

现代操作系统应用开发实验报告

学号： 15331046

班级： 晚上班

姓名： 陈志扬

实验名称： HW15 network

一 . 参考资料

请在这里列出对本实验有帮助你所参考的资料或者网站。

资料：老师上课提供的课件、TA 师兄提供的实验作业指导课件

网站：官方 API：<http://api.cocos.com/>

button 事件 addClickListener：

http://api.cocos.com/d8/d62/classcocos2d_1_1ui_1_1_widget.html#afb1cee076c2cb7c96a8769a68c5a7d90 或者

button 事件 addTouchEventListeners：

http://api.cocos.com/d8/d62/classcocos2d_1_1ui_1_1_widget.html#a79357b8c2d2a4385ce0586cf4f0f5ca1 这两个都可以实现 button 事件监听处理

HttpClient session 详解：http://blog.csdn.net/yj_cs/article/details/48271977

二 . 实验步骤

请在这里简要写下你的实验过程。

- ① 首先，学习老师课件的内容知识。主要是了解一些计算机网络基本概念，学习使用 HttpClient 和 HttpClient Session，学习使用 rapidjson 解析 JSON 语法。

下面是使用 `HttpClient` 的七个步骤：

1. 引入头文件和命名空间

```
#include "network/HttpClient.h"
using namespace cocos2d::network;
```

2. 使用 `HttpRequest` 无参数的构造函数构建实例

```
HttpRequest* request = new HttpRequest();
```

3. 设置连接方法的类型和待连接的地址

```
request->setRequestType(HttpRequest::Type::GET);
request->setUrl("http://www.httpbin.org/get");
```

4. 设置回调

```
Request->setResponseCallback(
    CC_CALLBACK_2>HelloWorld::onHttpComplete,this)
);
```

5. 添加请求到 `HttpClient` 任务队列

```
cocos2d::network::HttpClient::getInstance()->send(request);
```

6. 释放连接

```
request->release();
```

7. 处理网络回调函数

```
void HelloWorld::onHttpRequestCompleted(HttpClient *sender, HttpResponse * response) {
    if (!response) {
        return;
    }
    if (!response->isSucceed())
    {
        log("response failed");
        log("error buffer: %s", response->getErrorBuffer());
        return;
    }
    std::vector<char> *buffer = response->getResponseData();
    printf("Http Test, dump data: ");
    for (unsigned int i = 0; i < buffer->size(); i++)
    {
        printf("%c", (*buffer)[i]);
    }
    printf("\n");
}
```

② 然后先做登录 `login` 部分的网络请求。为 `login` 这个 `button` 添加一个监听处理事件 `addClickEventListener`，然后使用 `HttpClient` 将 `username` 作为请求数据发出 `POST` 请求，处理网络回调函数 `onHttpRequestCompleted_login`，利用本地数据库保存 `sessionid` 和返回的 `header` 和 `body`。具体代码如下：

```
#define database UserDefault::getInstance()
```

```

button->addClickEventListener(CC_CALLBACK_1(LoginScene::login_button_click, this));

void LoginScene::login_button_click(Ref * pSender)
{
    HttpRequest *request = new HttpRequest();
    request->setRequestType(HttpRequest::Type::POST);
    request->setUrl("http://localhost:8080/login");
    request->setResponseCallback(CC_CALLBACK_2(LoginScene::onHttpRequestCompleted_login, this));

    string playerName = "username=" + textField->getString();
    const char *postData = playerName.c_str();
    request->setRequestData(postData, strlen(postData));

    cocos2d::network::HttpClient::getInstance()->send(request);
    request->release();
}

void LoginScene::onHttpRequestCompleted_login(HttpClient * sender, HttpResponse * response)
{
    if (!response) return;
    if (!response->isSucceed()) {
        log("response failed");
        log("error buffer: %s", response->getErrorBuffer());
        return;
    }

    string responseData = Global::toString(response->getResponseData());
    string responseHeader = Global::toString(response->getResponseHeader());

    rapidjson::Document d;
    d.Parse<0>(responseData.c_str());
    // 解析错误
    if (d.HasParseError()) {
        CCLOG("GetParseError %s\n", d.GetParseError());
    }
    // login成功
    if (d.IsObject() && d.HasMember("result") && d["result"].GetBool()) {
        // 保存sessionId
        Global::gameSessionId = Global::getSessionIdFromHeader(responseHeader);
        database->setStringForKey("sessionId", Global::gameSessionId);
        // 保存header和body
        database->setStringForKey("header", responseHeader);
        database->setStringForKey("body", responseData);

        auto scene = GameScene::createScene();
        Director::getInstance()->replaceScene(scene);
    }
}

