

Android HTTP & JSON Lab


一、实验目的

1. 掌握 Android 用 HTTP 访问网络的方法
2. 掌握 JSON 的分析与使用

二、实验任务

设计一个支持师生在线同步课堂讨论任务的 Android 程序，下面是程序预期功能：

教师：

1. 添加：点击 ，在 dialog 中输入讨论主题，讨论内容，开始时间，结束时间。在点击确定后实时自动将该讨论项目插入服务器数据库，同时客户端也能看到新增的讨论项目。

讨论主题		
讨论内容		
12	44	AM
1	:	45 PM
2	46	
12	44	AM
1	:	45 PM
2	46	
Cancel		OK

2. 删除：对任何一个讨论项目长按，会出现是否删除的 dialog，点击 OK 就会删除该讨论项目，同时也将从服务器数据库中删除。



第二个讨论

警告!

是否删除这次讨论？

NO
OK

Note: 教师的添加和删除动作都是实时跟后台数据库同步的！

学生：

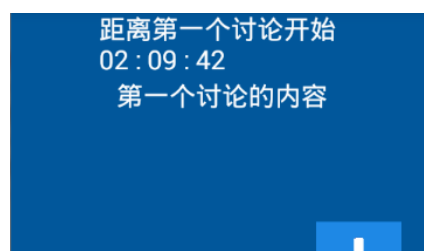
1. 查看单条讨论项目：点击任何一条讨论项目，会出现描述该讨论项目的 dialog，详细说明该讨论的主题，内容，开始时间和结束时间。



2. 获取讨论任务：点击菜单键，点击“获取讨论任务”将从服务器数据库拉取教师设定好的讨论项目，并在下面的列表中更新。

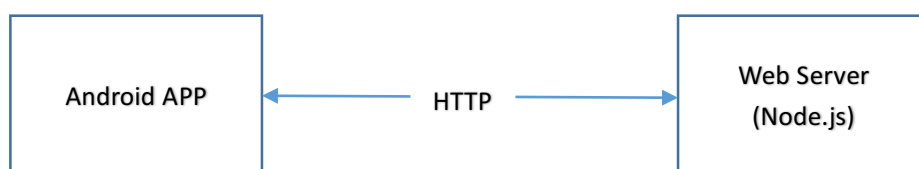


3. 当前状态：若当前处于任何一条讨论项目的时间内，则显示距离该讨论结束的倒计时，并显示讨论具体内容。若当前处于讨论间隙，则显示距离下个讨论开始的倒计时，并显示讨论具体内容。



三、实验原理

系统架构



本程序得以实现主要依靠 http,json 以及基于 node.js 的服务器之间互相配合。

本实验使用 http get 和 http post 来进行数据传输。

HTTP 是一个客户端和服务端请求和应答的标准（TCP）。客户端是终端用户，服务器端是网站。通过使用 Web 浏览器、网络爬虫或者其它的工具，客户端发起一个到服务器上指定端口（默认端口为 80）的 HTTP 请求。（我们称这个客户端）叫用户代理（user agent）。应答的服务器上存储着（一些）资源，比如 HTML 文件和图像。（我们称）这个应

答服务器为源服务器（origin server）。尽管 TCP/IP 协议是互联网上最流行的应用，HTTP 协议并没有规定必须使用它和（基于）它支持的层。事实上，HTTP 可以在任何其他互联网协议上，或者在其他网络上实现。HTTP 只假定（其下层协议提供）可靠的传输，任何能够提供这种保证的协议都可以被其使用。

GET 请求的数据会附在 URL 之后（就是把数据放置在 HTTP 协议头中），以?分割 URL 和传输数据，参数之间以&相连，如：

login.action?name=hyddd&password=idontknow&verify=%E4%BD%A0%E5%A5%BD。如果数据是英文字母/数字，原样发送，如果是空格，转换为+，如果是中文/其他字符，则直接把字符串用 BASE64 加密，得出如：%E4%BD%A0%E5%A5%BD，其中%XX 中的 XX 为该符号以 16 进制表示的 ASCII。而 POST 把提交的数据则放置在是 HTTP 包的包体中。

本实验上传和下载讨论项目时使用 json 来进行数据封装。

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于 JavaScript（Standard ECMA-262 3rd Edition - December 1999）的一个子集。JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。这些特性使 JSON 成为理想的数据交换语言。易于人阅读和编写，同时也易于机器解析和生成(网络传输速度)。

语法：

- 数据在名称/值对中
- 数据由逗号分隔
- 花括号保存对象
- 方括号保存数组

JSON 名称/值对：JSON 数据的书写格式是：名称/值对。名称/值对组合中的名称写在前面（在双引号中），值对写在后面(同样在双引号中)，中间用冒号隔开：

"firstName":"John"

