|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
| **实验报告** | | | | | | |
|  | 课程名称 |  | **移动应用软件设计** |  | |  |
| 专业班级 |  | 信计182 |  | |
| 姓 名 |  | 张家政 |  | |
| 学 号 |  | 1181010050 |  | |
| 联系电话 |  |  |  | |
|  |  | | | |  | |
|  |  | | | |  | |

# 实验七 客户端与服务器端交互

## 一、实验目的

1. 掌握Android客户端与服务器端网络连接和访问
2. 熟悉MySQL数据库及Servlet基本设计方法
3. 掌握Android HttpClient的使用方法
4. 掌握Android多线程和网络访问的使用方法

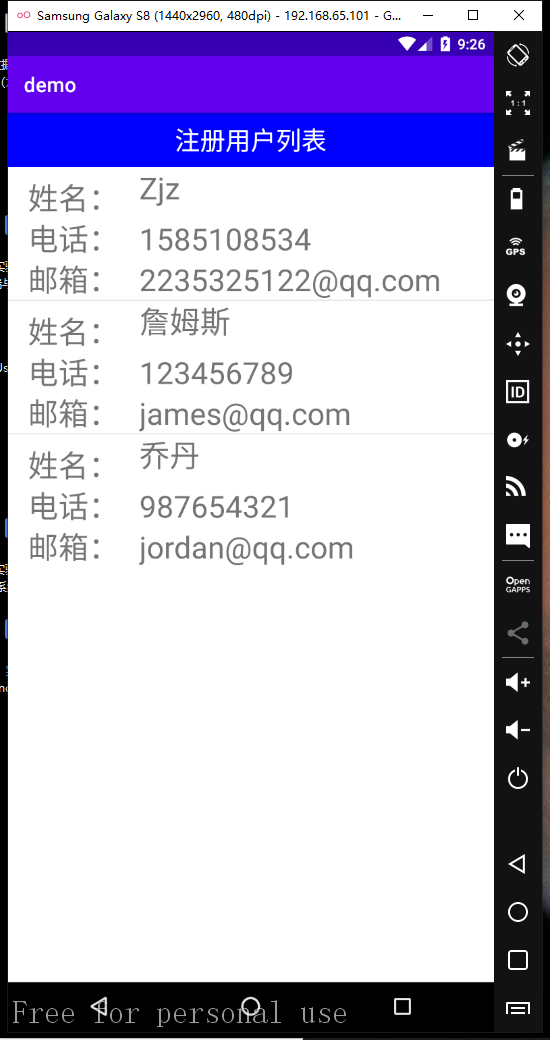
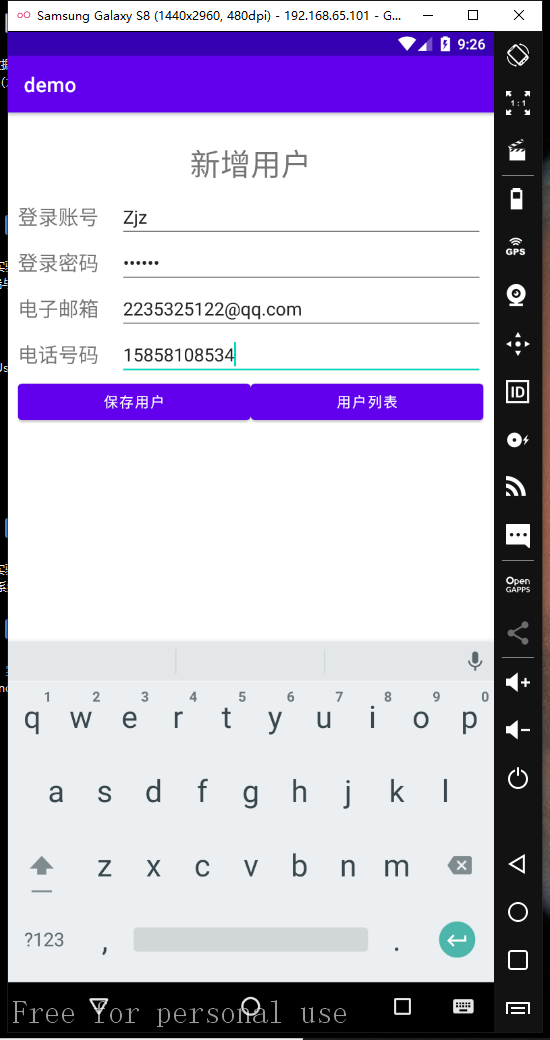
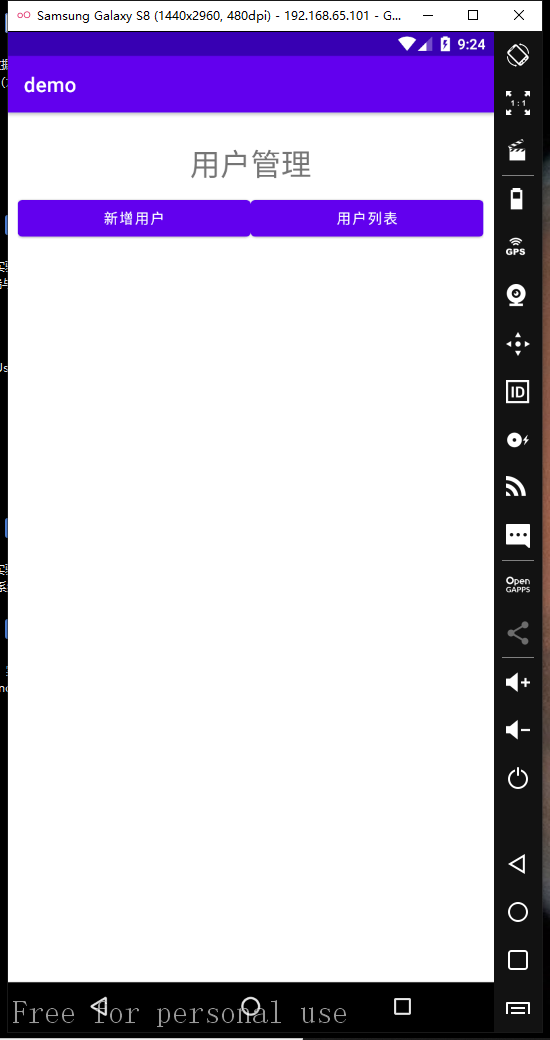
## 二、实验要求