```

- ③ 接着做玩家提交游戏分数 **submit** 的功能。具体实现过程与 **login** 类似。为 **submit** 这个 **button** 添加一个监听处理事件 **addClickEventListener**，然后使用 **HttpClient** 将 **score** 作为请求数据发出 **POST** 请求(需要判断输入的 **score** 是否是数字)，处理网络回调函数 **onHttpRequestCompleted_submit**，利用本地数据库保存返回的 **header** 和 **body**。具体代码如下：

```

#define database UserDefault::getInstance()

submit_button->addClickEventListener(CC_CALLBACK_1(GameScene::submit_button_click, this));

void GameScene::submit_button_click(Ref *pSender) {
    bool temp = true;
    if (score_field->getString().length() > 0) {
        for (int i = 0; i < score_field->getString().length(); i++) {
            // 判断score是否是数字
            if (!isdigit(score_field->getString()[i])) {
                temp = false;
            }
        }
        if (temp) {
            HttpRequest *request = new HttpRequest();
            request->setRequestType(HttpRequest::Type::POST);
            request->setUrl("http://localhost:8080/submit");
            request->setResponseCallback(CC_CALLBACK_2(GameScene::onHttpRequestCompleted_submit, this));

            string score = "score=" + score_field->getString();
            const char* postData = score.c_str();
            request->setRequestData(postData, strlen(postData));

            vector<string> headers;
            headers.push_back("Cookie:GAMESESSIONID=" + Global::gameSessionId);
            request->setHeaders(headers);

            cocos2d::network::HttpClient::getInstance()->send(request);
            request->release();
        }
    }
}

void GameScene::onHttpRequestCompleted_submit(HttpClient *sender, HttpResponse *response) {
    if (!response) return;
    if (!response->isSucceed()) {
        log("response failed");
        log("error buffer: %s", response->getErrorBuffer());
        return;
    }
    string responseData = Global::toString(response->getResponseData());
    string responseHeader = Global::toString(response->getResponseHeader());

    rapidjson::Document d;
    d.Parse<0>(responseData.c_str());
    // 解析错误
    if (d.HasParseError()) {
        CCLOG("GetParseError %s\n", d.GetParseError());
    }
    // submit成功
    if (d.IsObject() && d.HasMember("result") && d["result"].GetBool()) {
        // 获取最高分数
        if (d.HasMember("info")) {
            // 获取最高分数填入输入框中
            score_field->setText(d["info"].GetString());
            // 保存header和body
            database->setStringForKey("header", responseHeader);
            database->setStringForKey("body", responseData);
        }
    }
}

```

④ 接着做玩家提交游戏分数 rank 的功能。具体实现过程与 login

和 submit 类似。为 rank 这个 button 添加一个监听处理事件 addClickEventListener，然后使用 HttpClient 将 top=10 作为请求数据发出 GET 请求，处理网络回调函数 onHttpRequestCompleted_rank，利用本地数据库保存返回的 header 和 body。具体代码如下：

```
rank_button->addClickEventListener(CC_CALLBACK_1(GameScene::rank_button_click, this));

void GameScene::rank_button_click(Ref *pSender) {
    HttpRequest *request = new HttpRequest();
    request->setRequestType(HttpRequest::Type::GET);
    request->setUrl("http://localhost:8080/rank?top=10");
    request->setResponseCallback(CC_CALLBACK_2(GameScene::onHttpRequestCompleted_rank, this));

    vector<string> headers;
    headers.push_back("Cookie:GAMESESSIONID=" + Global::gameSessionId);
    request->setHeaders(headers);

    cocos2d::network::HttpClient::getInstance()->send(request);
    request->release();
}
```

```

void GameScene::onHttpRequestCompleted_rank(HttpClient * sender, HttpResponse * response)
{
    if (!response) return;
    if (!response->isSucceed()) {
        log("response failed");
        log("error buffer: %s", response->getErrorBuffer());
        return;
    }
    string responseData = Global::toString(response->getResponseData());
    string responseHeader = Global::toString(response->getResponseHeader());

    rapidjson::Document d;
    d.Parse<0>(responseData.c_str());
    // 解析错误
    if (d.HasParseError()) {
        CLOG("GetParseError %s\n", d.GetParseError());
    }
    // rank成功
    if (d.IsObject() && d.HasMember("result") && d["result"].GetBool()) {
        if (d.HasMember("info")) {
            string rank = d["info"].GetString();
            for (int i = 0; i < rank.length(); i++) {
                // "|"转换为"\n"
                if (rank[i] == '|') rank[i] = '\n';
            }
            // rank排行榜
            rank_field->setText("\n" + rank);
            // 保存header和body
            database->setStringForKey("header", responseHeader);
            database->setStringForKey("body", responseData);
        }
    }
}