JSON 值可以是：

- 数字（整数或浮点数）
- 字符串（在双引号中）
- 逻辑值（true 或 false）
- 数组（在方括号中）
- 对象（在花括号中）
- Null

本实验服务器基于 node.js 搭建，负责与客户端和后台数据库通信，让客户端和数据库可以交换数据。

Node.js 是一个基于 Chrome JavaScript 运行时建立的平台，用于方便地搭建响应速度快、易于扩展的网络应用。Node.js 使用事件驱动，非阻塞 I/O 模型而得以轻量 and 高效，非常适合在分布式设备上运行的数据密集型的实时应用。Node.js 是一个 Javascript 运行环境(runtime)。实际上它是对 Google V8 引擎进行了封装。V8 引擎执行 Javascript 的速度非常快，性能非常好。Node.js 对一些特殊用例进行了优化，提供了替代的 API，使得 V8 在非浏览器环境下运行得更好。

当客户端通过 http get 发起查询请求时，服务器将查询数据库并用 json 返回查询信息。当客户端通过 http get 发起删除请求时，服务器根据 url 中的 item 字段删除数据库中的相应条目，并返回“delete_success”，当客户端通过 http post 发起插入请求时，服务器解析客户端上传的 json 数据，并将获得的数据插入数据库中，并返回“insert_success”。

四、实验步骤

服务器搭建

1. 打开 <https://c9.io/>注册账号并进入 dashboard
2. 点击 demo-project, 点击 START EDITING
3. 安装 MySQL: 在 IDE 的控制台中输入 `npm install mysql` 并回车, 控制台中会显示安装信息, 只需记住你的 username, 密码默认为空, host 为 localhost, database 为 c9, port 为 3306
4. 进入 MySQL 命令: 在 IDE 的控制台中输入 `mysql-ctl cli` 并回车
5. 切换 Database: 在 IDE 的控制台中输入 `use c9;`并回车 (注意分号)
6. 创建表: 在 IDE 的控制台中输入下面的 SQL 语句并回车。

```
CREATE TABLE task_list(           //创建一个名称为 task_list 的表, 包含以下变量
id INT PRIMARY KEY AUTO_INCREMENT, //自增的 int 型 id,并设置为首要的 key
name VARCHAR(20) NOT NULL,        //name 为 varchar 类型, 最大长度 20, 不能为空
content VARCHAR(2000) NOT NULL,    //讨论内容, 最长 2000 字符
start LONG NOT NULL,              //讨论开始时间, 长整型, 相对于当天开始, 以毫秒计
stop LONG NOT NULL                //结束时间
)(ENGINE=InnoDB DEFAULT CHARSET=utf8; //数据库引擎 InnoDB, 默认字符集 utf8
```

7. 编辑 server.js: 打开 node.js 下的 server.js, 将示例代码粘贴进去, 修改 mysql.createConnection 的 user 参数为安装 MySQL 时提示的 username。

```
var http = require("http");    //请求 http 模块, 并且把它赋值给 http 变量
```

```
// create a server
var server = http.createServer();
// 创建 MySQL 连接池
var mysql = require('mysql');
var pool = mysql.createPool ({
  host: 'localhost',
  user: 'hzw1199',    //修改用户名、密码等
  password: '',
  database:'c9',
  port: 3306
});
```

```
// Note: when spawning a server on Cloud9 IDE,
// listen on the process.env.PORT and process.env.IP environment variables
```

```
// Click the 'Run' button at the top to start your server,
```

```

// then click the URL that is emitted to the Output tab of the console

// 收到 request 请求时调用该函数，再根据 post 或者 get 请求分别调用不同函数
var requestFunction = function (req, res){
    req.setEncoding('utf8');//请求编码

    if (req.method == 'POST'){
        return postResponse(req, res);
    }
    return getResponse(req, res);
};

// 收到 post 请求时的响应函数，此时客户端往服务器上传 json 格式的讨论项目，
插入成功向客户端返回“insert_success”
var postResponse = function(req, res) {
    var info = "";
    req.addListener('data', function(chunk){
        info += chunk;
    })
    .addListener('end', function(){
        res.setHeader('content-type','text/html; charset=UTF-8'); // 响应编码
        console.log(info);
        var obj = JSON.parse(info); // 将读取的 string 用 json 解析
        pool.getConnection(function(err, connection) {
            if (err) console.log(err);
            //insert
            connection.query('insert into task_list(name,content,start,stop) values(''
+ obj.name + ''',' + obj.content + ''',' + obj.start + ',' + obj.stop + ''')', function (err1, res1) {
                if (err1) console.log(err1);
                res.end('insert_success','utf8'); // 返回 “insert_success”
                console.log("INSERT Return ==> ");
                console.log(res1);
                connection.release();
            });
        });
    });
};

// 收到 get 请求时的响应函数，此时客户端向服务器查询 json 格式的讨论项目或者
请求删除某条讨论项目。将返回 json 格式的讨论项目，若讨论项目为空，将返回“nothing”，
删除成功将返回“delete_success”
var getResponse = function (req, res){
    res.writeHead(200, {'Content-Type': 'text/plain; charset=UTF-8'});

    // 读取 get 的 URL 中的 name 和 item 信息，name 为 query 时是查询，name 为

