//WebService.java
//SoDemo
WebService类 用于和sosocam平台打交道
#1.用户管理 <注册 / 忘记密码 / 登录 / 激活 / 账号信息>

1.1用户管理
说明：当需要用到平台的用户才使用WebService类，在使用WebService前，必须先初始化（[WebService get_share];），不用了也必须反初始化（[WebService release_share];），初始化和反初始化是成对调用的

1.1.1注册新用户
说明：如果没有用户，先注册一个用户，注册用户是直接打开一个链接即可，其他由web_app那边去实现
注册新用户链接：url = "http://www.sosocam.com/mobile_register.html";;//注册新用户连接
示例：
```
private WebView mwebView;
mwebView.loadUrl(url);//打开链接
```

1.1.2忘记密码
说明：如果忘记密码了，直接打开一个连接即可，其他由网页那边去实现
忘记密码链接：url = "http://www.sosocam.com/mobile_forget.html";;//忘记密码连接
示例：
```
private WebView mwebView;
mwebView.loadUrl(url);//打开链接
```

1.1.3三种登录方式
1）账号登录
说明：普通账号登录
函数： public static String get_vetification_code_image_url();//获取验证码连接，用WebView打开
```
public static ERROR login_with_account(String username, String pwd, String vertification_code
, boolean exit_on_failed)
//账号用户登录
```

参数：username:账号名
pwd:账号密码
vertification_code:验证码，不需要验证码时是空
exit_on_failed：失败后是否退出 如果设置true，失败后就直接退出;如果设置为false，当登录失败因为是error:NETWORK_ERROR，会继续尝试登录.
示列：

```
/*获取验证码 若不需要的先直接跳过*/
private WebView verify_code;
verify_code.loadUrl(WebService.get_vetification_code_image_url());

case R.id.login_btn:
    String user = user_text.getText().toString();
    String pwd = pwd_text.getText().toString();
    String code = verify_code_text.getText().toString();
    if(user.equals("")){
        Toast.makeText(LoginActivity.this,
                getResources().getString(R.string.user_name_toast), Toast.LENGTH_SHORT).show
                ();
        break;
    }
    m_login_type = Storage.ACCOUNT_USER;
    m_login_id = user;
    m_login_token = pwd;
```

```
    WebService.login_with_account(user, pwd, code, true);
break;
```

2）第三方登录

说明：第三方登录，先要去第三方去申请，然后添加第三方平台，reecam采用的是ShareSdk

函数：public static ERROR login_with_authorization(LOGIN_TYPE type, String open_id, String access_token, boolean exit_on_failed)

//第三方登录

参数：open_id：从第三方拿到的open_id参数

access_token:从第三方平台拿到的access_token参数

exit_on_failed：失败后是否退出 如果设置true，失败后就直接退出，

如果设置为false,当登录失败是因为error：NETWORK_ERROR，会继续尝试登录

1.1.4登录状态监控和异常处理

说明：一般登录会先添加监听，来监控登录状态的改变，当监控到登录的状态是LOGIN_STATUS.IDLE时，才去获取错误值，根据不同错误值做不同处理（主要是帐户未激活 / 用户名或密码错误需要重试 / 账号被锁），这些是息息相关的，此处先分别介绍，最后一起举例。

1）登录接口

说明：使用用户管理，在使用前添加登录监听，能实时监控登录状态的改变，不再需要监控用户状态时必须删除监听

函数：

```
public static void add_login_listener(Login_Listener listener);//添加登录监听
public static void remove_login_listener(Login_Listener listener);//删除登录监听
public void on_login_status_changed();//登录监听
```

2）登录状态

说明：此函数一般在登录监听函数中使用，每次登录状态有改变时，就实时获取当前登录状态，做出相应处理。共有三种登录状态，如下：

```
public static enum LOGIN_STATUS {
        IDLE,    //登录失败
        LOGGING_IN, //登录中
        LOGGED_ON //登录成功
    };
```