```

- ⑤ 加分项 B: 本地化保存 gameSessionId 等 cookies 信息, 实现自动登录功能。由于上面已经实现本地数据库存储 gameSessionId, 所以实现自动登录功能变得极其简单, 只需要在 AppDelegate.cpp 中加入这几行代码判断是否已存在 gameSessionId 即可。若存在, 切换为 GameScene 场景实现自动登录; 否则为 LoginScene 场景, 登录界面。具体代码如下:

```

#define database UserDefault::getInstance()

```

```
// 判断sessionid是否存在, 若存在, 切换为GameScene场景, 实现自动登录
if (database->getStringForKey("sessionid") != "") {
    Global::gameSessionId = database->getStringForKey("sessionid");
    auto scene = GameScene::createScene();
    director->runWithScene(scene);
}
else {
    auto scene = LoginScene::createScene();
    director->runWithScene(scene);
}
```

⑥ 加分项 A: 思考 HttpClient 的 enableCookies 函数的作用。

HttpClient session 详解中:

http://blog.csdn.net/yj_cs/article/details/48271977

在3.x的版本中, HttpClient::enableCookies方法支持http会话使用cookie。由于sessionID的保存和传递是通过cookie实现的, 所以我们可以很方便的使用HttpClient::enableCookies方法来实现http client session。

```
cocos2d::network::HttpClient::getInstance()->enableCookies(NULL);
```

我尝试着把这行代码加到如下图中的位置: 在 login、submit、rank 三个请求中都加入这行代码。

```
cocos2d::network::HttpClient::getInstance()->enableCookies(NULL);
cocos2d::network::HttpClient::getInstance()->send(request);
request->release();
```

官方文档中说 enableCookies 函数是默认保存了一个 cookieFile.txt 文件, 编译运行后在 UserDefault.xml 文件路径中发现了这个 cookieFile.txt 这个文件, 如下图:

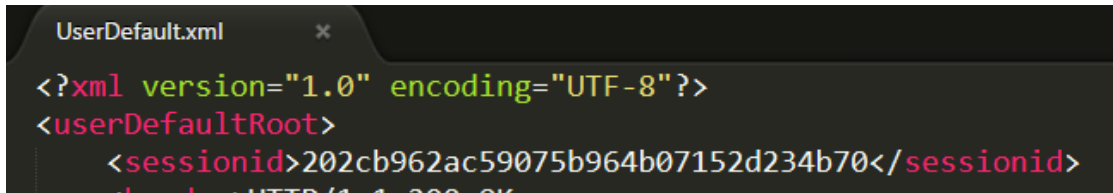
> OS (C:) > 用户 > linji > AppData > Local > HelloCocos			
名称	修改日期	类型	大小
cookieFile.txt	2017/6/4 19:32	文本文档	
UserDefault.xml	2017/6/4 19:32	XML 文件	

打开 cookieFile.txt 文件, 可以看到如下:

