现代操作系统应用开发实验报告

姓名：谷雨

学号：16341005

实验名称：lab6 实验报告

一、参考资料

1. **RapidJSON** 文档

   http://rapidjson.org/zh-cn/index.html

2. **Rapidjson** 的简单使用示例

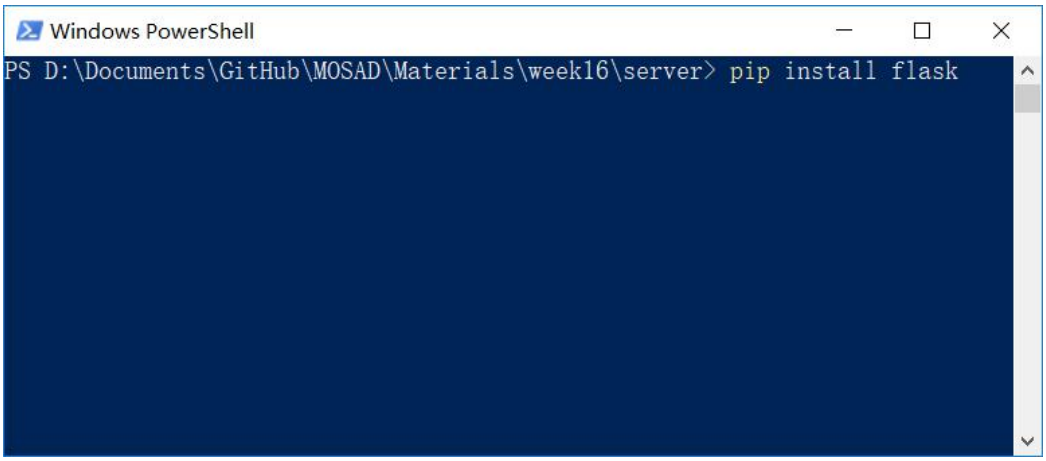
   https://blog.csdn.net/chary8088/article/details/72875072

3. **C++**项目 **RapidJson** 的详细用法总结

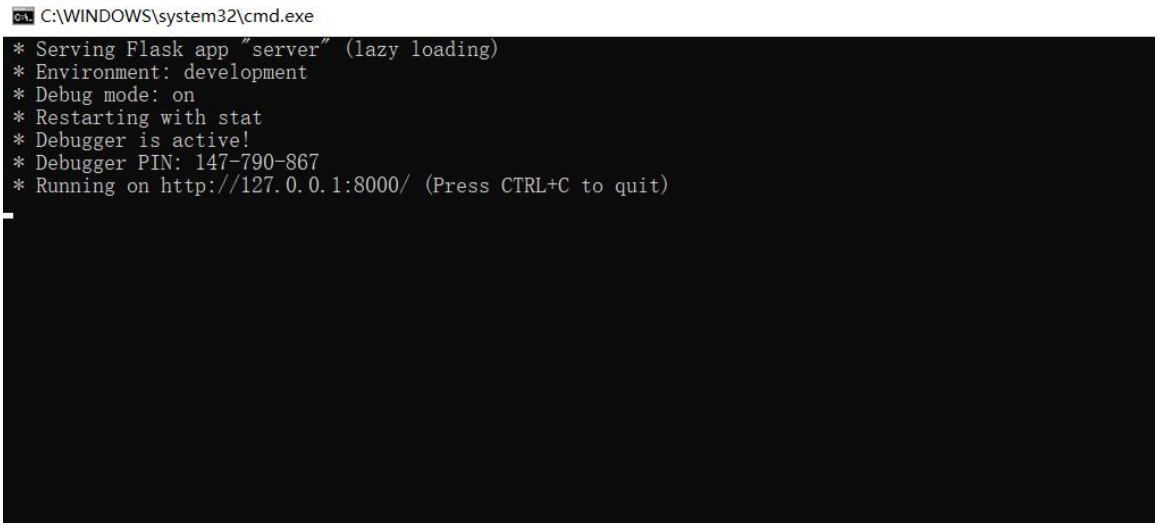
   https://blog.csdn.net/u014449046/article/details/79070301

