**BFC帐号系统SDK**

BFC小组 制定

2014-08-05

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **发布日期** | **主要内容** | **负责人** |
| V1.0 | 2014-10-15 | 毛坯版本，只提供了用户名注册、登录、登出、自定义事件功能 | 李超 |
| V1.1 | 2014-11-26 | Jar包添加工具方法：检查用户名、检查验证码 | 李超 |
|  | 2014-12-29 | 添加接口说明 | 邓俊辉 |

目录

目录 3

一、 版本信息 4

二、 适用机型 4

三、 依赖环境 4

四、 原理简介 4

五、 具体方法 4

（一） 获取并设置AccountManager 实例 5

（二） 帐号操作 5

（三） 通用接口的特殊操作 7

（四） 管理器销毁 8

六、 工具方法 8

（一）检查用户名是否可用 8

（二）检查图片验证码是否正确 9

（三）获取图片验证码 9

七、满足Studyos需求添加一下请求接口和模式 10

1. 在当前帐号下增加子用户 10

2. 获取某一个一级账号的所有用户块 10

3. 获在一级账户下从一级用户块切换到另一个用户块 10

4. 获取一级帐号下的当前用户帐号信息 11

5. 设置一级帐号下的当前用户帐号信息 11

附录一:键值表格 11

附录二：键值源码 15

# 版本信息

版本号：V1.0

# 适用机型

Android通用机型，包括第三方android手机平台

# 依赖环境

* 需要安装BFCPackage.apk(10-15以上)，提供帐号系统服务及相关功能；
* 需要将accountManager.jar 放在应用工程libs目录下，打包到apk中；
* 需要将BFCUtils.jar 放到应用工程libs目录下，功能包使用了其BFC版本检查的功能；

# 原理简介

* 帐号系统维护了一个帐号服务。其中主要有服务器对接的功能类和当前的用户帐号配置管理，帐号的注册、登录、登出等基本操作由功能类异步线程完成，服务器返回结果以广播的形式发出；自动完成帐号配置的初始化、更新，协助完成自动登录、异常恢复和离线登录等功能；
* accountManager.jar 主要封装了对 帐号服务的绑定和解绑、对帐号操作广播的注册、监听和注销，同时维护着帐号功能相关的AIDL功能接口，来实现各种操作功能；
* 应用通过设置accountManager.jar提供的 IAccountListener 来完成对帐号操作的结果处理；
* 服务主要提供公司级帐号系统的注册、登录登出操作，同时提供网络通用接口，主要针对部门级帐号子系统操作；二者信息传递的流程是一致的；

# 具体方法

下面以注册一个帐号为例，描述一下 accountManager.jar 各接口的使用方法。

## 获取并设置AccountManager 实例

注意，这里需要捕获异常，检查BFC组件是否支持帐号系统，如果版本过低或不存在，需要提示升级或安装BFC组件。

//第一步：根据获取单例实体的接口，拿到管理器实体；

try {

mAccountManager = AccountManager

.getInstance(getApplicationContext());

} catch (VersionException e) {

Log.e(TAG, "" + e.getCode());

e.printStackTrace();

}

//第二步：设置信息返回的监听接口

mAccountManager.registerListener(mIAccountListener);

## 帐号操作

用户需要根据 accountManager.jar 定义的关键字信息（具体见附录），组成一个参数表，通过enqueueRequest接口传递进入；后面还需要通过这些关键字分析出返回的结果，从而进行不同的处理。

//第一步：生成一个操作参数表

HashMap<String,String> paramMap = new HashMap<String,String>();

// 操作类型 操作码是：注册

paramMap.put(AccountInfo.KEY\_REQTYPE,

AccountInfo.VALUE\_REQTYPE\_REGISTER);

// 用户名

paramMap.put(AccountInfo.KEY\_USERNAME, “username”);

// 用户密码

paramMap.put(AccountInfo.KEY\_PASSWORD, “password”);

//第二步：插入操作到服务

mAccountManager.enqueueRequest(paramMap);