账号登录状态获取：WebService.login_status; //账号登录状态

3）错误值

说明：此函数一般当监控到登录的状态是LOGIN_STATUS.IDLE时，才去获取错误值，根据不同的错误值做出相应处理。共有6种错误值（注释掉的错误，表示此处不用），如下：

```
public static enum ERROR {
        NO_ERROR,  //没有错误，表示登录成功
        //BAD_STATUS,
        NETWORK_ERROR, //网络错误
        SVR_ERROR,  //服务器错误
        BAD_AUTH, //认证错误（用户名或密码错误）
        NEED_VERTIFICATION, //验证码错误
        NO_ACTIVATED, //账号未激活
        //REGISTERED_BY_OTHERS,
        //INVALID_SHARE_ID,
        //SHARE_ID_EXISTS,
    };
```

账号登录错误：WebService.error; //账号登录错误

4）账号未激活

说明：当获取的错误值为ERROR.NO_ACTIVATED时，才能去做激活处理，获取激活账号类型和激活码和激活链接打开激活链接页面即可，后面的就交给web_app去处理

```
WebService.activate_user_type;//激活账号的类型
```

```
WebService.activate_auth_code;//账号激活码
String url = "http://www.sosocam.com/mobile_verify.html";//激活链接
```

5）用户名或密码或验证码错误

说明：当获取的错误值是ERROR.BAD_AUTH时，首先获取账号是否被锁（最多重试5次，不包含验证码错误），如果没有被锁，当剩下重试次数为4次时（即返回一次错误后开始开启验证码验证功能，这个功能可以根据自己的需求定，不一定是剩下四次也不一定非要用验证码），开启验证码登录验证

```
public static boolean locked_badauth = false;
WebService.locked_badauth;//判断账号是否被锁

public static int retry_times_badauth = 0;//账号登录剩下允许重试的次数
WebService.retry_times_badauth;
```

示例：

```
WebService.add_login_listener(this);; //添加登录监听
/*监控登录状态，根据不同状态 做出相应处理*/
    public void on_login_status_changed() {
        switch (WebService.login_status) {
        case LOGGING_IN:
            /*登录中，在此处做相应处理*/
            Log.e("SoDemo","logining");
        case LOGGED_ON:
            /*登录成功，在此处做相应处理*/
            Log.e("SoDemo","--Login---success--");
            break;
        case IDLE:
             /*登录失败，根据不同错误值做不同处理*/
             //WebService.error 获取错误值
            if (WebService.error == ERROR.BAD_AUTH) {
                /*错误值为认证错误（用户名或密码错误）*/
                if (WebService.locked_badauth) {
                    /*获取到账号已经被锁住了，在此处做相应处理*/
                    Log.e("SoDemo","account already lock!");
                } else {
                    /*账号还未锁定，获取验证码和剩余重试次数，在此处做相应处理*/
                    Log.e("SoDemo","you can try" + WebService.retry_times_badauth + "times");
                    //获取剩余重试数
                }
            } else if (WebService.error == ERROR.NEED_VERTIFICATION) {
                /*错误值为验证码错误 重新获取验证码*/
                Log.e("SoDemo","vertification error! ");
                WebView verify_code;
                verify_code.loadUrl(WebService.get_vetification_code_image_url());
            } else if (WebService.error == ERROR.NO_ACTIVATED) {
                /*错误值为未激活账号*/
                Log.e("SoDemo","account not activated! ");
                String url = "http://www.sosocam.com/mobile_verify.html";
                url +="&user_type=" + WebService.activate_user_type + "&authcode=" +
                WebService.activate_auth_code;
                WebView mweb_view;
                mweb_view.loadUrl(url);
            } else {
                /*其他错误*/
                Log.e("SoDemo","Network ERROR");
            }
            break;
```

```
        default:
            break;
        }
    }
WebService.remove_login_listener(this);;//移除监听
```

1.1.5 登录成功获取到相关信息

说明：只有登录成功了，WebService.login_status == LOGIN_STATUS.LOGGED_ON，才能获取到以下相关信息，否则获取到的是空值

1)获取当前账号用户id

获取：WebService.user_id;//获取当前用户user_id

示例：String current_user_id = WebService.user_id;

2)获取当前账号用户别名

获取：WebService.user_alias;//获取当前用户名称

示例：String current_user_alias = WebService.user_alias;

