users，将docStatus=DONE
7. 使用新生成记录的objectId，在**User\_friend\_group**（朋友分组）中新建一个默认记录

{name:’我的好友’, userId: objectId, firendsInGroup:[]}

# 登录(login)

**Url：** http://domain/user

1. 采用modal：2个input（用户名和密码），一个checkbox（记住密码）

2. db：同“注册”

3. 输入参数

Values:{

“recordInfo”:{

name:{value:”1234”},

Account:{value:’123’},

Password:{value:”asdf”},

},

“curColl”:”users”,

method:4,

}

4. 步骤

4.1 检查格式

4.2 检查值

4.3 从usersSugar读取sugar，对密码进行加密，然后读取users中的password，进行比较。

4.4 如果比较一致，，产生sessionId，并存入redis

# 修改

**Url：** http://domain/user

**输入参数**

Values:{

“recodInfo”:{

name:{value:”1234”},

account:{value:”asdf”},

Password:{value:”asdf”},

},

“method”:”2”,

}

**步骤：**

1. 检查用户状态（NO\_SESS/NOT\_LOGIN/LOGIN)
2. 检查输入参数中part的格式和值
3. 检查输入参数是否正确
4. 如果输入数据有password，从session中获得userId，然后在db中获得sugar，进行hash操作，并赋给输入数据
5. 根据session中的userId，在db中获得对应的记录（docStatus必须为done）
6. 输入数据和db记录比较，如果是相同的，直接删除字段（因为无需更新到db中）
7. 如果有unique的字段，预先检查；如果有外键，还需要检查docStatus（记录是否成功创建）
8. 如果有account，首先判断account是否已经存在usedAccount，是，结束。否，usedAccount的长度是否超出定义，是，将开头的元素删除直接小于定义；否，将当前account插入usedAccount，并设置lastAccountUpdateDate
9. 产生sugar
10. 用sugar加密password，替换原始的password
11. 对server产生的字段进行检查（dev only，pro not use）
12. 将用户信息加入users，置docStatus=PENDING。
13. 将sugar加入userSugar
14. 更新users，将docStatus=DONE
15. 使用新生成记录的objectId，在**User\_friend\_group**（朋友分组）中新建一个默认记录

# 找回密码

**Url：** http://domain/user/retrieve

**输入参数**

Values:{

“singleField”:{

account:{value:”asdf”},

},

}

步骤：

1. 查找account（且docStatus=DONE），是否有匹配用户
2. 如果step1没有找到记录，查找usedAccount（且docStatus=DONE），是否有匹配记录
3. 如果step2没有，直接返回{rc:0}，告知用户已经发送新密码，而不是告知未找到（防止恶意猜测用户名）
4. 找到记录的话，通过generateRandomString函数产生6位长的密码，hash后更新到db，然后发送到account，并返回{rc:0}

# 上传头像

**Url：** http://domain/user/uploadPhoto

**输入参数: NA（只需要req）**

步骤：

1. 检查req中的userId是否存在，不存在，报错

2. 获得上传文件，并做出对应的检查（height/width不合格，直接返回错误（因为如果是cilent传入，必定是正确，不正确，说明是恶意攻击））；如果更是不正确，转换

3. 上传文件名进行md5转换，然后保存到指定目录，并保存到db中（md5名，不包含目录）

# 获得验证码

**Url：** http://domain/user/captcha

**输入参数: NA（只需要req）**

**firstTime:session**中，第一次产生captcha的时间,

**lastTime**：session中，最近一次产生的时间，

**firstTimeInDuration**: duration中，第一次captcha的时间

**numberInDuration**:在定义的时间段中，产生的次数

步骤：

1. 检测req.session.captcha，如果是空，设置所有4个参数

2. 如果存在lastTime，那么和现在的时间比较，是否2次请求的间隔在定义的时间内

3. 如果存在firstTimeInDuration，则和当前的时间间隔是超出duration。没有超出（还在duration），则**numberInDuration**是否超出预定义的数量，超出，报错推出，没有超出，**numberInDuration+1**；否，开始新周期，重新初始化3个参数

4. 产生长度为4的随机字符，存入redis，并生成图片，发给client