//在结果监听中分析返回的参数表

IAccountListener mIAccountListener = new IAccountListener() {

//返回接口

public void onAccountEvent(Map<String, String> paramMap) {

String respType = paramMap.get(AccountInfo.KEY\_RESPTYPE);

Log.d("BFCDemo","onAccountEvent:"+respType);

//处理返回类型是注册的结果

if (respType.equals(AccountInfo.VALUE\_RESPTYPE\_REGISTER)){

//取出返回码和用户名

StringrespCode=

paramMap.get(AccountInfo.KEY\_RESPCODE).toString();

String userName = paramMap.get(AccountInfo.KEY\_USERNAME).toString();

if(respCode.equals(AccountInfo.VALUE\_RESPCODE\_SUCCESS)){

// 注册成功

Log.d("BFCDemo","register username success:"+userName);

Toast.makeText(getApplicationContext(), "register username success:"+userName,Toast.LENGTH\_SHORT).show();

} else {

// 注册失败

Log.e("BFCDemo","register username fail:"+userName);

}

}

}

};

## 通用接口的特殊操作

通用接口只是负责固定的网络通道，APP与部门级帐号子系统自行沟通信息传递结构，通过此接口以登录等同样的信息传递流程进行通信。示例如下：

private void getCurrentUserInfo() {

Log.d(TAG, "getCurrentUserInfo~");

HashMap<String, String> paramMap = new HashMap<String, String>();

// 操作类型

paramMap.put(AccountInfo.KEY\_REQTYPE, AccountInfo.VALUE\_REQTYPE\_COMMON);

//是否是对当前登录用户操作

paramMap.put(AccountInfo.KEY\_CURRENT\_ACCOUNT\_OPERATE, "true");

//url 后面的"/UserInfoSystem/user/getUserInfo"是需要同服务器协商的接口

paramMap.put(AccountInfo.KEY\_COMMON\_URL, AccountInfo.KF2\_SERVER\_URL+"/UserInfoSystem/user/getUserInfo");

// 插入操作到服务

mAccountManager.enqueueRequest(paramMap);

}

在IAccountListener的回调中处理返回值，拿到返回数据：

if (respType.equals(AccountInfo.VALUE\_RESPTYPE\_COMMON)) {

String respCode = paramMap.get(AccountInfo.KEY\_RESPCODE).toString();

if (respCode.equals(AccountInfo.VALUE\_RESPCODE\_SUCCESS)) {

// 成功

String resultString = paramMap.get(AccountInfo.KEY\_COMMON\_GET\_DATE).toString();

ResultPojo<UserBasicInfo> result = mGsonTool.fromJson(resultString,

new TypeToken<ResultPojo<UserBasicInfo>>() {

}.getType());

} else {

// 失败

String errorCode = paramMap.get(AccountInfo.KEY\_ERROR\_CODE).toString();

Log.e(TAG, "common test fail " + respCode + " errorCode:" + errorCode);

Toast.makeText(getApplicationContext(), "common test fail==== " + errorCode, Toast.LENGTH\_SHORT)

.show();

}

}

## 管理器销毁

//在 onDestroy 回调中调用管理器的destroy接口即可

public void onDestroy(){

super.onDestroy();

mAccountManager.destroy();

}

# 六、 工具方法

## （一）检查用户名是否可用

所在类：com.eebbk.bfc.sdk.account. AccountTool

/\*\*

\* 检查用户名是否可用 注意：本方法涉及网络操作 请单独开线程使用

\* @param context 上下文

\* @param checkName 被检查的用户名

\* @return AccountTool.CHECK\_FAIL 检查失败

\* CHECK\_NAME\_USEFUL 用户名可用

\* CHECK\_NAME\_ALREADY\_EXIST 用户名已存在

\* AccountInfo.VALUE\_ERROR\_SERVER\_OTHER 服务器返回异常

\*/

public static String checkUserNameAvailable(Context context, String checkName)