3)获取当前账号连接，用来管理此用户

函数：public static String get_current_user_info_url();//获取当前账号信息连接

示例：

```
case R.id.set_user_info_onclick:
    String url = WebService.get_current_user_info_url();
    if (url.equals("")) {
        Tools.showLongToast(this, getString(R.string.must_logon));
    } else {
        if (Tools.check_language().equals(""))
            url += "&lng=zh_cn";//中文显示，传递给web_app的语言参数
        else
            url +="&lng=en";//英文显示，传递给web_app的语言参数
        Log.e("SoDemo","-----set_user_info_onclick---------url----" + url);
        WebView mwebView;
        mwebView.loadUrl(url);
        }
    break;
```

4）获取当前摄像机

说明：获取当前账号的摄像机列表，返回一个列表，数组的每一个元素是一个SERVICE_CAMERA_INFO对象，一般用不和本地存储的摄像机做比较用，具体如下：

```
//  String obj_id; 摄像机sosocam_id
//  String alias; 摄像机昵称
//  String id;  摄像机id
//  boolean https; 摄像机传输是否加密
//  String user;  摄像机登录用户名
//  String pwd;  摄像机登录密码
```

获取从服务器获取账号里添加的摄像机：

```
public static ArrayList<SERVICE_CAMERA_INFO> svr_cameras = new ArrayList<SERVICE_CAMERA_INFO
>();
svr_cameras = WebService.cameras;//当前账号里添加的摄像机
```

示例：

```
svr_cameras = WebService.cameras;//从服务器上获取当前账号添加的cams
Storage.merge_cameras(svr_cameras);
//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false
/*一般如果返回false的话，都会先把IPCamMgr的所有摄像机清除，然后再从本地获取cams重新添加到IPCamMgr*/
IPCamMgr.delete_all_cameras();//清除IPCamMgr所有的cams
```

```
for(CAMERA_INFO cam : Storage.get_cameras()) { //获取本地cams,//逐一添加cam到IPCamMgr
    IPCam ipcam = IPCamMgr.add_camera(cam.getAlias(), cam.getId(), cam.getUser(), cam.getPwd
    (), cam.getHttps());
    ipcam.cover = cam.getCover();
    ipcam.model = cam.getModel();
    ipcam.id_4_sosocam = cam.getObj_id();
}
```

5）获取当前账号收藏摄像机

说明：从服务器上获取当前账号返回一个列表，数组的每一个元素是一个SERVICE_COLLECTION_INFO对象，具体如下：

```
//  String collection_id
//  String alias;
//  String cover_url;
```

获取当前账号收藏的摄像机：

```
public static ArrayList<SERVICE_COLLECTION_INFO> current_collections = new ArrayList<
SERVICE_COLLECTION_INFO>();
current_collections = WebService.collections;//当前账号收藏的摄像机
```

示例：

```
current_collections = WebService.collections;//从服务器上获取当前账号收藏的cams
Storage.merge_collections(WebService.collections);
//服务器上获取到的摄像机和当前账号本地的cams做比较,如果一样返回true,如果不一样，此函数把服务器的和本地的做同步，且返回false
```

1.1.6注销账号

说明：需要退出账号时，使用次函数，当然app的页面根据自己需求也要做相应处理

函数：public static void logout();//注销账号登录，即登出

示例：

```
WebService.logout();//退出当前账号
```

#2.服务器摄像机管理<添加 / 更新 / 删除 / >

2.1服务器上添加摄像机

说明：添加成功后，一般都要同步添加到Storage，再添加到IPCamMgr才能管理新添加的cam

回调函数：

```
public void on_add_camera_result(ERROR error, String obj_id);;//add_camera_listener

public static ERROR add_camera(String alias, String id, boolean https, int model,
add_camera_listener listener);//添加摄像机到服务器
```

