```
//
//  Storage.java
//  SoDemon
//
```

#1.本地存储用户管理<获取用户列表/更新用户/删除用户/设置当前用户>

1.本地存储用户管理

说明：以下函数是在需要用户管理才用到的函数，包括用户列表的获取 / 删除用户 / 更新用户 / 设置当前用户

1）Storage的初始化：

说明：初始化存储环境;

函数：public static void init(Context context);

2）本地用户信息获取

说明：通过此函数可以获取到本地的所有用户以及用户信息。

函数：public static ArrayList<USER_INFO> get_users_info();

返回：返回一个USER_INFO类型的列表，每一个user对象都有以下属性，说明如下：

```
String id      //账号的user_id，注册的时候用user_id为tag
```

获取函数：public String getId()

设置函数：public void setId(String id)

```
String alias      //账号昵称
```

获取函数：public String getAlias()

设置函数：public void setAlias(String alias)

```
/*
 *登录类型，共有四种
 *OFFLINE_USER = 0      离线登录，即不用账号直接使用，不使用平台功能
 *ACCOUNT_USER = 1      账号登录
 *QQ_USER = 2           第三方 QQ登录
 *FACEBOOK_USER = 3     第三方 facebook登录
 */
int user_type     //登录类型
```

获取函数：public int getUserType()

设置函数：public void setUserType(int user_type)

```
/*
 *name和pwd，当login_type = ACCOUNT_USER时才用到
 */
String name       //账号名
```

获取函数：public String getName()

设置函数：public void setName(String name)

```
String pwd        //账号密码
```

获取函数：public String getPwd()

设置函数：public void setPwd(String pwd)

```
/*

 *open_id和access_token，使用第三方账号登录时，第三方服务器返回的参数，需成对使用，才能登录成功
 *当login_type = FACEBOOK_USER | QQ_USER 时才用到
 */
String open_id     //openId
```

获取函数：public String getOpenID()

设置函数：public void setOpenID(String open_id)

```
String access_token     //accessToken
```

获取函数：public String getAccessToken()

设置函数：public void setAccessToken(String access_token)

```
/*
 *是否是最新登录，如果是最新登录的，可以保存起来，如果是true,下次可以直接后台登录；
 */
boolean latest      //是否是最新登录
```

获取函数：public boolean getLatest()

设置函数：public void setLatest(boolean latest)

3）更新用户

说明：通过此函数可以更新本地用户信息。当切换用户或登录等，只要登录状态有改变，都需要用到此函数。

参数：

```
String user_id;    //user_ID 是服务器返回来的用户标记码
int user_type; //登录类型
/*
 *如果是OFFLINE_USER类型，name为空
 *如果是ACCOUNT_USER类型，name就是账号名name
 *如果是QQ_USER或FACEBOOK_USER类型，name就是open_id
 */
String name;
/*
 *如果是OFFLINE_USER类型，pwd为空
 *如果是ACCOUNT_USER类型，pwd就是账号名pwd
 *如果是QQ_USER或FACEBOOK_USER类型，pwd就是access_token
 */
String pwd;
String alias;//别名
```

函数：

```
public static void update_user(String user_id, String name, String pwd, int user_type
, String alias)
```

4）清除登录记录及删除账号

说明：以下两个分别为清除最近一次登录记录和删除本地指定账号

函数：

```
public static void clear_user_latest() ; //清除最近一次登录记录，即设置latest为false

public static void remove_user(String user_id); //删除指定的账号
```

参数：

user_id：指定删除的账号user_id

5）设置当前账号

说明：以下函数可以实现，设置指定账号为当前用户，以及获取当前用户的user_id和名称

函数：

```
public static void set_current_user(String user_id); //设置user_id为当前账号

public static String get_current_user_id();//获取当前账号的user_id

public static String get_current_user_alias();//获取当前账号的昵称
```