## （二）检查图片验证码是否正确

所在类：com.eebbk.bfc.sdk.account. AccountTool

/\*\*

\* 检查验证码是否正确 注意：本方法涉及网络操作 请单独开线程使用

\* @param context 上下文

\* @param randcodeKey 验证码图片键值

\* @param randcode 验证码答案

\* @return AccountTool.CHECK\_FAIL 检查失败

\* CHECK\_CAPTCHA\_RIGHT 验证码正确

\* CHECK\_CAPTCHA\_WRONG 验证码错误

\* AccountInfo.VALUE\_ERROR\_SERVER\_OTHER 服务器返回异常

\*/

public static String checkCaptchaRight(Context context, String randcodeKey,String randcode)

## （三）获取图片验证码

所在类：com.eebbk.bfc.sdk.account. CaptchaTool

/\*\*

\* 涉及到网络请求 使用该函数请单独开线程

\* @param width 验证码宽度

\* @param height 验证码高度

\* @return 验证码结构

\*/

public Captcha captchaGet(int width,int height)

## （四）获取手机验证码和检查手机验证码 同上

# 七、满足Studyos需求添加一下请求接口和模式

### 1. 在当前帐号下增加子用户

请求键值

(AccountInfo.KEY\_REQTYPE,AccountInfo.VALUE\_REQTYPE\_ADD\_CHILD\_USERINFO);

添加的用户名

paramMap.put(AccountInfo.KEY\_CHILD\_USERNAME, "djh\_A");

返回说明（在监听返回中对以下字段进行监听）

if (respType.equals(AccountInfo.VALUE\_RESPTYPE\_ADD\_CHILD\_USERINFO)) {

String respCode1 = paramMap.get(AccountInfo.KEY\_RESPCODE).toString();

if (respCode1.equals(AccountInfo.VALUE\_RESPCODE\_SUCCESS)) {

// 成功

Log.d(TAG, "增加succsess~~");

} else {

// 失败 Log.e(TAG, "增加子节点 fail~ " + errorCode);

}

### 2. 获取某一个一级账号的所有用户块

请求键值

(AccountInfo.KEY\_REQTYPE,AccountInfo.VALUE\_REQTYPE\_GET\_CHIRLD\_USER);

如何取返回值

请在请求成功监听返回后，试用以下来得到所有信息

if (respType.equals(AccountInfo.VALUE\_RESPTYPE\_GET\_CHIRLD\_USER)) {

String respCode = paramMap.get(AccountInfo.KEY\_RESPCODE).toString(); if (respCode.equals(AccountInfo.VALUE\_RESPCODE\_SUCCESS)) {

List<UserInfoVo> userList =AccountTool.getUserInfoVoList();

｝

### 3. 获在一级账户下从一级用户块切换到另一个用户块

请求键值

(AccountInfo.KEY\_REQTYPE, AccountInfo.VALUE\_REQTYPE\_SWITCH\_CHILD\_USER)

返回说明

if (respType.equals(AccountInfo.VALUE\_RESPTYPE\_SWITCH\_CHILD\_USER)) {

String respCode = paramMap.get(AccountInfo.KEY\_RESPCODE).toString();

if (respCode.equals(AccountInfo.VALUE\_RESPCODE\_SUCCESS)) {

//成功

｝

### 4. 获取一级帐号下的当前用户帐号信息

请求键值

AccountInfo.KEY\_REQTYPE, AccountInfo.VALUE\_REQTYPE\_GET\_USERINFO

AccountInfo.KEY\_CHILD\_REQTYPE, AccountInfo.VALUE\_REQTYPE\_GET\_CHILD\_USERINFO

请求返回

在返回监听成功中对返回的MAP获取键值得到当前用户信息。

Iterator<String> iter = paramMap.keySet().iterator();

while (iter.hasNext()) {

String key = iter.next();

String value = paramMap.get(key);

Log.v(TAG, "VALUE\_RESPTYPE\_GET\_USERINFO " + key + " " + value);

}

### 5. 设置一级帐号下的当前用户帐号信息

请求键值（并自行填写用户信息键值码）

AccountInfo.KEY\_REQTYPE, AccountInfo.VALUE\_REQTYPE\_UPDATE\_USERINFO

AccountInfo.KEY\_CHILD\_REQTYPE,AccountInfo.VALUE\_REQTYPE\_UPDATE\_CHILD\_USERINFO