示例：

```
private void add_camera_to_server() {
    if (WebService.ERROR.NO_ERROR != WebService.add_camera(name, m_ipcam.id(), m_ipcam.https
    (), m_ipcam.model, this)) {
        Log.e("SoDemo","failed_add_camera");
    } else {
        Log.e("SoDemo","add_camera_success");

    }
}

public void on_add_camera_result(WebService.ERROR error, String obj_id) {
    if (WebService.ERROR.NO_ERROR == error) {
        CAMERA_INFO add_cam = new CAMERA_INFO();
        if (obj_id == null) obj_id = "";
        add_cam.setObj_id(obj_id);
        add_cam.setAlias(m_camera_name);
        add_cam.setId(m_ipcam.id());
        add_cam.setUser(m_ipcam.user());
```

```
        add_cam.setPwd(m_ipcam.pwd());
        add_cam.setHttps(m_ipcam.https());
        add_cam.setModel(m_ipcam.model);
        Storage.add_camera(add_cam);
        Storage.save_cameras();
        IPCam ipcam = IPCamMgr.add_camera(m_camera_name, m_ipcam.id(), m_ipcam.user(),
        m_ipcam.pwd(), m_ipcam.https());
        if (ipcam != null) {
            ipcam.model = m_ipcam.model;
            ipcam.id_4_sosocam = obj_id;
        }
        Log.e("SoDemo","add_camera_ok");
    } else if (WebService.ERROR.REGISTERED_BY_OTHERS == error) {
        Log.e("SoDemo","the camera already add to other user");
    }else{
        Log.e("SoDemo","the camera add fail");
    }
}
```

2.2服务器上更新摄像机参数

说明：更新成功后，一般都要同步添加到Storage

函数：public static ERROR update_camera(String obj_id, String params, update_camera_listener listener);

回调函数：public void on_update_camera_result(ERROR error);

//更新摄像机参数到服务器上

示例：

```
if (WebService.ERROR.NO_ERROR != WebService.update_camera(Storage.get_camera_obj_id(
m_camera_id),
                        "alias=" + name, this)) {
    Log.e("SoDemo","update name to service fail!");
} else {
    Log.e("SoDemo","update name to service succeed!");
}
public void on_update_camera_result(WebService.ERROR error) {
    if (WebService.ERROR.NO_ERROR == error) {
        m_ipcam.set_alias(m_camera_name);
        Storage.update_camera_alias(m_camera_id, m_camera_name);
        Storage.save_cameras();
        Log.e("SoDemo","update_camera_name_ok");
    } else {
        Log.e("SoDemo","failed_update_camera_name");
    }
}
```

2.3服务器上删除摄像机

说明：删除成功后，一般都要去Storage删除此cam，再到IPCamMgr删除此cam

函数：public static ERROR delete_camera(String obj_id, delete_camera_listener listener)

回调函数：public void on_delete_camera_result(ERROR error);

接口：public interface delete_camera_listener

示列：

```
public void delete_camera_from_server(){
    if (WebService.ERROR.NO_ERROR != WebService.delete_camera(obj_id, DeleteCameraDialog.this
    )) {
         Log.e("SoDemo","remove camera from service fail!");
    }
}
public void on_delete_camera_result(WebService.ERROR error) {
```

```
    if (WebService.ERROR.NO_ERROR == error){
        Storage.remove_camera(m_camera_id);
        Storage.save_cameras();
        IPCamMgr.delete_camera(m_camera_id);
        Log.e("Sodemo","remove camera from service success!");
    }else{
        Log.e("Sodemo","remove camera from service fail!");
    }
}
```

#3.服务器报警信息管理<获取cams/cam报警信息>

3.1服务器上获取cams报警信息

1）执行获取动作

说明：获取cams报警信息，且添加监听，如果不是返回ERROR.NO_ERROR 说明获取失败

函数：public static void get_cameras_alarm_list(get_cameras_alarm_list_listener listener);

2）获取到结果后有回调函数

说明：回调函数返回两个参数，ERROR类型error值，error == ERROR.NO_ERROR表示获取成功；

另一个是一个list,每一个元素都是CAMERA_ALARM_INFO对象，具体如下：

```
public String obj_id = "";//摄像机sosocam_id
public String id = ""; //摄像机id
public int unread = 0; //未读报警数目
public int total = 0; //报警总数目
```

函数：public void on_get_cameras_alarm_list_result(ERROR error, ArrayList<CAMERA_ALARM_INFO> cameras_alarm_list);