```

delete 时删除数据库中 name=item 的讨论项目

```

var name = require('url').parse(req.url,true).query.name;
var item = require('url').parse(req.url,true).query.item;

if(name == 'query'){
    pool.getConnection(function(err, connection) {
        if (err) console.log(err);
        //query
        connection.query('select * from task_list', function (err2, rows) {
            if (err2) console.log(err2);

            console.log("SELECT ==> ");
            if(rows.length == 0){
                res.end('nothing');
            }else{
                for (var i in rows) {
                    console.log(rows[i]);
                    var json = JSON.stringify(rows); //将结果解析成 jsonstr
                    res.end(json); // 返回查询结果
                }
            }
            connection.release();
        });
    });
}else if(name == 'delete'){
    pool.getConnection(function(err, connection) {
        if (err) console.log(err);
        //delete
        connection.query('delete from task_list where name = ' + "\"" + item + "\"",
function (err0, res0) {
            if (err0) console.log(err0);
            console.log("DELETE Return ==> ");
            console.log(res0);
            res.end('delete_success','utf8'); // 返回 “delete_success”
            connection.release();
        });
    });
}

};

server.on('request',requestFunction); // 收到 request 请求时调用函数
server.listen(process.env.PORT, process.env.IP); // 监听 IP 和端口

```

8. 运行: Ctrl+S 保存后, 点击上方的 Run 按钮, 会出现一个新的控制台, 告知你 URL, 用

这个 URL 替换 Android 源码中 MainActivity 中的 URL 变量。

Android 程序编写

URLConnection 实验中用到的几个方法：

方法摘要	
int	getResponseCode() 从 HTTP 响应消息获取状态码。
String	getResponseMessage() 获取与来自服务器的响应代码一起返回的 HTTP 响应消息（如果有）。
void	setRequestMethod(String method) 设置 URL 请求的方法， GET POST HEAD OPTIONS PUT DELETE TRACE 以上方法之一是合法的，具体取决于协议的限制。
abstract void	connect() 打开到此 URL 引用的资源的通信链接（如果尚未建立这样的连接）。
void	setConnectTimeout(int timeout) 设置一个指定的超时值（以毫秒为单位），该值将在打开到此 URLConnection 引用的资源的通信链接时使用。
void	setReadTimeout(int timeout) 将读超时设置为指定的超时值，以毫秒为单位。
void	setRequestProperty(String key, String value) 设置一般请求属性。

JSONArray 实验中用到的几个方法：

方法摘要	
int	length() 返回数组中值的个数。
JSONObject	getJSONObject(int index) 返回处于 index 位置的 JSONObject。

JSONObject 实验中用到的几个方法：

方法摘要

JSONObject	put(String name,Object value) 将 value 映射到 name，若已存在该 name 的键值对则覆盖。
String	getString(String name) 如果存在则返回映射到 name 的值，若不存在则抛出异常。
String	toString () 将这个 JSONObject 编码成 JSON 字符串，例如： {"query":"Pizza","locations":[94043,90210]}。

有五个关键点需要自行编写，前三个在 MainActivity 中，后两个在 WebAccessTools 中，均已用段注释符标出，并指明编程目标。下面是实验步骤。

