Sheely.Wang
2014年4月18日

移动平台开发框架(v 1.0)

我们的开发框架目前还在整理，略有不合适之处请各位包涵，有任何新想法，好的建议请与我联系。

1.http网络数据收发

1.1 http客户端发送

普通api调用（post方式）

SHPostTaskM 类

```
SHPostTaskM * task = [[SHPostTaskM alloc]init];
task.URL = URL_FOR(@“login”);//http://api.local.com/login
task.delegate = self;
[task start];
```

1.2 http客户端接收

网络数据返回

code =0 的情况下回调

```
- (void)taskDidFinished:(SHTask *)task;
```

[task result]对象为数据集合，api下一定为json中的data字段集合;

举例:

```
NSDictionary * result = (NSDictionary*)[task result];
Entironment.instance.sessionid = [result objectForKey:@“session_id"];
```

code !=0的情况下回调

```
- (void)taskDidFailed:(SHTask *)task;
```

一般会进行错误处理，比如提示错误

```
[task.respinfo show];
[self dismissWaitDialog];
```

task状态的使用task.respinfo

task.respinfo.code标示api返回状态

task.respinfo.message标示api返回状态字符串

举例:(密码错误,OK)

有参情况下SHPostTask 的使用

```
[task.postArgs setValue:self.custId forKey:@“id”];
```

缓存处理

```
typedef enum
{
    CacheTypeNone,
    CacheTypeKey,
    CacheTypeTimes//时间缓存
}CacheType;
task.cachetype = CacheTypeKey;
```

1.3 http网络数据格式

名词解释:

api:应用程序编程接口，在这里指客户端访问服务端获取数据的接口，我们弱化俗称api

举例：http://local.com:7003/beast_pos/login

1.2.1 传输协议

一般我们现在采用http post方式收发数据

举例

http://local.com:7003/beast_pos/login

1.2.2 传输数据格式

不管是服务端，还是客户端规范的数据采用json在网络上传输

3.接受api返回数据的格式

接受数据格固定分为3块

{code:[int] message:[string] data:[string] }的json ，其中[]内为数据类型

code:表示 api返回的状态

我们定义code=0 表示api反回了“预期”的结果

code>0表示api没有报错，但是得到的结果非客户端预期（比如密码错误）

code<0表示api内部出错了。

需要注意的时<0也表示一些固化的值，比如#define CODE_RELOGIN -5

表示session过期，需要重新登陆。

-1000<0这段值为固化值

下面是举例：

登陆接口:

http://test.thebeastshop.com:7003/beast_pos/login

返回

{"code":0,"message":"OK","data":
{"session_id":"2c784070-4a3f-48a7-91f4-61dba9167966", name:sheely}}

1.2.3 列表类嵌套

http://localhost:7003/beast_pos/get_products

{"code":0,

"message":"OK",

"data":{

"products":[

{"category":79,"price":650,"image_url":"http://
statics.thebeastshop.com/media/catalog/product/
2/2/222222222222.jpg","name":"故事订花（鲜
花）","sku":"10010001",customize_price: true},

{"category":79,"price":650,"image_url":"http://
statics.thebeastshop.com/media/catalog/product/
2/2/222222222222.jpg","name":"故事订花（鲜
花）","sku":"10010001",customize_price: false}

],

"categories":[{"name":"干花","category_ids":[88,87,86]},{"name":"服
饰","category_ids":[91,90,89]}]

}

1.2.4 发送数据格式

发送的数据和接受的数据类似会采用固化的格式

通常情况，我们会采用post来发送数据，数据采用json格式，数据会放入postbody中，保持url干净,格式如下

{identication: [string] data:[string]}

identication:标识客户端的一些固有信息，比如用户名，密码

identication:标识里有个“type”字段表示用户信息验证类型。值”basic”表示用户密码认证，值:session标识session认证

identication标识每个api都会有且相同(如果服务器支持session的话，登陆完成后，identication标识里就不在有用户名和密码，转而由session_id替代)，

identication标识里还有客户端版本，客户端类型，等一些硬性标识

data:表示可以理解层客户端传递给服务端对应api的参数

下面是举例

登陆

http://test.thebeastshop.com:7003/beast_pos/login

```
{
    identication: {
        type: "basic",
        username: "admin",
        password: “MD3TWWRRT"
    },
    data: {
                shop: 4
    }
}
```

获取产品

http://localhost:7003/beast_pos/get_products

```
{
    identication: {
        type: "session",
        session_id: "9e504d7e-c9b9-4ea9-9acb-2f49e7339254"
    },
    data: {
    }
}
```

下面枚举了identication标识里出现的类型

type	区分basic和session两种模式
username	用户名
password	用户密码
session_id	session的id
imei	手机标识
info	操作系统id
version	客户端版本

2 TCP 网络数据收发

2.1 tcp 网络数据包格式

header	data	v

header拥有 16个字节

标记	version	count	backup

头四个字节：0xffffffff

第5到第8字节 int (低位在前，高位在后)消息版本

第9到第12字节 int (低位在前，高位在后)消息版本 表示数据包长度(为data的字节数组长度)

剩下字节是保留

2.2 结构定义

客户端完成tcp通讯主要依靠如下几个类

SHMsg	消息基类
SHMsgM	支持版本0x1定义的发送消息。
SHResMsgM	支持版本0x1定义的接收消息，以及推送消息
SHMsgManager	消息处理类,一般情况下使用者使用不到该类