编写程序，实现基于Android客户端与服务器端交互的会员注册、登录、个人资料管理等功能，要求数据存储在MySQL数据库中，Servlet提供后台数据服务接口，Android做为客户端程序。

## 三、实验步骤

## **1.实验效果**

**主页面-------------------------->用户注册---------------->浏览用户列表**



## 代码

### 后端代码：

### 安卓端代码：

新增用户的逻辑代码：

public class UserAddActivity2 extends AppCompatActivity {

    private EditText et\_username,et\_userpassword,et\_number,et\_email;

    private Button btn\_save,btn\_display;

    //http://172.19.170.146:8080/addServlet

    private String url = HttpUtil.BASE\_URL+"/addServlet";

    private Map<String, String> map;

    private Handler handler = new Handler(){

        @Override

        public void handleMessage(Message msg) {

            if(msg.what == 1){

                //新增成功

                Toast.makeText(UserAddActivity2.this,"新增成功",Toast.LENGTH\_SHORT).show();

            }else if(msg.what == 0){

                //新增失败

                Toast.makeText(UserAddActivity2.this,"新增失败",Toast.LENGTH\_SHORT).show();

            }

        }

    };

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_user\_add);

        et\_username = (EditText) findViewById(R.id.username);

        et\_userpassword = (EditText) findViewById(R.id.userpassword);

        et\_number = (EditText) findViewById(R.id.number);

        et\_email = (EditText) findViewById(R.id.email);

        btn\_save = (Button) findViewById(R.id.save);

        btn\_display = (Button) findViewById(R.id.display);

        btn\_save.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                String username = et\_username.getText().toString().trim();

                String password = et\_userpassword.getText().toString().trim();