1. 点击“获取讨论任务”时从服务器拉取讨论项目，并在下面的 ListView 中展现。
 - a. 创建一个线程，下面的动作都在线程的 run 函数中进行


```
new Thread(new Runnable() {
                        @Override
                        public void run() {}
                    }).start();
```
 - b. 用 WebAccessTools 的 javaHttpGet 方法获取讨论项目，此处需熟悉 httpget 知识。已知通过 httpget 查询数据库接口：name=query
 - 若数据库中有讨论项目，则以 string 形式返回 json 格式讨论项目
 - 若数据库为空，返回字符串“nothing”

```
String respond = WebAccessTools.javaHttpGet(URL + "?name=query");//注意修改 URL
```
 - c. 根据返回的字符串判断有无数据
 - d. 若有数据
 - 清空 datas 和 datas_string 两个 List


```
datas.clear();
datas_string.clear();
```
 - 解析获取到的 json 格式数据，并存入 datas 和 datas_string


```
JSONArray json=new JSONArray(respond);
for(int i =0;i<json.length();i++){
    JSONObject obj = json.getJSONObject(i); //从 jsonstr 里获取 JSONObject
    timeset_1 = Long.parseLong(obj.getString("start")); // 从 JSONObject
    里获取 start 字段
    timeset_2 = Long.parseLong(obj.getString("stop")); // 从 JSONObject 里
    获取 stop 字段
    Long[] long_tmp = {timeset_1, timeset_2};
    String[] string_tmp = {obj.getString("name"), obj.getString("content")};
    datas.add(new_added, long_tmp);
    datas_string.add(new_added, string_tmp);
}
```
 - 调用 mHandler.obtainMessage(2).sendToTarget();刷新界面，再调用 saveArray(datas, datas_string);本地缓存两个 List。
 - e. 若无数据

- 清空 datas 和 datas_string 两个 List


```
datas.clear();
datas_string.clear();
```
 - 调用 mHandler.obtainMessage(2).sendToTarget();刷新界面,再调用 saveArray(datas, datas_string);本地缓存两个 List。
 - f. 若抛出异常,则调用 mHandler.obtainMessage(3).sendToTarget();发出连接服务器异常的 Toast。
2. 教师新增讨论项目时输入了讨论主题,讨论内容,开始时间和结束后,点击 OK,向服务器上传讨论项目,并在下面的 ListView 中展现新的项目。
- a. 创建一个线程,下面的动作都在线程的 run 函数中进行


```
new Thread(new Runnable() {
    @Override
    public void run() {}
}).start();
```
 - b. 创建 JSONObject,将刚刚设置的讨论主题,讨论内容,开始时间和结束时间,以 json 格式存储


```
JSONObject json = new JSONObject();
json.put("name",name_tmp);
json.put("content",content_tmp);
json.put("start",(long) (timePicker_start.getCurrentHour() * 3600000 +
timePicker_start.getCurrentMinute() * 60000));
json.put("stop",(long) (timePicker_stop.getCurrentHour() * 3600000 +
timePicker_stop.getCurrentMinute() * 60000));
```
 - c. 将 JSONObject 转换为 jsonstr


```
String jsonstr = json.toString();
```
 - d. 用 WebAccessTools 的 doHttpPost 方法向服务器传输讨论项目,此处需熟悉 http post 知识。**已知通过 http post 上传数据接口: 上传 json 格式数据,成功后服务器返回 "insert_success"。**

```
String respond = WebAccessTools.javaHttpPost(URL,jsonstr);
```
 - e. 若返回"insert_success",则调用


```
listener_start.onTimeSet(timePicker_start, timePicker_start.getCurrentHour(),
timePicker_start.getCurrentMinute());
listener_stop.onTimeSet(timePicker_stop, timePicker_stop.getCurrentHour(),
timePicker_stop.getCurrentMinute());
```

 来响应 listener_start 和 listener_stop 这两个 OnTimeSetListener 实现将新讨论主题的信息加入下面的 ListView。
 - f. 若抛出异常,则调用 mHandler.obtainMessage(3).sendToTarget();发出连接服务器异常的 Toast。
3. 教师删除项目后,从服务器数据库中删除对应讨论项目,同时从 ListView 中删除对应条目
- a. 创建一个线程,下面的动作都在线程的 run 函数中进行


```
new Thread(new Runnable() {
```

```
@Override
public void run() {}

}.start();
```

- b. 用 WebAccessTools 的 doHttpsGet 方法删除数据库中的讨论项目, 此处需熟悉 httpget 知识。已知通过 **httpget** 删除数据库条目接口: **name=delete&item=要删除的条目的 name 字段**。删除成功则返回字符串“delete_success”。

Note: 一般要删除的条目的 name 字段会包含中文, 所以需要对 name 字段进行转码 (不能直接对整个 url 转码) 进行转码: : URLEncoder.encode(String s, String charsetName);其中 s 为 name 字段, charsetName 为"UTF-8"。

```
String respond = WebAccessTools.javaHttpsGet(URL + "?name=delete&item="+
URLEncoder.encode(datas_string.get(position)[0], "UTF-8"));
```