返回说明

if (respType.equals(AccountInfo.VALUE\_RESPTYPE\_UPDATE\_USERINFO)) {

if (AccountInfo.VALUE\_RESPCODE\_SUCCESS.equals(respCode)) {

Log.d(TAG, "更新用户信息成功");

} else {

Log.d(TAG, "更新用户信息失败");

}

}

### 6. 关于监听设置的改变

可以注册多个监听，请在发送请求是声明你注册的哪个监听，如果没有声明默认全发（及所有监听器都能收到）。

mAccountManager.registerListener(mBIAccountListener);

IAccountListener mBIAccountListener = new IAccountListener() {

public void onAccountEvent(Map<String, String> paramMap) {

}

public String getAccountListenerId() {

return "B";

};

如你注册的是B监听，需要只是B监听接收在请求时 请增加如下请求。

paramMap.put(AccountInfo.KEY\_LISTENER\_ID,"B");

# 附录一:键值表格

请求类别如下：

|  |  |
| --- | --- |
| **请求类别** | **请求码** |
| 用户注册 | REQUEST\_TYPE\_REGISTER |
| 用户登录 | REQUEST\_TYPE\_LOGIN |
| 用户登出 | REQUEST\_TYPE\_LOGOUT |
| 是否有登陆 | REQUEST\_TYPE\_HAS\_LOGIN |
| 修改密码 | *REQTYPE\_CHANGE\_PASSWORD* |
| 设置用户信息 | REQUEST\_TYPE\_SET\_USERINFO |
| 得到用户信息 | REQUEST\_TYPE\_GET\_USERINFO |
| 重置密码 | REQUEST\_RESET\_PASSWORD |
| 通用请求 | REQTYPE\_COMMON |
|  |  |

响应类别如下：

|  |  |
| --- | --- |
| **响应类别** | **响应码** |
| 用户注册 | RESPONSE\_TYPE\_REGISTER |
| 用户登录 | RESPONSE\_TYPE\_LOGIN |
| 用户登出 | RESPONSE\_TYPE\_LOGOUT |
| 是否有登陆 | RESPONSE\_TYPE\_HAS\_LOGIN |
| 修改密码 | RESPONSE\_TYPE\_MODIFY\_PASSWORD |
| 设置用户信息 | RESPONSE\_TYPE\_SET\_USERINFO |
| 得到用户信息 | RESPONSE\_TYPE\_GET\_USERINFO |
| 重置密码 | RESPONSE\_RESET\_PASSWORD |
| 通用请求响应 | RESPONSE\_TYPE\_COMMON |

错误码类别如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 服务器返回码 | **BFC返回码** | **说明** | **备注** |
| 010101 | VALUE\_ERROR\_CAPTCHA | 验证码错误 | 注册 |
| 010102 |  | 短信或邮箱验证码超出发送次数限制 |  |
| 010201 |  | 用户名为空 |  |
| 010202 | VALUE\_ERROR\_NAME\_ALREADY\_EXIST | 用户名已存在 | 注册 |
| 010203 |  | 用户名不存在 |  |
| 010204 |  | 用户名无效 |  |
| 010301 |  | 密码为空 |  |
| 010302 |  | 密码不符合规范 |  |
| 010303 |  | 旧密码无效 |  |
| 010304 |  | checkcode无效 |  |
| 010401 |  | 签名无效 |  |
| 010402 | VALUE\_ERROR\_NAME\_PASSWORD | 用户名或密码错误 |  |
| 010403 |  | 登录状态无效 |  |
| 010404 |  | 登录状态超时 |  |
| 000001 | VALUE\_RESPCODE\_SUCCESS | 操作成功标识 | KEY\_RESPCODE |
| 000002 | VALUE\_ERROR\_SERVER\_OTHER | 其他错误 |  |

