1. **概述**

数据（包含文本和二进制数据）使用JSON格式上传，二进制数据用Base64编码后再上传。

1. **前端API**
   1. 方法

PushSender.sendMessage(String action, Map<String, Object> map[, int connection\_timeout[, int read\_timeout]])

* 1. 参数

|  |  |
| --- | --- |
| **参数** | **说明** |
| action | 上传到的功能模块 |
| map | 要上传的数据的键值对 |
| connection\_timeout | 可选，连接服务器超时时间，单位为毫秒，默认值为8000 |
| read\_timeout | 可选，从服务器读取数据超时时间，单位为毫秒，默认值为5000 |

* 1. 返回值

|  |  |
| --- | --- |
| **值** | **说明** |
| “network error” | 主机未连网 |
| “error” | 连接服务器失败或发生IO错误 |
| “param is null” | map参数为空 |
| 其他 | 服务器返回的数据 |

* 1. 说明

1. Map参数键/值对中的值若为Bitmap类型或File类型，则识别为要上传的文件，其中Bitmap类型将转换为png格式的图片
2. 由于服务器端接收到的文件首先存储在内存中，为防止服务器负载过大，建议每次上传的数据不超过4M
3. action参数的取值和传送的数据格式参见python后台相应文档
4. **后台API**

在Handler对象中的post方法中访问self.request.body属性获得上传的JSON字符串

1. **示例**

* 前端代码

Map<String, Object> param = new HashMap<String, Object>();

param.put("format", "rmvb");

param.put("userid", "1234");

param.put("file", new File("123.rmvb"));

PushSender.sendMessage("upload", param);

为防止阻塞UI线程，可以借助AsyncTask将上传操作放到单独的线程中：

Upload upload = new Upload();

upload.excute();

private class Upload extends AsyncTask<Void, Void, String> {

@override

protected String doIn Background(Void... params) {

Map<String, Object> param = new HashMap<String, Object>();

param.put("format", "rmvb");

param.put("userid", "1234");

param.put("file", new File("123.rmvb"));

PushSender.sendMessage("upload", param);

}

@override

protected void onPreExecute() {

}

@override

protected void onPostExecute(String result) {

if (result.equals("network error")) {

//设备未连网

} else if (result.equals("error") {

//连接服务器失败或发生IO错误

} else {

//处理服务器返回的数据

}

}

}

* 后台代码

import tornado.httpserver

import tornado.ioloop

import tornado.web

import json

import base64

class Handler(tornado.web.RequestHandler):

def post(self):

content = self.request.body

j = json.loads(content)

filename = j['userid'] + '.' + j['format']

f = open("/data/uploads/%s" % filename, "wb")

f.write(base64.decodestring(j['file']))

f.close()

if \_\_name\_\_ == "\_\_main\_\_":

app = tornado.web.Application([(r"/api/upload", Handler)])

http\_server = tornado.httpserver.HTTPServer(app)

http\_server.listen(8080)

tornado.ioloop.IOLoop.instance().start()