- c. 根据返回字符串判断是否删除成功
- g. 若删除成功
- 从 datas 及 datas_string 两个 List 中移除位置为 position 的条目

```
datas.remove(position);
datas_string.remove(position);
```
 - 调用 mHandler.obtainMessage(2).sendToTarget();刷新界面, 再调用 saveArray(datas, datas_string);本地缓存两个 List。
- g. 若删除失败, 则调用 mHandler.obtainMessage(3).sendToTarget();发出连接服务器异常的 Toast。
- h. 若抛出异常, 则调用 mHandler.obtainMessage(3).sendToTarget();发出连接服务器异常的 Toast。

4. 使用 HTTP GET 通过 https 获取数据的函数 String javaHttpsGet (String serverURL)

- a. 设置信任的 TLS 证书

```
TrustManager[] trustAllCerts = new TrustManager[]{new X509TrustManager() {
    public X509Certificate[] getAcceptedIssuers() {
        return null;
    }
    public void checkClientTrusted(X509Certificate[] certs, String authType) {
    }
    public void checkServerTrusted(X509Certificate[] certs, String authType) {
    }
}};
// Install the all-trusting trust manager
SSLContext sc = SSLContext.getInstance("TLS");
sc.init(null, trustAllCerts, new SecureRandom());
HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
```

- b. 创建 URL, 并通过 HttpURLConnection 的 API 设置请求参数

```
URL url = new URL(serverURL);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");    // 使用 get 请求
conn.setConnectTimeout(10000);    // 连接超时 单位毫秒
conn.setReadTimeout(6000);    // 读取超时 单位毫秒
```

```
conn.connect();
```

- c. 通过 `InputStream` 获取输入流，所获得的 `resultData` 就是服务器返回的信息

```
InputStream is = conn.getInputStream();    //获取输入流
InputStreamReader isr = new InputStreamReader(is);
BufferedReader bufferReader = new BufferedReader(isr);
String inputLine  = "";
String resultData = "";
while((inputLine = bufferReader.readLine()) != null){
    resultData += inputLine;
}
```

5. 使用 HTTP POST 通过 https 获取数据的函数 `String javaHttpPost (String serverURL , String json)`

- a. 设置信任的 TLS 证书

```
TrustManager[] trustAllCerts = new TrustManager[]{new X509TrustManager() {
    public X509Certificate[] getAcceptedIssuers() {
        return null;
    }
    public void checkClientTrusted(X509Certificate[] certs, String authType) {
    }
    public void checkServerTrusted(X509Certificate[] certs, String authType) {
    }
}};
// Install the all-trusting trust manager
SSLContext sc = SSLContext.getInstance("TLS");
sc.init(null, trustAllCerts, new SecureRandom());
HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
```

- d. 创建 URL，并通过 `HttpURLConnection` 的 API 设置请求参数

```
URL url = new URL(serverURL);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("POST");    // 使用 get 请求
conn.setConnectTimeout(10000);    // 连接超时 单位毫秒
conn.setReadTimeout(6000);    // 读取超时 单位毫秒
```

- b. 通过 `HttpURLConnection` 的 API 设置报文 Header

```
conn.setRequestProperty("Content-Type", "application/json;charset=utf-8");
conn.setRequestProperty("Accept-Charset", "utf-8");
conn.connect();
```

- c. 通过 `PrintWriter` 将 `jsonstr` 通过输出流上传至服务器，注意字符集设置为 UTF-8

```
PrintWriter      outprint      =      new      PrintWriter(new
OutputStreamWriter(conn.getOutputStream(), "UTF-8"));
outprint.write(json);
outprint.flush();
outprint.close();
```

- d. 通过 `InputStream` 获取输入流，所获得的 `resultData` 就是服务器返回的信息

```
InputStream is = conn.getInputStream();
InputStreamReader isr = new InputStreamReader(is);
BufferedReader bufferReader = new BufferedReader(isr);
String inputLine = "";
String resultData = "";
while((inputLine = bufferReader.readLine()) != null){
    resultData += inputLine;
}
```

五、 思考题

- 在获取讨论任务时让下载下来的讨论任务按开始时间的顺序在 **ListView** 中排列。
- 尝试理解创建表的 **MySQL** 语句后修改 **MySQL** 表的构造，新增字段，修改服务器代码，并在 **Android** 程序上作出相应的支持新字段的修改。
- 利用 **AsyncTask** 代替线程，完成同样的工作

六、 实验报告要求

- 成功搭建自己的服务器，并提供服务器成功运行的控制台截图。
- 根据实验步骤完成实验内容。
- 要求有 **Android** 程序成功获取讨论任务及成功添加讨论任务的截图，将自己的学号作为讨论项目名称添加一条讨论项目。