                String number = et\_number.getText().toString().trim();

                String email = et\_email.getText().toString().trim();

                map = new HashMap<String, String>();

                map.put("username",username);

                map.put("password",password);

                map.put("number",number);

                map.put("email",email);

                new Thread(){

                    public void run(){

                        try {

                            String returndata = HttpUtil.postRequest(url,map);

                            JSONObject jsondata = new JSONObject(returndata);

                            Message msg = new Message();

                            if(jsondata.getBoolean("flag")){

                                //新增成功

                                msg.what = 1;

                            }else{

                                msg.what = 0;

                                //新增失败

                            }

                            handler.sendMessage(msg);

                        } catch (Exception e) {

                            e.printStackTrace();

                        }

                    }

                }.start();

            }

        });

        btn\_display.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                startActivity(new Intent(UserAddActivity2.this,UserListActivity.class));

            }

        });

    }

}

用户列表展示的逻辑代码：

public class UserListActivity extends AppCompatActivity {

    private ListView list;

    List<Map<String,Object>> data;

    String url=HttpUtil.BASE\_URL+"/listServlet";

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_user\_list);

        list = (ListView)findViewById(R.id.listView1);

        List<Map<String,Object>> ListData = getListData();

        SimpleAdapter adapter = new SimpleAdapter(this,ListData, R.layout.list\_item,

                new String []{"username","number","email"},

                new int []{R.id.username,R.id.number,R.id.email});

        list.setAdapter(adapter);

    }

    private List<Map<String,Object>> getListData() {

        data = new ArrayList<Map<String, Object>>();

        new Thread() {

            public void run() {

                try {

                    String returndata = HttpUtil.getRequest(url);

                    JSONArray jsondata = new JSONArray(returndata);

                    for (int i = 0; i < jsondata.length(); i++) {

                        //获取json数组中的每一个元素

                        JSONObject jsonObject = jsondata.getJSONObject(i);

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

                        map.put("username", jsonObject.getString("username"));

                        map.put("number", jsonObject.getString("number"));

                        map.put("email", jsonObject.getString("email"));

                        data.add(map);

                    }

                } catch (Exception e) {

                    e.printStackTrace();

                }

            }

        }.start();

        return data;

    }

}

HttpUtil代码：

public class HttpUtil {

    // 正式更新URL

    public static final String BASE\_URL = "http://172.19.170.146:8080";

    private static final String TAG = "HttpUtil";

    // 获得Get请求对象request

    public static HttpGet getHttpGet(String url) {

        HttpGet request = new HttpGet(url);

        return request;

    }

    // 获得Post请求对象request

    public static HttpPost getHttpPost(String url) {

        HttpPost request = new HttpPost(url);

        return request;

    }

    // 根据请求获得响应对象response

    public static HttpResponse getHttpResponse(HttpGet request) throws ClientProtocolException, IOException {

        HttpResponse response = new DefaultHttpClient().execute(request);

        return response;

    }

    // 根据请求获得响应对象response

    public static HttpResponse getHttpResponse(HttpPost request) throws ClientProtocolException, IOException {

        HttpResponse response = new DefaultHttpClient().execute(request);

        return response;

    }

    // 发送Post请求，获得响应查询结果

    public static String queryStringForPost(String url) {

        HttpPost request = HttpUtil.getHttpPost(url);

        HttpParams httpParameters = new BasicHttpParams();

        HttpConnectionParams.setConnectionTimeout(httpParameters, 10000);

        HttpConnectionParams.setSoTimeout(httpParameters, 10000);

        DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);

        Log.i(TAG, url);

        String result = null;

        try {

            // HttpUtil.getHttpResponse(request);//原方法，没做网络超时处理

            HttpResponse response = httpClient.execute(request);// 新方案，做了网络超时设置

            // 判断是否请求成功

            if (response.getStatusLine().getStatusCode() == 200) {

                // 获得响应

                result = EntityUtils.toString(response.getEntity());

                return result;

            } else {

                Log.i(TAG, response.getStatusLine().getStatusCode() + "");