2.2 类属性含义

@interface SHMsg : NSObject

@property (strong,nonatomic,readonly) NSString * guid ;消息标示

- (NSData*) data;（消息的发送必须转换成nsdata类型）

@end

@interface SHMsgM : SHMsg

@property (strong,nonatomic) NSString * target;（发给服务器，用于辨识消息调用的接口）

@property (strong,nonatomic) NSMutableDictionary* args;(接口所附带的参数)

@end

@interface SHResMsgM : SHMsg

@property (strong,nonatomic) NSObject * result;(结果，这里支持0x1版本的为json)

@property (strong,nonatomic) Respinfo * respinfo;(结果状况)

@property (strong,nonatomic) NSString * response;(对应的接口，客户端需要依靠这个接口名来对应该返回接口的用途)

@end

2.4 tcp 数据包客户端发送

讨论version为 0x1时消息体处理模型

当header中version为1时，表示消息体data为jason数据格式并约定如下

{id：target：args：}

id:每个消息时唯一的一般为guid

target:消息发送目标

args：消息参数

举例

```
{
        id:”uuii-9988-jksw-27ui-ioo8-8i8i1-uiwn-89ik”
        target:”login”
        args:
        {
                login:”sheely”
                password:”WE$@QSA45”
        }
}
```

实例:

```
SHMsgM * msg = [[SHMsgM alloc]init];
    msg.target  = @"login";
    [msg.args setValue:@"sheely" forKey:@"login"];
    [msg.args setValue:@"WE$@QSA45" forKey:@"password"];
    [msg start:^(SHResMsgM *) {

    } taskWillTry:^(SHResMsgM *) {

    } taskDidFailed:^(SHResMsgM *) {

    }];
```

2.3 tcp 心跳包处理

和普通包没有很大区别，主要再taget:heart 参数为空，心跳包每10秒发送一个。

2.5 tcp 普通数据包客户端接收

客户端接收的数据包需要处理粘包

格式和发送包雷同

header格式一样

我们普通协议主要考虑version版本为0x1时的情况

data依然为json

{id,code,message,response,data}

guid 对应发送消息的guid，主要处理一对一消息

code message,data与http模式下的含义相同

response,表示响应通知,一般和发送消息的target相同

举例:

```
{
        id:”uuii-9988-jkuw-26ui-ioo8-8i8i1-uiwn-89ik”
```

code:0

message:“成功”

response:“login”

data{

session:duii-wew3-jsuw-26ui-8kid-usxm-uiwn-82oiu

}

}

实例

```
SHMsgM * msg = [[SHMsgM alloc]init];
  msg.target  = @"login";
  [msg.args setValue:@"sheely" forKey:@"login"];
  [msg.args setValue:@"WE$@QSA45" forKey:@"password"];
  [msg start:^(SHResMsgM *) {

  } taskWillTry:^(SHResMsgM *) {

  } taskDidFailed:^(SHResMsgM *) {
        [rmsg.respinfo show];
  }];
```

2.6 tcp 推送包客户端接收

和普通接收消息雷同

区别时guid值为“”或没有key,

但是此时response必须不为空

3 对话框

一般常用对话框模式，在shviewcontroller基类里均包涵了（Android 在BaseActivity）

- (void)showAlertDialog:(NSString*)content;

- (void)showAlertDialogForCancel:(NSString*)content;

- (void)showAlertDialog:(NSString*)content otherButton:(NSString*)button;

- (void)showAlertDialog:(NSString*)content button:(NSString*)button otherButton:(NSString*)otherbutton;

-(void)showAlertDialog:(NSString*)content button:(NSString*)button otherButton:(NSString*)otherbutton tag:(int)tag;

对话框点击回调

- (void)alertViewCancelOnClick;

-(void)alertViewEnSureOnClick;

一般等待框同样在shviewconroller基类中

- (void)showWaitDialog:(NSString*)title state:(NSString*)state;

- (void)showWaitDialogForNetWork;

-(void)showWaitDialogForNetWorkDismissBySelf;

关闭则使用

- (void)dismissWaitDialog;

- (void)dismissWaitDialog:(NSString*)msg;

- (void)dismissWaitDialogSuccess;

4 页面流转

内聚页面跳转和传统方式类似

耦合页面采用intent方式跳转

iOS

```
SHIntent * intent = [[SHIntent alloc]init];
intent.target = @“SHTitleContentViewController";
[intent.args setValue:[dic valueForKey:@"newssubject"]forKey:@"utitle"];
[intent.args setValue:[dic valueForKey:@"newscontent"]forKey:@"content"];
[intent.args setValue:@"详情"forKey:@"title"];
intent.container = self.navigationController;
[[UIApplication sharedApplication]open:intent];
```

Android

Android所有页面全部采用fragment编写

```
intent.putExtra("class",
SHModuleManager.getInstance().getModuleName("interview"));
startActivity(intent);
SHApplication.getInstance().open(SHModuleManager.getInstance().getModuleN
ame(“interview"));
```

5 标准控件

iOS

iOS在Common文件夹下，具体查看Common.h

6 样式

iOS

参见

style.xml 样式表