二、实验步骤

安装 flask



启动 server 端



Client 代码

1. 登录：

```
/*
POST /auth
请求：
{
    "username": "dtc",
    "password": "123"
}
返回：
若成功，status 为 true；若失败，status 为 false
已登录的再次登录，status 为 true
*/
```

使用 Post 方式，将账号密码作为 RequenstData 部分。

数据部分先使用 Rapidjson 构建 json 实例，再转为字符串。

```
void LoginRegisterScene::loginButtonCallback(cocos2d::Ref * pSender) {
 // Your code here
   HttpRequest* request = new HttpRequest();
   request->setRequestType(HttpRequest::Type::POST);
   rapidjson::Document document;
   document.SetObject();
   rapidjson::Document::AllocatorType&  allocator = document.GetAllocator();
   auto name = rapidjson::StringRef(usernameInput->getString().c_str());
   auto pass = rapidjson::StringRef(passwordInput->getString().c_str());
   document.AddMember("username", name, allocator);
   document.AddMember("password", pass, allocator);
   rapidjson::StringBuffer buffer;
   rapidjson::Writer<rapidjson::StringBuffer> writer(buffer);
   document.Accept(writer);
   string postdata = buffer.GetString();
   request->setRequestData(postdata.c_str(),postdata.size());
   request->setUrl("http://127.0.0.1:8000/auth");
   request->setResponseCallback(CC_CALLBACK_2(LoginRegisterScene::onLoginResposeComplete, this));
   HttpClient::getInstance()->enableCookies("ck");
   cocos2d::network::HttpClient::getInstance()->send(request);
   request->release();

}
```

返回内容的解析

```
void LoginRegisterScene::onLoginResposeComplete(HttpClient *sender, HttpResponse* response) {
   auto buffer = response->getResponseData();
   rapidjson::Document doc;
   doc.Parse(buffer->data(), buffer->size());
   if (doc["status"] == true) {
       this->messageBox->setString(std::string("Login OK\n") + doc["msg"].GetString());
   }
   else {
       this->messageBox->setString(std::string("Login Failed\n") + doc["msg"].GetString());
   }
```

2. 注册

```
/*POST /users
请求：
{
    "username": "dtc",
    "password": "123"
}
返回：
若成功，status 为 true；若失败，status 为 false
有相同用户名的情况，status 是 false
*/
```

与登录基本相同

```
void LoginRegisterScene::registerButtonCallback(Ref * pSender) {
 // Your code here
   HttpRequest* request = new HttpRequest();
   request->setRequestType(HttpRequest::Type::POST);
   rapidjson::Document document;
   document.SetObject();
   rapidjson::Document::AllocatorType&  allocator = document.GetAllocator();
   auto name = rapidjson::StringRef(usernameInput->getString().c_str());
   auto pass = rapidjson::StringRef(passwordInput->getString().c_str());
   document.AddMember("username", name, allocator);
   document.AddMember("password", pass, allocator);
   rapidjson::StringBuffer buffer;
   rapidjson::Writer<rapidjson::StringBuffer> writer(buffer);
   document.Accept(writer);
   string postdata = buffer.GetString();
   log("%s", postdata.c_str());
   request->setRequestData(postdata.c_str(), postdata.size());
   request->setUrl("http://127.0.0.1:8000/users");
   request->setResponseCallback(CC_CALLBACK_2(LoginRegisterScene::onRegisterResposeComplete, this));
   cocos2d::network::HttpClient::getInstance()->send(request);
   request->release();
```