            }

        } catch (ClientProtocolException e) {

            e.printStackTrace();

            result = "网络异常！";

            return result;

        } catch (IOException e) {

            e.printStackTrace();

            result = "网络异常！";

            return result;

        }

        return null;

    }

    // 获得响应查询结果

    public static String queryStringForPost(HttpPost request) {

        String result = null;

        try {

            // 获得响应对象

            HttpResponse response = HttpUtil.getHttpResponse(request);

            // 判断是否请求成功

            if (response.getStatusLine().getStatusCode() == 200) {

                // 获得响应

                result = EntityUtils.toString(response.getEntity());

                return result;

            }

        } catch (ClientProtocolException e) {

            e.printStackTrace();

            result = "网络异常！";

            return result;

        } catch (IOException e) {

            e.printStackTrace();

            result = "网络异常！";

            return result;

        }

        return null;

    }

    // 发送Get请求，获得响应查询结果

    public static String queryStringForGet(String url) {

        // 获得HttpGet对象

        HttpGet request = HttpUtil.getHttpGet(url);

        String result = null;

        try {

            // 获得响应对象

            HttpResponse response = HttpUtil.getHttpResponse(request);

            // 判断是否请求成功

            if (response.getStatusLine().getStatusCode() == 200) {

                // 获得响应

                result = EntityUtils.toString(response.getEntity());

                return result;

            }

        } catch (ClientProtocolException e) {

            e.printStackTrace();

            result = "网络异常！";

            return result;

        } catch (IOException e) {

            e.printStackTrace();

            result = "网络异常！";

            return result;

        }

        return null;

    }

    /\*\*

     \*

     \* @param url

     \*            发送请求的URL

     \* @return 服务器响应字符串

     \* @throws Exception

     \*/

    public static String getRequest(String url) throws Exception {

        // 创建HttpGet对象。

        HttpGet get = new HttpGet(url);

        HttpParams httpParameters = new BasicHttpParams();

        HttpConnectionParams.setConnectionTimeout(httpParameters, 3000);

        HttpConnectionParams.setSoTimeout(httpParameters, 5000);

        DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);

        // 发送GET请求

        HttpResponse httpResponse = httpClient.execute(get);

        // 如果服务器成功地返回响应

        if (httpResponse.getStatusLine().getStatusCode() == 200) {

            // 获取服务器响应字符串

            String result = EntityUtils.toString(httpResponse.getEntity());

            return result;

        }

        return null;

    }

    /\*\*

     \* 请求远程服务器，并封装参数信息

     \* @param url

     \* @param rawParams

     \* @return

     \* @throws Exception

     \*/

    public static String postRequest(String url, Map<String, String> rawParams) throws Exception {

        // 创建HttpPost对象。

        HttpPost post = new HttpPost(url);

        // 如果传递参数个数比较多的话可以对传递的参数进行封装

        List<NameValuePair> params = new ArrayList<NameValuePair>();

        for (String key : rawParams.keySet()) {

            // 封装请求参数

            params.add(new BasicNameValuePair(key, rawParams.get(key)));

        }

        //Logger.i(TAG, "params------------------->" + params);

        // 设置请求参数

        post.setEntity(new UrlEncodedFormEntity(params, "UTF-8"));

        HttpParams httpParameters = new BasicHttpParams();

        HttpConnectionParams.setConnectionTimeout(httpParameters, 3000);

        HttpConnectionParams.setSoTimeout(httpParameters, 15000);

        DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);

        // 发送POST请求

        HttpResponse httpResponse = httpClient.execute(post);

        // 如果服务器成功地返回响应

        String result = null;

        if (httpResponse.getStatusLine().getStatusCode() == 200) {

            // 获取服务器响应字符串

            result = EntityUtils.toString(httpResponse.getEntity(), "UTF-8");

            Log.i(TAG, "result-------->" + result);

        }

        return result;

    }

}

1. **实验报告要求**
2. 严格按照实验报告格式完成实验报告并上交
3. 按照步骤对实验过程进行截图和分析（包括核心源代码、文字说明）