参数：

user_id：指定设定为当前账号的账号user_id

#2.本地存储摄像机管理 <获取摄像机 / 和服务器比较 / 添加 / 删除 / 更新>

2.本地存储摄像机管理

1）摄像机list获取

说明:以下函数用来获取本地存储的摄像机，以及摄像机的信息

函数:

```
public static ArrayList<CAMERA_INFO> get_cameras();
```

返回：返回一个CAMERA_INFO对象的列表，列表的每一个元素是一个CAMERA_INFO对象，如下：

```
String id    //摄像机id
获取函数：public String getId()
设置函数：public void setId(String id)

String alias     //摄像机昵称
获取函数：public String getAlias()
设置函数：public void setAlias(String alias)

String user     //摄像机登录用户名
获取函数：public String getUser()
设置函数：public void setUser(String user)


String pwd    //摄像机登录密码
获取函数：public String getPwd()
设置函数：public void setPwd(String pwd)

boolean https  //加密安全传输，两个值 0:非加密     1:加密
获取函数：public boolean getHttps()
设置函数：public void setHttps(boolean https)

String obj_id    //摄像机sosocam_id
获取函数：public String getObj_id()
设置函数：public void setObj_id(String obj_id)

byte [] cover    //摄像机预览图data数据
获取函数：public Bitmap getCover()
设置函数：public void SetCover(Bitmap cover)

String model //摄像机模型，两个值，0 云台机  1 卡片机
获取函数：public int getModel()
设置函数：public void setModel(int model)
```

示例：

```
m_ipcams_list = Storage.get_cameras();
```

2）比较

说明：把从服务器上获取到的摄像机和当前账号本地的cams做比较，
如果不一样先进行同步，然后返回true，
如果m_user_id为空，本地的cams和服务器上的cams未做同步工作，且返回false

函数：

```
public static boolean merge_cameras(ArrayList<SERVICE_CAMERA_INFO>
server_cameras_list);
```

参数：server_cameras_list:服务器上获取到的摄像机

返回：返回BOOL值，True:服务器上的cams和本地的cams已进行同步。False:
服务器上的cams和本地的cams未进行同步。

示例：

```
Storage.merge_cameras(WebService.cameras);
```

3）添加

说明：添加摄像机到本地存储。一般添加了摄像机或从服务器上获取到了新摄像机都要存储到本地。这时候就要适用此函数。

函数：

```
public static void add_camera(CAMERA_INFO camera_info);
```

参数：

```
String id         //摄像机id
String alias     //摄像机昵称
String user     //摄像机登录用户名
String pwd      //摄像机登录密码
boolean https   //加密安全传输，两个值 0:非加密     1:加密
String obj_id     //摄像机sosocam_id
String model    //摄像机模型，两个值，0 云台机  1 卡片机
```

示例：

```
Storage.add_camera(add_cam);  //add_cam为CAMERA_INFO类型的对象
Storage.save_cameras();
//添加一个摄像机，删除一个摄像机，更新摄像机等只要与camera有关的，都必须调用此函数进行保存
```

4）删除

说明：此函数用来从本地删除指定的摄像机。

函数：

```
public static void remove_camera(String camera_id);
```

参数：

camera_id：即将被删除的摄像机id

示例：

```
Storage.remove_camera("RTEST-001015-DUGMR");
```

5）更新

说明：以下函数都是用来更新摄像机本地信息的。（名称 / 登录名 / 登录密码 / https / ID / sosocam_id/cover）

```
/*
 *更新摄像机的名称
 *参数
 *camera_id:要被更新的摄像机id
 *camera_alias:摄像机昵称更新为alias
 */
```

函数：public static void update_camera_alias(String camera_id, String camera_alias);

```
/*
 *更新摄像机的登录名
 *参数
 *camera_id:要被更新的摄像机id
 *camera_user:摄像机登录名更新为user
 */
public static void update_camera_user(String camera_id, String camera_user);

/*
 *更新摄像机的登录密码
 *参数
 *camera_id:要被更新的摄像机id
 *camera_pwd:摄像机登录密码更新为pwd
 */
```

```
public static void update_camera_pwd(String camera_id, String camera_pwd) ;

/*
 *更新摄像机的安全传输值
 *参数
 *camera_id:要被更新的摄像机id
 *camera_https:摄像机安全传输值更新为https
 */
public static void update_camera_https(String camera_id, boolean camera_https);
/*
 *更新摄像机的sosocam_id
 *参数
 *camera_id:要被更新的摄像机id
 *obj_id:摄像机的sosocam_id更新为sosocam_id
 */
public static void update_camera_obj_id(String camera_id, String obj_id);
/*
 *更新摄像机的预览图
 *参数
 *camera_id:要被更新的摄像机id
 *cover:摄像机的预览图数据更新为cover
 */
public static void update_camera_cover(String camera_id, Bitmap cover) ;
```

#3.本地存储收藏摄像机管理 <获取收藏摄像机 / 和服务器比较 / 添加 / 删除 / 更新>

3.本地存储收藏摄像机管理

1）获取本地收藏摄像机

函数：public static ArrayList<COLLECTION_INFO> get_collections();//获取收藏的所有cams

```
/*
 *返回一个列表，列表的每一个元素是一个COLLECTION_INFO类型的对象，具体如下：
 *
 *String collection_id    分享id
 *boolean valid     当前camera的收藏是否有效，
 *                  true：有效可继续观看
 *                  false：当前收藏无效，即向你分享此camera的用户，已设置为不再分享
 *String alias      昵称
 *String user       登录名
 *String pwd        登录密码
 *String cover_url  预览图url
 */
```