示例：

```
WebService.get_cameras_alarm_list(CamAlarmListActivity.this);;//获取cams报警信息且添加监听
如果不是返回ERROR.NO_ERROR 说明获取失败

/*获取到结果后触发回调函数，当回调函数中返回error == ERROR.NO_ERROR表示获取成功*/
public void on_get_cameras_alarm_list_result(ERROR error, ArrayList<CAMERA_ALARM_INFO>
svr_alarm_list) {
    if (error == ERROR.NO_ERROR) {
        for (CAMERA_ALARM_INFO local_alarm : alarm_list) {
            for (CAMERA_ALARM_INFO svr_alarm : svr_alarm_list) {
                if (local_alarm.id.equals(svr_alarm.id)) {
                    local_alarm.obj_id = svr_alarm.obj_id;
                    local_alarm.total = svr_alarm.total;
                    local_alarm.unread = svr_alarm.unread;
                }
            }
        }
    }
}
```

3.2服务器上获取单个cam报警信息

1）函数：

```
public static void get_alarm_detail_list(String obj_id, get_alarm_detail_list_listener
listener);
```

回调函数：public void on_get_alarm_detail_list_result(ERROR error, ArrayList<ALARM_DETAIL> alarm_detail_list);

说明：回调函数返回两个参数，ERROR类型error值，error == ERROR.NO_ERROR表示获取成功；

另一个是一个list,每一个元素都是ALARM_DETAIL对象，具体如下：

```
public String aid;//摄像机报警id
public int alarm_type; //摄像机报警类型
public int image_numbers; //报警图片数量
```

```
public boolean read;//是否读过
public Date date;//报警时间
public String thumb_url;//预览的url地址
```

示例：

```
WebService.get_alarm_detail_list(activity.m_obj_id, activity);
```

2）获取指定报警图像的信息列表

函数：public static ERROR get_alarm_image_info_list(String aid,
get_alarm_image_info_list_listener listener)

回调函数：public void on_get_alarm_image_info_list_result(ERROR error, ArrayList<
ALARM_IMAGE_INFO> alarm_image_info_list);

说明：回调函数返回两个参数，ERROR类型error值，error == ERROR.NO_ERROR表示获取成功；
另一个是list，每一个元素都是ALARM_IMAGE_INFO对象，具体如下：

```
public Date date;//报警时间
public String url;//每张图片所代表的url地址
```

示例：

```
if (ERROR.NO_ERROR == WebService.get_alarm_image_info_list(m_aid, this))
    Log.e("SoDemo","download_alarm_image_list_now");
else
    Log.e("SoDemo","download_alarm_image_list_failed");

public void on_get_alarm_image_info_list_result(ERROR error, ArrayList<ALARM_IMAGE_INFO>
alarm_image_info_list) {
    if (ERROR.NO_ERROR == error) {
        for (ALARM_IMAGE_INFO info : alarm_image_info_list) {
            Log.e("SoDemo","处理每张图片")
            }
        Log.e("SoDemo","download_alarm_image_list_succeed")
    } else {
        Log.e("SoDemo","download_alarm_image_list_failed");
        }
    }
```

取消获取指定报警图像信息的异步任务

说明：取消当前正在执行的获取指定报警图像信息的异步任务，主要在于释放资源，当离开显示报警图片界面或者退出apk时可调用

函数：public static void cancel_get_alarm_image_info_list_tasks();

3）获取一张报警的图片

函数：public static void get_alarm_image(String image_id, String url,
get_alarm_image_listener listener)

回调函数：public void on_get_alarm_image_result(String image_id, byte [] data);

```
//image_id:图像ID
//data:图像数据
```

示例：

```
WebService.get_alarm_image(pic.cache_filename, info.url, this);

public void on_get_alarm_image_result(String image_id, byte [] data) {
    if (data != null) {
        Bitmap bitmap = data;
        Storage.write_jpg_file(data, Storage.get_alarm_folder(m_id, m_aid) + "/" + filename);
        }
}
```

取消获取指定报警图像信息的异步任务

说明：取消当前正在执行的获取指定报警图片的异步任务，主要在于释放资源，当离开显示报警图片界面或者退出apk时可调用

```
public static void cancel_get_alarm_image_tasks()
```

#4.分享和收藏<删除 / 添加>

4.1分享开启和关闭

1）开启分享：

函数：public static ERROR share_camera(String obj_id, share_camera_listener listenter)