用户信息键值码：UserInfo

|  |  |
| --- | --- |
| *KEY\_CHILD\_USERNAME* | *子用户名* |
| *KEY\_ACCOUNT\_ID* | *帐号ID* |
| *KEY\_LOGIN\_MODE* | *登录模式* |
| *KEY\_OLD\_PASSWORD* | *修改密码使用 旧密码 键值* |
| *KEY\_NEW\_PASSWORD* | *修改密码使用 新密码 键值* |
| *KEY\_REGIST\_NAME* | *注册名称键值 对应的值* |
| *KEY\_LOGIN\_NAME* | *登录名称键值* |
| *KEY\_PASSWORD* | *用户密码键值* |
| *KEY\_LOGIN\_COUNT* | *登录次数* |
| *KEY\_USERALIAS* | *昵称* |
| *KEY\_AVATAR* | *个人信息键值 用户头像 个人信息直接做服务器参数使用* |
| *KEY\_TELEPHONE* | *电话号* |
| *KEY\_LOCATION\_PROVINCE* | *省* |
| *KEY\_LOCATION\_CITY* | *市* |
| *KEY\_LOCATION\_DISTRICT* | *区* |
| *KEY\_BIRTHDAY\_YEAR* | *生日的年* |
| *KEY\_BIRTHDAY\_MONTH* | *月* |
| *KEY\_BIRTHDAY\_DAY* | *日* |
| *KEY\_QQ* | *QQ* |
| *KEY\_SEX* | *个人信息键值 用户性别* |
| *KEY\_SCHOOL* | *学校* |
| *KEY\_GRADE* | *年级* |
| *KEY\_PRESS* | *版本信息* |
| *KEY\_PERSONAL\_SIGN* | *标志* |
| *KEY\_CURRENT\_ACCOUNT\_OPERATE* | *是否为对当前登录用户操作 该键对应的值应当是BOOL型的字符串表示* |
| *KEY\_COMMON\_URL* | *通用接口的地址键* |
| *KEY\_COMMON\_GET\_DATE* | *通用接口 请求返回的数据* |
| *KEY\_CAPTCHA\_ID* | *验证码ID 键值* |
| *KEY\_CAPTCHA\_PICTURE\_ANSWER* | *图片验证码答案 键值* |
| *KEY\_CAPTCHA\_MOBILE\_ANSWER* | *手机验证码答案 键值* |
| *KEY\_MACHINE\_ID* | *机器码* |
|  |  |
|  |  |

# 附录二：键值源码

package com.eebbk.bfc.module.account.share;

public class AccountInfo {

**public** **static** **final** String *KF2\_SERVER\_URL* = "http://172.28.14.109:8080";

/\*\*

\* 请求类型键值

\*/

**public** **static** **final** String *KEY\_REQTYPE* = "REQUESTTYPE";

/\*\*

\* 请求类型：注册

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_REGISTER* = "REQTYPE\_REGISTER";

/\*\*

\* 请求类型：登录

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_LOGIN* = "REQTYPE\_LOGIN";

/\*\*

\* 请求类型：登出

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_LOGOUT* = "REQTYPE\_LOGOUT";

/\*\*

\* 请求类型：查询登录状态

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_HAS\_LOGIN* = "REQTYPE\_HAS\_LOGIN";

/\*\*

\* 通用请求类型 用户自定义类型

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_COMMON* = "REQTYPE\_COMMON";

/\*\*

\* 获取用户基本信息

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_GET\_USERINFO* = "REQTYPE\_GETUSERINFO";

/\*\*

\* 设置用户基本信息

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_UPDATE\_USERINFO* = "REQTYPE\_UPDATEUSERINFO";

/\*\*

\* 修改密码 对应参数键值有：KEY\_OLD\_PASSWORD、KEY\_NEW\_PASSWORD、LOGIN\_NAME

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_CHANGE\_PASSWORD* = "REQTYPE\_CHANGE\_PASSWORD";