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

localhost      FALSE   /      FALSE   0      GAMESESSIONID
202cb962ac59075b964b07152d234b70
```

而 UserDefault.xml 中对应的 sessionId 也是相同的：



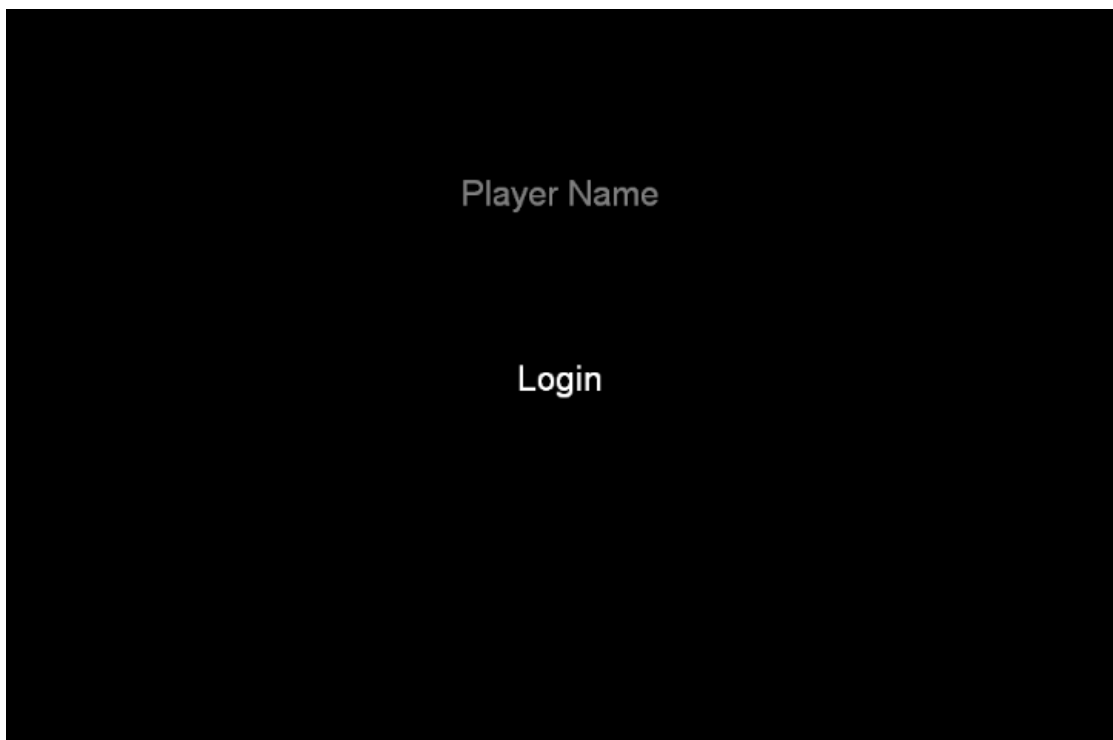
```
UserDefault.xml
<?xml version="1.0" encoding="UTF-8"?>
<userDefaultRoot>
  <sessionId>202cb962ac59075b964b07152d234b70</sessionId>
  <language>HTTP/1.1 200 OK
```

因此，我们可以了解到 enableCookies 函数通过创建本地文件 cookieFile.txt 从而实现本地化保存 gameSessionId 等 cookies 信息。

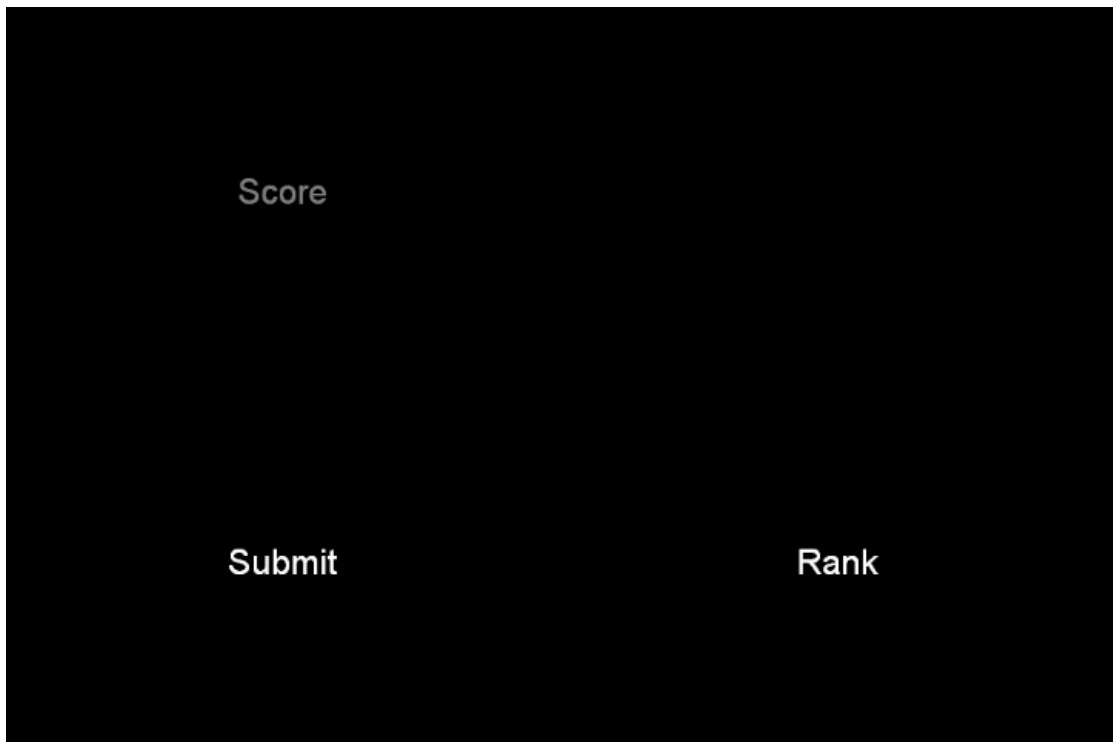
三．实验结果截图

请在这里把实验所得的运行结果截图。

① 程序 login 登录界面：



② 登录之后界面：



③ 因为我把 header 和 body 都保存在 UserDefault.xml 中，所以
登录之后我们查看 UserDefault.xml 文件，可以看到文件内容
如下：