3. 获取用户信息

```
/*
GET /users?limit=$num
limit
返回：
若成功，status 为 true；若失败，status 为 false
data 中是一个对象(object)的数组(array)，每个对象表示一个用户；对象中有一个 username 字段表示用户名，和一个 deck 字段表示卡组；卡组也是一个对象的数组，每个对象表示一个卡组，卡组中的键值对的意思是“键是字符串，代表卡的名字；值是整数，代表卡的数量”
*/
```

使用 GET 方法

```
// Your code here
// Your code here
HttpRequest* request = new HttpRequest();
request->setRequestType(HttpRequest::Type::GET);

log("%s", limitInput->getString().c_str());
request->setUrl("http://127.0.0.1:8000/users?limit="+limitInput->getString());
request->setResponseCallback(CC_CALLBACK_2(UsersInfoScene::onGetUserResposeComplete, this));
cocos2d::network::HttpClient::getInstance()->send(request);
request->release();
```

返回内容解析

```
UsersInfoScene::onGetUserResposeComplete(HttpClient *sender, HttpResponse* response) {
uto buffer = response->getResponseData();
apidjson::Document doc;
oc.Parse(buffer->data(), buffer->size());
 (doc["status"] == true) {
  rapidjson::Value &dataArray = doc["data"];
  string result="";
  if (dataArray.IsArray())
  {
      for (rapidjson::SizeType i = 0; i < dataArray.Size(); i++)
      {
          const rapidjson::Value& object = dataArray[i];
          result+="\nusername : ";
          result += object["username"].GetString();
          result += "\ndeck : ";
          auto &deckArray = object["deck"];
          for (rapidjson::SizeType j = 0; j < deckArray.Size(); j++) {
              for (auto &mem : deckArray[j].GetObjectW()) {
                  result += "\n";
                      result += mem.name.GetString();
                  result += " : ";
                  result += std::to_string(mem.value.GetInt());
              }
          }
      }
```

4. 修改用户信息

```
/*
PUT /users
请求：
```

```
{
    "deck": [
        {
            "Black Magician": 2,
            "Black Magician Girl": 2,
            "Star Dust Dragon": 1
        },
        {
            "A": 1,
            "B": 23,
            "C": 345
        }
    ]
}
```

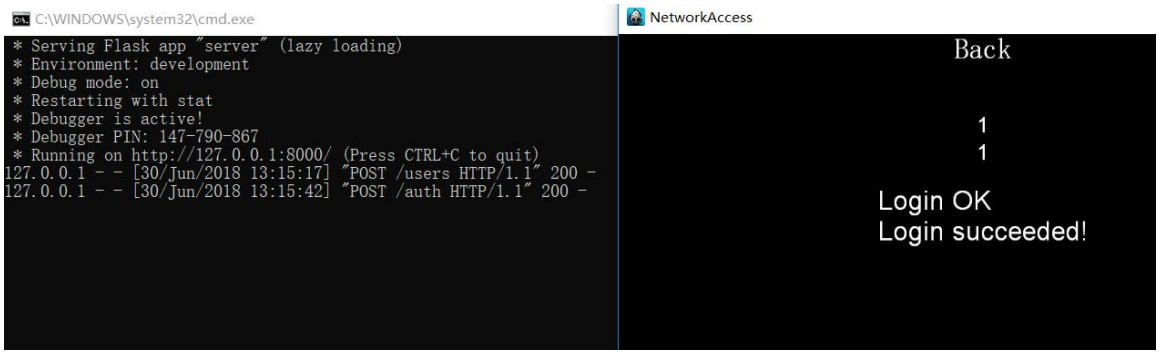
```
返回：
若成功，status 为 true；若失败，status 为 false
有相同用户名的情况，status 是 false
*/
```

```cpp
void ModifyUserScene::putDeckButtonCallback(Ref * pSender) {
 // Your code here
    HttpRequest* request = new HttpRequest();
    request->setRequestType(HttpRequest::Type::PUT);
    std::string putdata = "{\"deck\":" + deckInput->getString() + "}";

    request->setRequestData(putdata.c_str(), putdata.size());

    request->setUrl("http://127.0.0.1:8000/users");
    request->setResponseCallback(CC_CALLBACK_2(ModifyUserScene::onpPutDeckResposeComplete, this));
    cocos2d::network::HttpClient::getInstance()->send(request);
    request->release();


}
void ModifyUserScene::onpPutDeckResposeComplete(HttpClient *sender, HttpResponse* response) {
    auto buffer = response->getResponseData();
    rapidjson::Document doc;
    doc.Parse(buffer->data(), buffer->size());
    if (doc["status"] == true) {
        this->messageBox->setString(std::string("Update OK\n") + doc["msg"].GetString());
    }
    else {
        this->messageBox->setString(std::string("Update Failed\n") + doc["msg"].GetString());
    }
```

