Android 案例开发 实验报告

成绩	

实验名称 无线点餐系统

专业班级 软件 172 班级学号 201715030208

姓 名 朱洪龙 实验日期 2019/06/15

（报告内容包括：实验题目、实验环境、实验步骤、实验结果、遇到的问题和解决办法、实验小结等。

目录

【实验题目】

1. 点餐系统前台主要实现登录验证、更新数据、点餐功能和结账功能；后台资源服务器使用Struts2 + Hibernate3.x 框架搭建REST 服务并返回JSON数据给前台（有些对象通过序列化的形式返回）。
2. 即时通讯目前实现闪屏后登录或注册，还有主界面中的三个 fragment（消息、联系人和动态）。

【实验环境】

OS:Windows 10
IDE:Android Studio 2.3.2
AVD:Nexus 5X API 23
minSdkVersion 14
targetSdkVersion:25(android 7.1.1)

【实验步骤、程序调试过程中所有遇到的问题和解决办法】

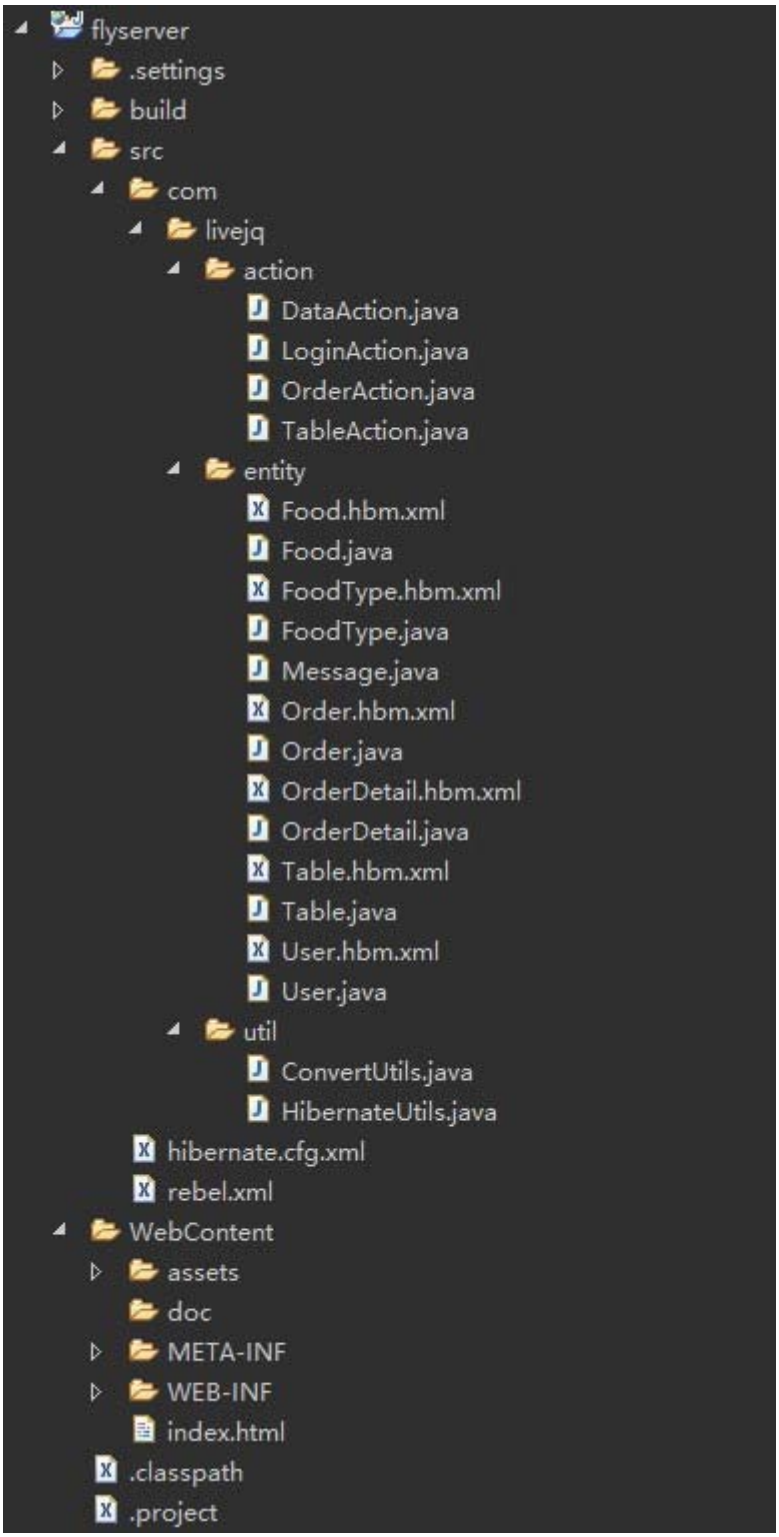
无线点餐系统

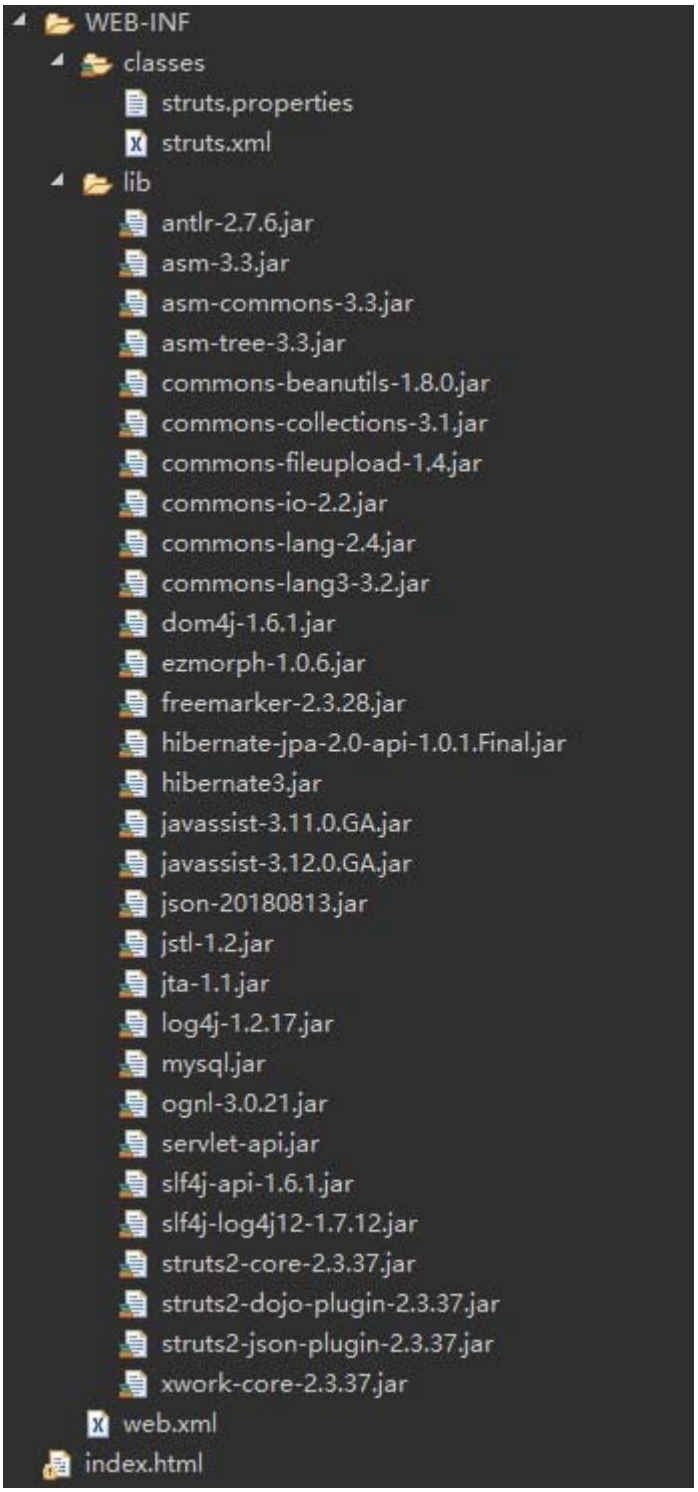
资源服务器的 REST 服务搭建

主要结构和配置：

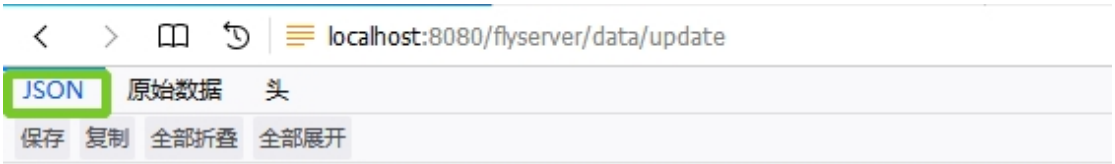
由于主要介绍前台 android 功能，所以后台服务器只是简单介绍

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <package name="login" namespace="/user" extends="json-default">
        <action name="*" class="com.livejq.action.LoginAction" method="{1}">
            <result type="json">
                <param name="excludeProperties">code,password</param>
            </result>
        </action>
    </package>