/\*\*

\* 重置密码 对应键值有：KEY\_REGIST\_NAME、KEY\_MOBILEPHONE、KEY\_NEW\_PASSWORD、KEY\_CAPTCHA\_MOBILE\_ANSWER

\*/

**public** **static** **final** String *VALUE\_REQTYPE\_RESET\_PASSWORD* = "REQTYPE\_RESET\_PASSWORD";

/\*\*

\* 请求模式 对应值 VALUE\_REQ\_MODE\_LOCAL 本地<br>

\* VALUE\_REQ\_MODE\_REMOTE 远程<br>

\* 在查询登录、与获取用户信息时有效

\*/

**public** **static** **final** String *KEY\_REQ\_MODE* = "REQUEST\_MODE";

/\*\*

\* 本地信息请求

\*/

**public** **static** **final** String *VALUE\_REQ\_MODE\_LOCAL* = "LOCAL\_REQUEST";

/\*\*

\* 远程信息请求

\*/

**public** **static** **final** String *VALUE\_REQ\_MODE\_REMOTE* = "REMOTE\_REQUEST";

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 响应码 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// 响应码键值

**public** **static** **final** String *KEY\_RESPCODE* = "RESPONSETCODE";

// 响应码值：成功

**public** **static** **final** String *VALUE\_RESPCODE\_SUCCESS* = "RESPONSETCODE\_SUCCESS";

// 响应码值：失败

**public** **static** **final** String *VALUE\_RESPCODE\_FAIL* = "RESPONSETCODE\_FAIL";

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 响应类型 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// 响应类型键值

**public** **static** **final** String *KEY\_RESPTYPE* = "RESPONSETTYPE";

// 响应类型值：注册

**public** **static** **final** String *VALUE\_RESPTYPE\_REGISTER* = "RESPONSE\_TYPE\_REGISTER";

// 响应类型值：登录

**public** **static** **final** String *VALUE\_RESPTYPE\_LOGIN* = "RESPONSE\_TYPE\_LOGIN";

// 响应类型值：登出

**public** **static** **final** String *VALUE\_RESPTYPE\_LOGOUT* = "RESPONSE\_TYPE\_LOGOUT";

// 响应类型值：通用类型响应

**public** **static** **final** String *VALUE\_RESPTYPE\_COMMON* = "RESPONSE\_TYPE\_COMMON";

**public** **static** **final** String *VALUE\_RESPTYPE\_HAS\_LOGIN* = "RESPONSE\_TYPE\_HAS\_LOGIN";

**public** **static** **final** String *VALUE\_RESPTYPE\_MODIFY\_PASSWORD* = "RESPONSE\_TYPE\_MODIFY\_PASSWORD";

**public** **static** **final** String *VALUE\_RESPTYPE\_UPDATE\_USERINFO* = "RESPONSE\_TYPE\_SET\_USERINFO";

**public** **static** **final** String *VALUE\_RESPTYPE\_GET\_USERINFO* = "RESPONSE\_TYPE\_GET\_USERINFO";

**public** **static** **final** String *VALUE\_RESPTYPE\_RESET\_PASSWORD* = "RESPONSE\_RESET\_PASSWORD";

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 数据键值 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* 登录模式 检查登录世返回