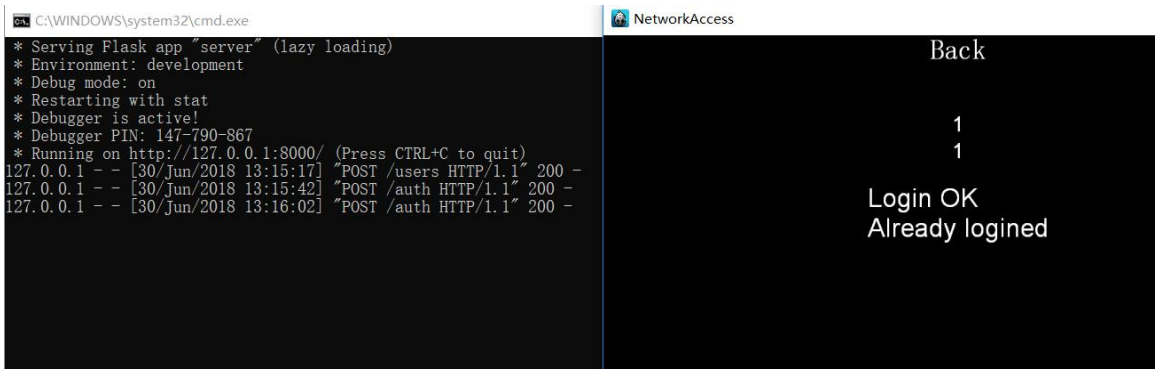
三、关键步骤截图

用户注册

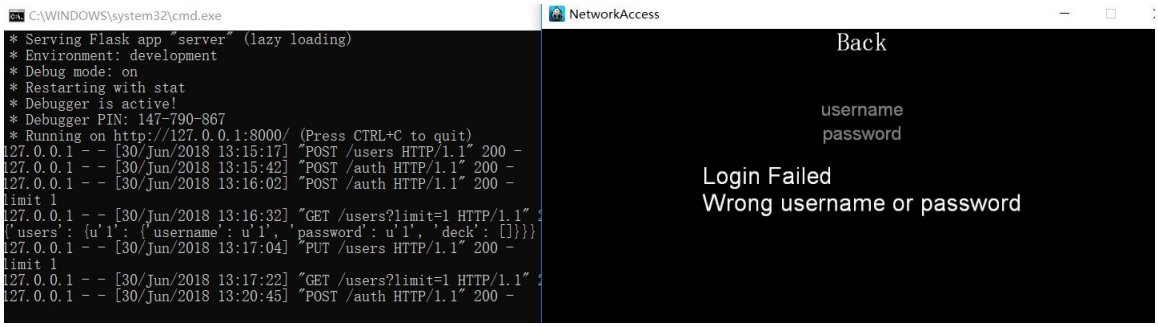
用户登录



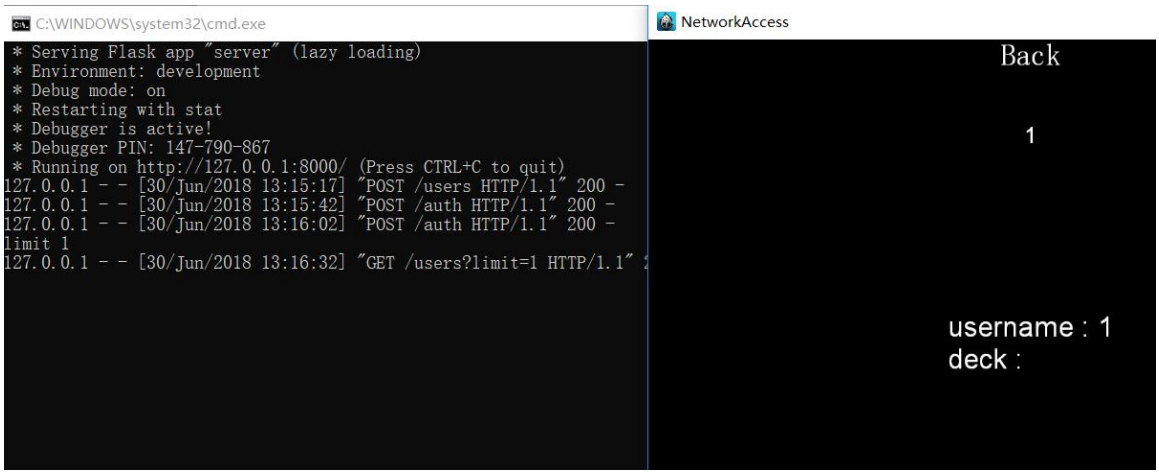
重复登录



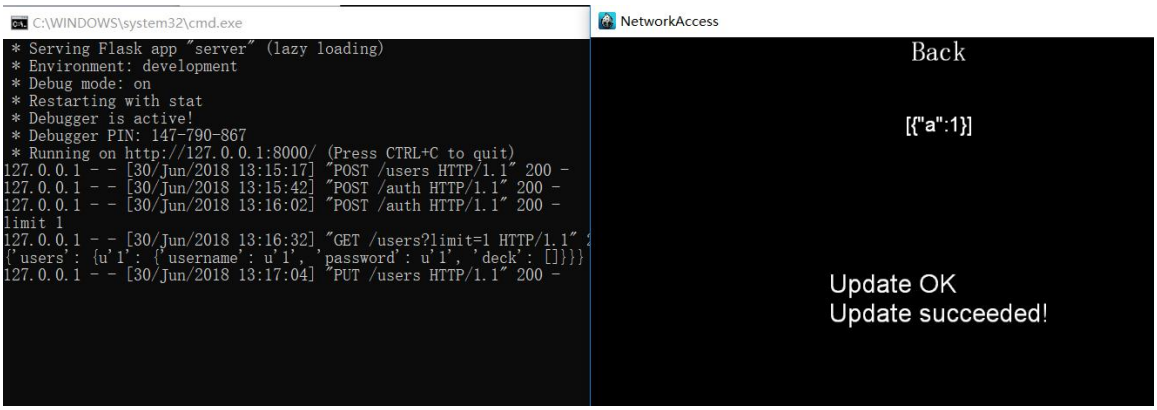
错误登录



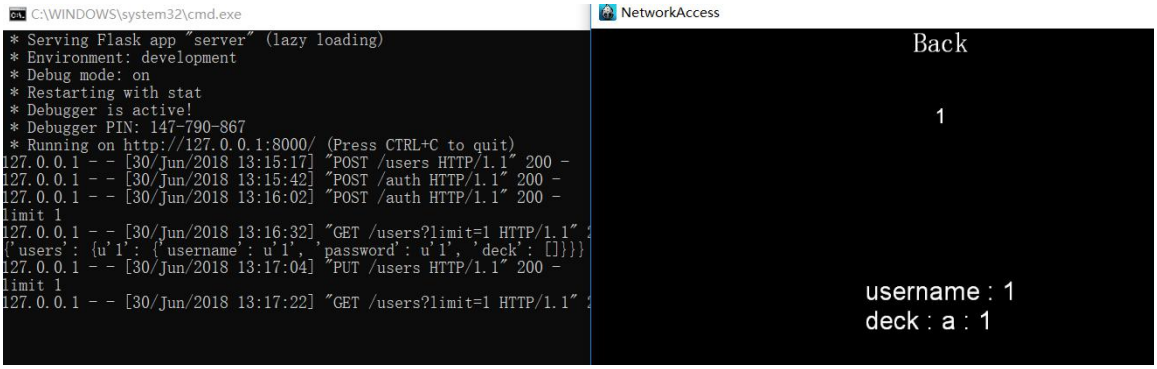
得到用户信息



修改卡组



再次查看信息

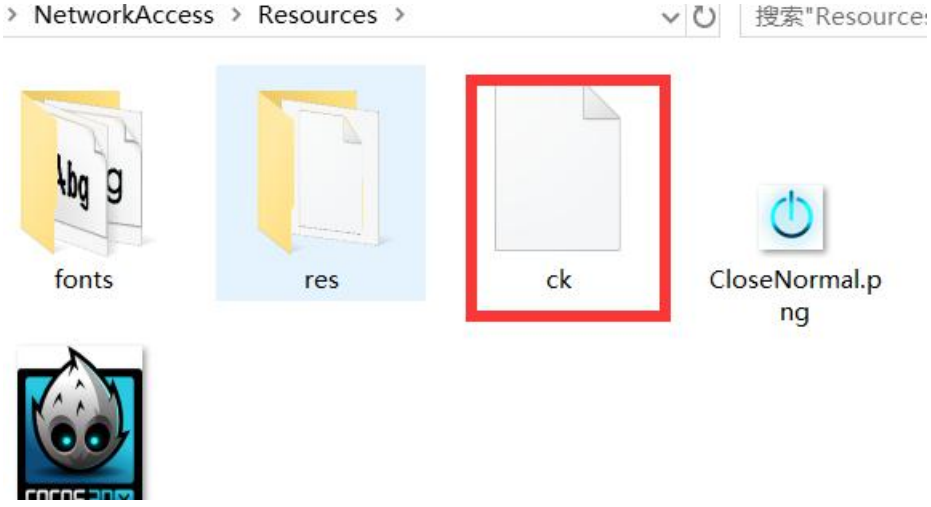


四、亮点与改进（可选）

实现了 COOKIE 记录保持 session

```
request->setUrl( http://127.0.0.1:8000/auth );