    <package name="order" namespace="/order" extends="json-default">
        <action name="*" class="com.livejq.action.OrderAction" method="{1}">
            <result type="json">
                <param name="excludeProperties">code,orderId</param>
            </result>
        </action>
    </package>
    <package name="data" namespace="/data" extends="json-default">
        <action name="*" class="com.livejq.action.DataAction" method="{1}">
            <result type="json"/>
        </action>
    </package>
    <package name="table" namespace="/table" extends="json-default">
        <action name="*" class="com.livejq.action.TableAction" method="{1}">
            <result type="json">
                <param name="excludeProperties">seats,needEmpty</param>
            </result>
        </action>
    </package>
</struts>
```


flyserver
.settings
build
src
com
livejq
action
DataAction.java
LoginAction.java
OrderAction.java
TableAction.java
entity
Food.hbm.xml
Food.java
FoodType.hbm.xml
FoodType.java
Message.java
Order.hbm.xml
Order.java
OrderDetail.hbm.xml
OrderDetail.java
Table.hbm.xml
Table.java
User.hbm.xml
User.java
util
ConvertUtils.java
HibernateUtils.java
hibernate.cfg.xml
rebel.xml
WebContent
assets
doc
META-INF
WEB-INF
index.html
.classpath
.project

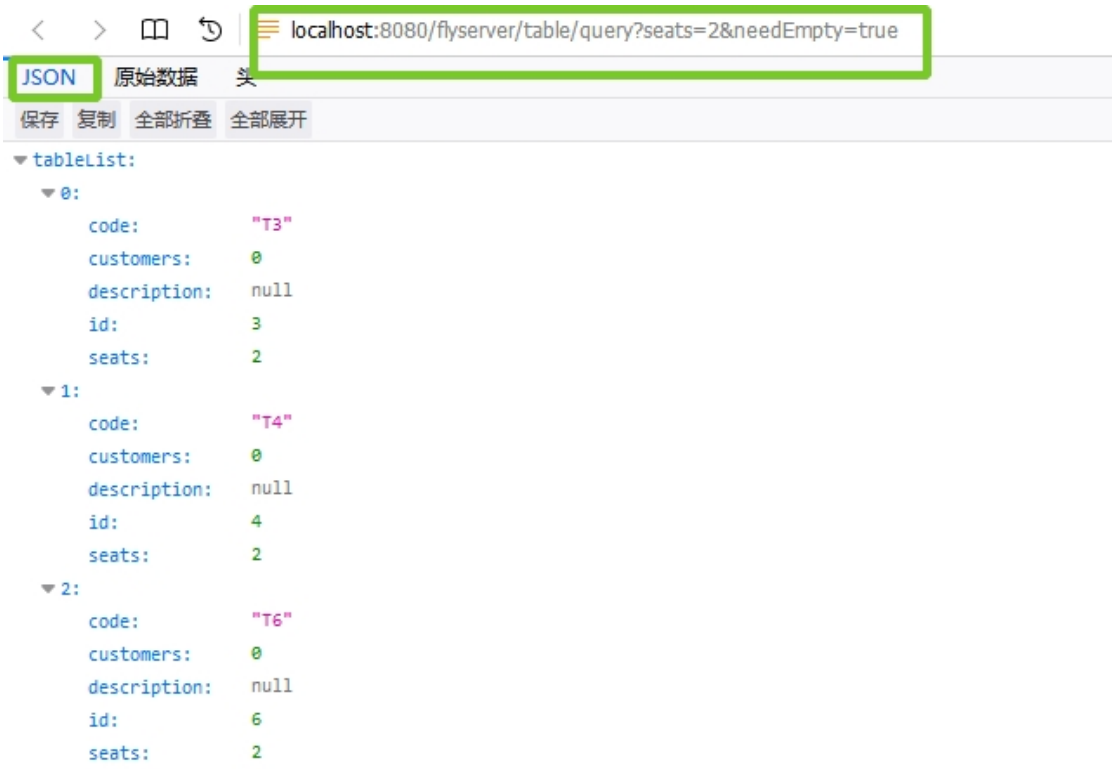

WEB-INF
classes
struts.properties
struts.xml
lib
antlr-2.7.6.jar
asm-3.3.jar
asm-commons-3.3.jar
asm-tree-3.3.jar
commons-beanutils-1.8.0.jar
commons-collections-3.1.jar
commons-fileupload-1.4.jar
commons-io-2.2.jar
commons-lang-2.4.jar
commons-lang3-3.2.jar
dom4j-1.6.1.jar
ezmorph-1.0.6.jar
freemarker-2.3.28.jar
hibernate-jpa-2.0-api-1.0.1.Final.jar
hibernate3.jar
javassist-3.11.0.GA.jar
javassist-3.12.0.GA.jar
json-20180813.jar
jstl-1.2.jar
jta-1.1.jar
log4j-1.2.17.jar
mysql.jar
ognl-3.0.21.jar
servlet-api.jar
slf4j-api-1.6.1.jar
slf4j-log4j12-1.7.12.jar
struts2-core-2.3.37.jar
struts2-dojo-plugin-2.3.37.jar
struts2-json-plugin-2.3.37.jar
xwork-core-2.3.37.jar
web.xml
index.html

返回的主要 **JSON** 数据：