```
UserDefault.xml x
<?xml version="1.0" encoding="UTF-8"?>
<userDefaultRoot>
  <sessionId>202cb962ac59075b964b07152d234b70</sessionId>
  <header>HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Set-Cookie: GAMESESSIONID=202cb962ac59075b964b07152d234b70

Content-Type: application/json;charset=UTF-8

Transfer-Encoding: chunked

Date: Sun, 04 Jun 2017 11:52:11 GMT

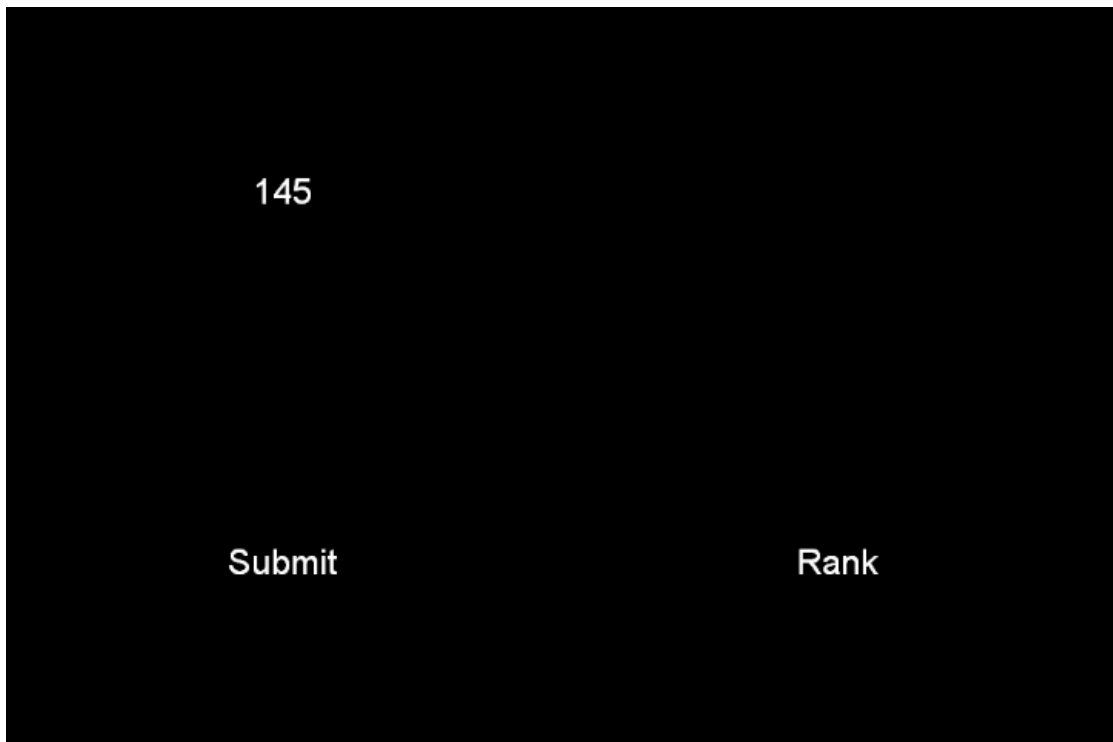
</header>
  <body>{"result":true,"info":"0"}</body>
</userDefaultRoot>
```

对比 `cookieFile.txt` 文件中的 `gameSessionId`，可以发现两者是相同的：

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

localhost      FALSE /      FALSE 0      GAMESESSIONID
202cb962ac59075b964b07152d234b70
```