回调函数：

```
public void on_share_camera_result(ERROR result, String share_id, String share_url, String share_qrcode);
```

回调函数结果具体如下：

```
result;//sosocam平台返回的分享结果ERROR值
share_id;//分享码的id
share_url;//分享码的url地址
share_qrcode;//分享码的二维码地址
```

2）关闭分享

函数：public static ERROR disshare_camera(String obj_id, disshare_camera_listener listenter)

回调函数：public void on_disshare_camera_result(ERROR result);

4.2收藏增加和删除

1）获取当前账号下收藏的camera：

函数：public static void get_collection_camera(String share_id, get_collection_camera_listener listenter)

回调函数：public void on_get_collection_camera_result(ERROR result, String id, boolean ssl);

示例：

```
WebService.get_collection_camera(m_collection.collection_id, this);
public void on_get_collection_camera_result(ERROR result, String id, boolean ssl) {
    if (result == ERROR.INVALID_SHARE_ID){
        Log.e("SoDemo","invalid_collection");
        return;
    } else if(result == ERROR.NO_ERROR){
        IPCam ipcam = IPCamMgr.get_disposable_camera();
        ipcam.add_listener(this);
    }else{
        Log.e("SoDemo","failed_to_get_collection_camera");
        return;
    }
}
```

2）获取收藏

函数：public static void get_collection(String share_id, get_collection_listener listenter)

回调函数：public void on_get_collection_result(ERROR result, String alias, String cover_url);

示例：

```
WebService.get_collection(m_Share_id, this);
public void on_get_collection_result(ERROR result, String alias,String cover_url) {
    if(result == ERROR.NO_ERROR){
        COLLECTION_INFO collection = new COLLECTION_INFO();
        collection.collection_id = m_Share_id;
        collection.alias = alias;
        collection.cover_url = cover_url;
        Storage.add_collection(collection);
        Log.e("Sodemo","add_collection_camera_success");
    } else if(result == ERROR.INVALID_SHARE_ID){
        Log.e("Sodemo","invalid_collection");
    } else{
        Log.e("Sodemo","add_collection_camera_failed");
    }
}
```

3）添加收藏

函数：public static ERROR add_collection(String share_id, add_collection_listener listenter)

回调函数：public void on_add_collection_result(ERROR result, String alias, String cover_url);

示例：

```
WebService.add_collection(m_Share_id, this);
public void on_add_collection_result(ERROR result, String alias,String cover_url) {
    if(result == ERROR.NO_ERROR){
        COLLECTION_INFO collection = new COLLECTION_INFO();
        collection.collection_id = m_Share_id;
        collection.alias = alias;
        collection.cover_url = cover_url;
        Storage.add_collection(collection);
        Log.e("Sodemo","add_collection_camera_success");
    } else if(result == ERROR.INVALID_SHARE_ID){
        Log.e("Sodemo","invalid_collection");
    } else if(result == ERROR.SHARE_ID_EXISTS){
        Log.e("Sodemo","collection_exists");
    } else{
        Log.e("Sodemo","add_collection_camera_failed");
    }
}
```

4）删除收藏

函数：public static ERROR remove_collection(String share_id, remove_collection_listener listenter)

回调函数：public void on_remove_collection_result(ERROR result, String share_id);

示例：

```
if (ERROR.NO_ERROR == WebService.remove_collection(m_Share_id, CollectionDeleteDialog.this)) {
    Log.e("SoDemo","succeed_delete_collection");
} else {
    Log.e("SoDemo","failed_delete_collection");
}
public void on_remove_collection_result(ERROR result, String share_id) {
    if(result == ERROR.NO_ERROR){
        Storage.remove_collection(m_Share_id);
        Log.e("SoDemo","delete_collection_ok");
    } else {
        Log.e("SoDemo","failed_delete_collection");
    }
}
```

#5.新版本更新（app和fw）

5.1服务器上检查固件是否有更新：//内部函数，用户无需使用

函数：public static void check_latest_camera_fw(String current_version, check_latest_version_listener listener)

5.2服务器检查apk是否有可更新的版本;//内部函数，用户无需使用

函数：public static void check_latest_apk(String current_version, check_latest_version_listener listener)