request->setResponseCallback(CC_CALLBACK_2(LoginRegisterScene::
HttpClient::getInstance()->enableCookies("ck");
cocos2d::network::HttpClient::getInstance()->send(request);
request->release();
```

文件位置



内容

ck

```
# Netscape HTTP Cookie File
# https://curl.haxx.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

#HttpOnly_127.0.0.1 FALSE   /   FALSE   0   session eyJ1c2VybmFtZSI6IjEifQ.Dhijfg.FptBkHY22kAhlL4ot7UVbHtM0X4

```

原理

HTTP 是一种无状态的协议。

同一个 Client 对服务端的两次访问，对服务端来说是无法确定它们是否为同一用户的访问。

所以引入 Session 机制。在 Session 保持期间，服务端是应该能知道同一用户的访问的。

具体实现一般在 Client 端借助 Cookie 机制，记录一段唯一识别的 ID；在服务端将 session 的记录保存内存中或者数据库中。在进行连接时进行比对，识别客户端。

五、遇到的问题

基本没有问题，除了在 Rapidjson 的使用方法查找上。

六、思考与总结

1. 学到了在 C++中创建解析 json 对象的工具——Rapidjson
2. 学会了使用 Cocos 的网络 HTTP 模块，并且和 Web2.0 课程中学到的内容有了呼应交融。