④ 我们输入 145 作为 `score` 游戏分数，然后点击 `submit`：



由于 submit 后我也把返回的 header 和 body 保存在 UserDefault.xml 中，如图：可以看到 body 部分的 info 已改为当前最高分数 145：

```
UserDefault.xml
<?xml version="1.0" encoding="UTF-8"?>
<userDefaultRoot>
  <sessionId>202cb962ac59075b964b07152d234b70</sessionId>
  <header>HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Content-Type: application/json;charset=UTF-8

Transfer-Encoding: chunked

Date: Sun, 04 Jun 2017 11:57:26 GMT

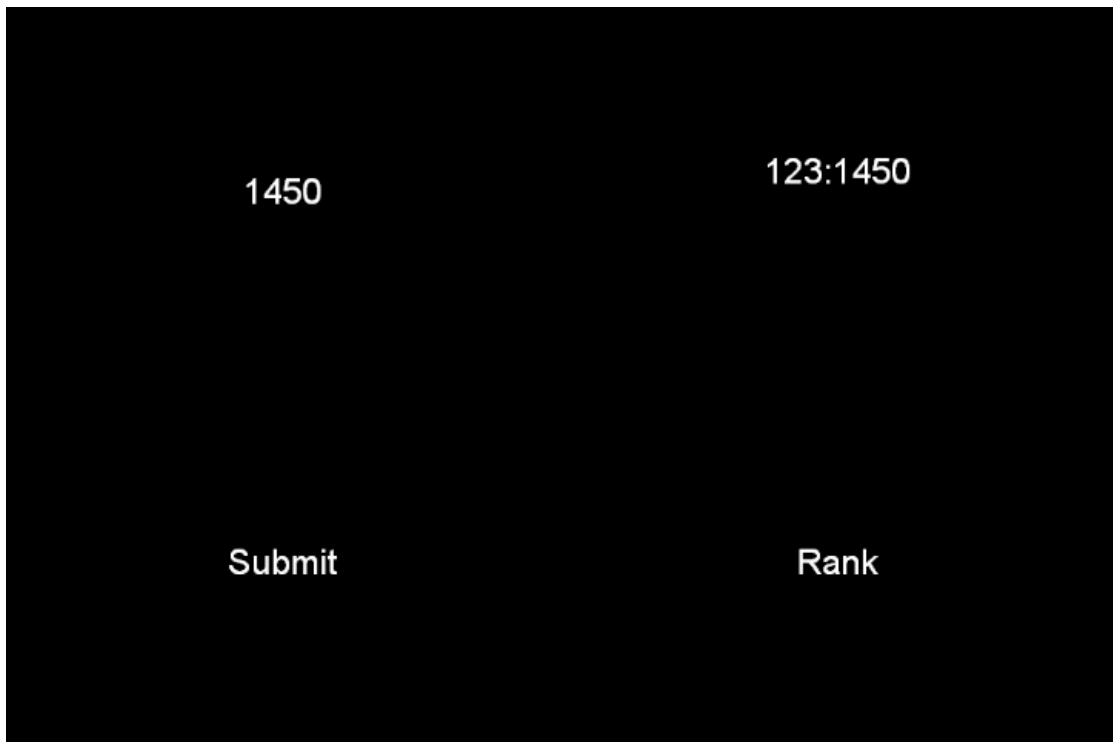

  </header>
  <body>{"result":true,"info":"145"}</body>
</userDefaultRoot>
```

⑤ 第一次 rank:



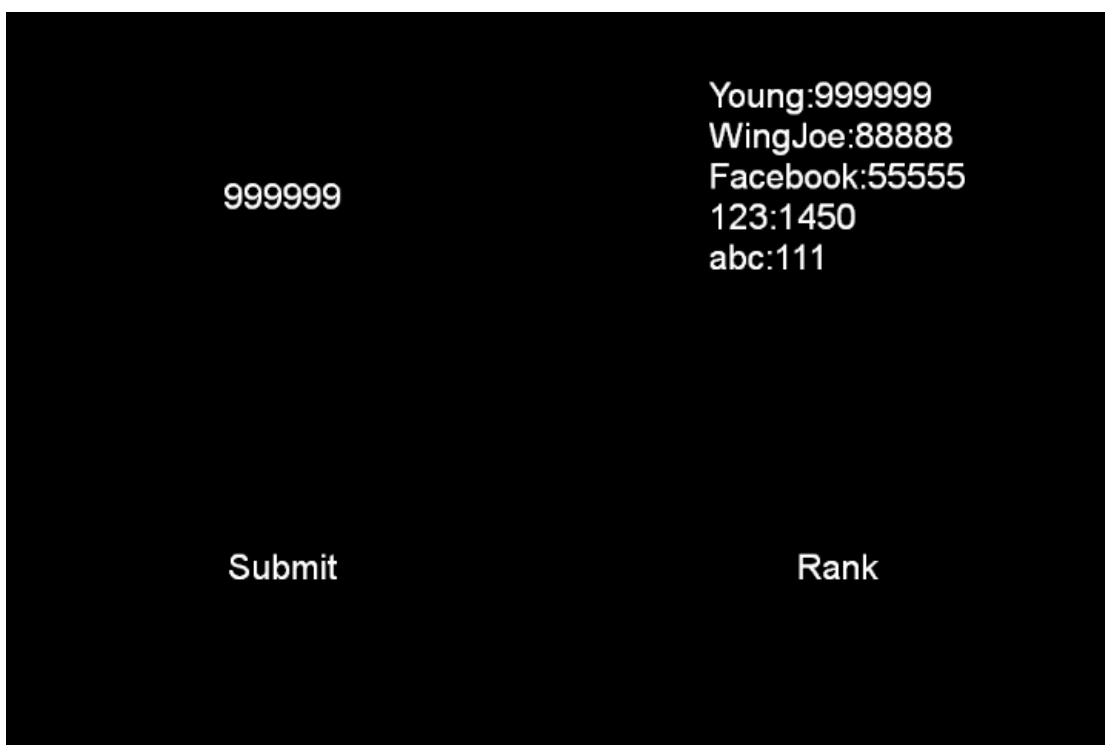
⑥ 然后修改 145 为 14, 由于 14 小于 145, 所以点击 submit, rank 后 score 依然显示 145, 保存的 body 中 info 也不会修改, 依然是 145 (截图略, TA 从截图中无法看出来)

⑦ 然后修改 145 为 1450, 由于 1450 大于 145, 所以点击 submit, rank 后 score 改为 1450, 保存的 body 中的 info 也改为 1450, 如图:



```
</header>  
  <body>{"result":true,"info":"|123:1450|"}</body>  
</userDefaultRoot>
```

- ⑧ 关闭程序，注册多个用户名 username，然后依次赋予分数，并点击 rank（设置最多 10 个，这里不举那么多例子了），结果如下图：



UserDefault.xml 文件中 body 部分如下图：

```
</header>
  <body>{"result":true,"info":"|Young:999999|WingJoe:88888|Facebook:55555|123:1450|abc:111|"}</body>
</userDefaultRoot>
```

⑨ 关闭程序，不修改 sessionid，重新运行程序，可以实现自动登录（这里截图省略，TA 师兄同样无法辨别（破涕而笑））

四．实验过程遇到的问题

请在这里写下你在实验过程中遇到的问题以及解决方案。

1. 一个比较坑的问题是 rank 的 top 参数怎么传的问题，一开始直接按照作业指导发送 Url `http://localhost:8080/rank`，结果一直显示错误 no provided top，然后在群里发现也有同学遇到同样的问题，最后发现要把 top 参数放到 Url 中：

```
request->setUrl("http://localhost:8080/rank?top=10");
```

2. 对作业指导中的返回值不是很理解，不知道怎么把它显示出来。

后来我直接把它保存在本地中，才真正理解了返回值中 `body` 部分的 `result` 和 `info` 的具体含义。

3. 要注意 **GET** 和 **POST**。GET 是获取数据，POST 是发送数据，不能混淆。

五. 思考与总结

请在这里写下你本次试验的心得体会以及所思所想。

在上课时，老师讲了很多计算机网络的知识，我自己觉得很难理解，以为这次作业又和上次那么难了，但整个作业完成后，发现其实很简单，虽然并不能完全理解计算机网络的基本知识，但是还是学会了 `cocos2d-x` 中 `HttpClient` 的使用。

总的来说，本次实验作业不算难，很多代码可以参考课件，接下来要好好学习计算机网络知识，服务器客户端非常重要！

接下来应该没有实验作业了，剩下一个期末项目，加油！

1. 实验报告提交格式为 pdf。
2. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。