示例：

```
Storage.get_collections();
```

2）比较

说明：把从服务器上获取到的收藏摄像机和当前账号本地收藏的cams做比较,如果一样返回True，如果不一样，此函数把服务器上的和本地的做同步，且返回No

函数：

```
public static boolean merge_collections(ArrayList<SERVICE_COLLECTION_INFO>
server_collections_list);
```

返回：返回BOOL值，True:服务器上收藏的cams和本地收藏的cams一样。Fakse:服务器上收藏的cams和本地收藏的cams不一样。

示例：

```
Storage.merge_collections(WebService.collections);
```

3）添加

说明：用此函数，添加收藏摄像机

函数：public static void add_collection(COLLECTION_INFO collection_info);

参数：collection_info //为COLLECTION_INFO对象

示例：

```
Storage.add_collection(collection);
```

4）删除

说明：从本地指定删除收藏的一个摄像机

函数：

```
public static void remove_collection(String collection_id);
```

参数：collection_id：即将被删除的收藏摄像机collection_id

示例：

```
Storage.add_collection(collection_id);
```

5）更新

说明：以下函数用来更新本地收藏的摄像机信息

函数：

```
/*
 *更新本地收藏的摄像机是否依旧合法，可观看视频
 *collection_id:要被更新的收藏摄像机分享id
 *valid:收藏摄像机的valid本地更新为valid
 */
public static void update_collection_valid(String collection_id, boolean valid);

/*
 *更新本地收藏摄像机的访问密码
 *collection_id:要被更新的收藏摄像机分享id
 *pwd:收藏摄像机的pwd本地更新为pwd
 */

public static void update_collection_pwd(String collection_id, String pwd);
```

示例：

```
Storage.update_collection_pwd(m_collection.collection_id, pwd);
```

#4.本地录像和图片 <获取文件 / 删除>

4.本地录像和图片

说明：使用以下函数，必须是使用了账号登录，未使用账号登录的也必须是离线账号，即local@localhost，即m_user_id不为空。

1）获取摄像机的拍照和录像文件路径

说明：此函数可以获取指定摄像机拍照和录像文件存放的路径

```
public static String get_album_folder(String camera_id);
```

参数：camera_id:摄像机的id

返回：返回camera_id的摄像机拍照和录像文件存放的路径

2）获取摄像机的拍照和录像文件

说明：此函数可以获取指定摄像机的拍照和录像的所有文件

函数：public static ArrayList<ALBUM_ITEM> get_local_records_list(String camera_id)

返回：返回一个list，列表的每一个元素是一个ALBUM_ITEM对象，详细如下：

```
Date t       //时间NSDate
```

获取函数：public Date getDate(); //获取当前图片或录像生成的时间参数

```
long size    // 文件大小
```

获取函数：public long getSize();//获取当前文件的大小

```
boolean image  //文件类型，两个值，true:图片 false:录像
获取函数：public boolean getImage();
//两个返回值，true：为图片文件，false：为录像文件

String path    //文件路径
获取函数：public String getPath();
设置函数：public void setPath(String path);

示例：
Storage.get_local_records_list(cam.id());
```

2）删除

```
说明：删除指定摄像机的文件
函数：
/*
 *删除指定摄像机的所有的录像和拍照文件
 *参数：camera_id:摄像机id
 */
public static void delete_local_records_list(String camera_id);
//删除该摄像机下所有图片和录像文件

/*
 *删除指定摄像机指定的录像或拍照文件
 *参数：camera_id:摄像机id
 */
public static void delete_local_records_list_file(String path);
//删除某一个文件（图片或录像文件）

示例：
Storage.delete_local_records_list("RTEST-001015-DUGMR");//删除所有文件
Storage.delete_local_records_list_file(path);//删除当前路径对应的一个文件
```

#5.本地存储报警图片<添加 / 获取>

5.本地报警图片操作

1）获取本地报警图片文件夹路径

```
说明：获取本地报警图片文件夹路径
函数：public static String get_alarm_folder(String camera_id, String alarm_id)
/*camera_id:摄像机id
 *alarm_id:当前摄像机报警id
 */
```

2）删除报警图片文件夹

```
说明：从本地删除报警文件夹
函数：public static void delete_alarm_folder()
```

3）写jpg文件到对应路径

```
说明：
函数：public static void write_jpg_file(byte[] data, String filepath)
/*data: 图片数据
*filepath:路径，写入图片数据的路径
*/
```