\*/

**public** **static** **final** String *KEY\_LOGIN\_MODE* = "loginMode";

/\*\*

\* 无人登录

\*/

**public** **static** **final** String *VALUE\_MODE\_NONE\_LOGIN* = "noneLogin";

/\*\*

\* 已登录

\*/

**public** **static** **final** String *VALUE\_MODE\_ONLINE* = "online";

/\*\*

\* 离线登录

\*/

**public** **static** **final** String *VALUE\_MODE\_OFFLINE* = "offline";

/\*\*

\* 修改密码使用 旧密码 键值

\*/

**public** **static** **final** String *KEY\_OLD\_PASSWORD* = "oldPassword";

/\*\*

\* 修改密码使用 新密码 键值

\*/

**public** **static** **final** String *KEY\_NEW\_PASSWORD* = "newPassword";

/\*\*

\* 注册名称键值 对应的值可以是手机、邮箱、用户名

\*/

**public** **static** **final** String *KEY\_REGIST\_NAME* = "userName";

/\*\*

\* 登录名称键值 对应的值可以是手机、邮箱、用户名

\*/

**public** **static** **final** String *KEY\_LOGIN\_NAME* = "loginName";

/\*\*

\* 用户密码键值

\*/

**public** **static** **final** String *KEY\_PASSWORD* = "password";

/\*\*

\* 登录次数

\*/

**public** **static** **final** String *KEY\_LOGIN\_COUNT* = "count";

/\*\*

\* 昵称

\*/

**public** **static** **final** String *KEY\_USERALIAS* = "userAlias";

/\*\*

\* 个人信息键值 用户头像 个人信息直接做服务器参数使用

\*/

**public** **static** **final** String *KEY\_AVATAR* = "headPortrait";

/\*\*

\* 个人信息键值 用户性别

\*/

**public** **static** **final** String *KEY\_SEX* = "sex";

/\*\*

\* 个人信息 手机号

\*/

**public** **static** **final** String *KEY\_MOBILEPHONE* = "telephone";

/\*\*

\* 个人信息键值 用户位置 省市区

\*/

**public** **static** **final** String *KEY\_LOCATION\_PROVINCE* = "province";

**public** **static** **final** String *KEY\_LOCATION\_CITY* = "city";

**public** **static** **final** String *KEY\_LOCATION\_DISTRICT* = "district";

/\*\*

\* QQ号

\*/

**public** **static** **final** String *KEY\_QQ* = "qq";

/\*\*

\* 生日 年

\*/

**public** **static** **final** String *KEY\_BIRTHDAY\_YEAR* = "birthday\_year";

/\*\*

\* 生日 月

\*/

**public** **static** **final** String *KEY\_BIRTHDAY\_MONTH* = "birthday\_month";

/\*\*

\* 生日 日

\*/

**public** **static** **final** String *KEY\_BIRTHDAY\_DAY* = "birthday\_day";

/\*\*

\* 个人信息键值 用户学校

\*/

**public** **static** **final** String *KEY\_SCHOOL* = "school";

/\*\*

\* 个人信息键值 用户年级

\*/

**public** **static** **final** String *KEY\_GRADE* = "grade";

/\*\*

\* 个人信息键值 出版社信息

\*/

**public** **static** **final** String *KEY\_PRESS* = "coursePublishAgency";

/\*\*

\* 个人信息

\*/

**public** **static** **final** String *KEY\_PERSONAL\_SIGN* = "personalSign";

/\*\*

\* 是否为对当前登录用户操作 该键对应的值应当是BOOL型的字符串表示

\*/

**public** **static** **final** String *KEY\_CURRENT\_ACCOUNT\_OPERATE* = "CURRENT\_ACCOUNT\_OPERATE";

/\*\*

\* 通用接口的地址键

\*/

**public** **static** **final** String *KEY\_COMMON\_URL* = "COMMON\_URL";

/\*\*

\* 通用接口 请求返回的数据

\*/

**public** **static** **final** String *KEY\_COMMON\_GET\_DATE* = "COMMON\_GET\_DATE";

/\*\*

\* 验证码ID 键值

\*/

**public** **static** **final** String *KEY\_CAPTCHA\_ID* = "codeKey";

/\*\*

\* 图片验证码答案 键值

\*/

**public** **static** **final** String *KEY\_CAPTCHA\_PICTURE\_ANSWER* = "pictureCode";

/\*\*

\* 手机验证码答案 键值

\*/

**public** **static** **final** String *KEY\_CAPTCHA\_MOBILE\_ANSWER* = "telephoneCode";

/\*\*

\* 机器码

\*/

**public** **static** **final** String *KEY\_MACHINE\_ID* = "machineId";

**public** **static** **final** String *KEY\_ACCOUNT\_ID* = "accountId";

/\*\*

\* ACTION

\*/

// 帐号操作结果广播

**public** **static** **final** String *ACCOUNT\_BROADCAST\_ACTION* = "com.eebbk.bfc.module.account.NOTIFY";

// 帐号服务Action

**public** **static** **final** String *ACCOUNT\_SERVICE\_ACTION* = "com.eebbk.bfc.module.account.BIND";

/\*\*

\* 错误码

\*/

**public** **static** **final** String *KEY\_ERROR\_CODE* = "ERROR\_CODE";

/\*\*

\* 验证码错误

\*/

**public** **static** **final** String *VALUE\_ERROR\_CAPTCHA* = "CAPTCHA\_ERROR";

/\*\*

\* 用户名已存在 注册错误

\*/

**public** **static** **final** String *VALUE\_ERROR\_NAME\_ALREADY\_EXIST* = "NAME\_ALREADY\_EXIST\_ERROR";

/\*\*

\* 用户名或密码错误

\*/

**public** **static** **final** String *VALUE\_ERROR\_NAME\_PASSWORD* = "NAME\_PASSWORD\_ERROR";

**public** **static** **final** String *VALUE\_ERROR\_NAME\_PASSWORD\_NULL* = "NAME\_PASSWORD\_IS\_NULL";

/\*\*

\* 服务器返回空

\*/

**public** **static** **final** String *VALUE\_ERROR\_SERVER\_RETNULL* = "SERVER\_RETNULL\_ERROR";

/\*\*

\* 服务器其它类型错误

\*/

**public** **static** **final** String *VALUE\_ERROR\_SERVER\_OTHER* = "SERVER\_OTHER\_ERROR";

/\*\*

\* 尚未登录错误

\*/

**public** **static** **final** String *VALUE\_ERROR\_NO\_LOGIN* = "NO\_LOGIN";

/\*\*

\* 未匹配到

\*/

**public** **static** **final** String *VALUE\_ERROR\_UNKNOWN* = "UNKNOWN\_ERROR";

**public** **static** **final** String *VALUE\_ERROR\_LOGIN\_TIMEOUT* = "LOGIN\_TIMEOUT";

**public** **static** **final** String *VALUE\_ERROR\_NO\_AUTHORITY* = "NO\_AUTHORITY";

**public** **static** **final** String *VALUE\_ERROR\_ILLEGAL\_ARGUMENT* = "ILLEGAL\_ARGUMENT";

**public** **static** **final** String *VALUE\_ERROR\_TIMEOUT* = "TIMEOUT";

**public** **static** **final** String *VALUE\_ERROR\_SERVER\_IO\_EXCEPTION* = "SERVER\_IO\_EXCEPTION";

**public** **static** **final** String *VALUE\_ERROR\_SERVER\_RUNRIME\_EXCEPTION* = "SERVER\_RUNRIME\_EXCEPTION";

**public** **static** **final** String *VALUE\_ERROR\_SERVER\_DATABASE\_EXCEPTION* = "SERVER\_DATABASE\_EXCEPTION";

**public** **static** **final** String *VALUE\_ERROR\_SERVER\_HTTP\_EXCEPTION* = "SERVER\_HTTP\_REQUEST\_EXCEPTION";

**public** **static** **final** String *VALUE\_ERROR\_SERVER\_CACHE\_EXCEPTION* = "SERVER\_CACHE\_EXCEPTION";

**public** **static** **final** String *VALUE\_ERROR\_SERVER\_OTHER\_ERROR* = "SERVER\_OTHER\_ERROR";

}

# 附录三：用户信息

public class UserInfoVo implements Serializable

{

private static final long serialVersionUID = 1L;

private int accountId;

private String userName;

private String email;

private String mobilePhone;

private int state;// 账户状态 0正常；1永久锁定；2临时锁定

private int parentId;// 父id

private int currentId;//当前id

private int accountInfoId；

private String machineId;

private String qq;

private String telephone;

private String birthday;//1991-08-11

private String sex;

private int role;// 1表示超级管理员 2表示管理员 3表示普通用户

private int temporary;// 临时账户 （0表示正常账户，1表示临时账户）

private int serialNumber;// 临时账户对应的序

private int personCenterId;

private String school;

private String grade；

private String province;

private String city;

private String district;// 地区

private int score;// 积分

private String headPortrait;// 个人头像

private String userAlias;// 用户昵称

private String personalSign;// 个性签名

private String coursePublishAgency;//课程\_出版社\_封面

private String areaId;// 区域Id

private String userIndex;// 子用户下标

private Timestamp createTime; // 创建时

private Timestamp updateTime;// 更新时间

}