```
foodTypes:
  0:
    id:          1
    name:        "热菜"
  1:
    id:          2
    name:        "凉菜"
  2:
    id:          3
    name:        "烧烤"
  3:
    id:          4
    name:        "酒类"
  4:
    id:          5
    name:        "其他"
foods:
  0:
    code:        "F1"
    description: null
    id:          1
    name:        "干炸里脊"
    price:       20
    typeId:      1
  1:
    code:        "F2"
    description: null
    id:          2
    name:        "水煮鱼"
    price:       40
    typeId:      1
  2:
    code:        "F3"
    description: null
    id:          3
    name:        "酸菜鱼"
    price:       40
    typeId:      1
  3:
    code:        "F4"
    description: null
    id:          4
    name:        "剁椒鱼头"
    price:       40
    typeId:      1
  4:
    code:        "F5"
    description: null
    id:          5
```


localhost:8080/flyserver/order/get?code=7502
JSON
原始数据
头
保存
复制
全部折叠
全部展开
order:
code: "7502"
customers: 1
description: "sdf"
foodId: 1
id: 8
name: "干炸里脊"
num: 1
orderTime: "2019-06-14T00:00:00"
price: 20
status: 1
tableId: 1
waiterId: 5
result: null


localhost:8080/flyserver/table/query?seats=2&needEmpty=true
JSON
原始数据
头
保存
复制
全部折叠
全部展开
tableList:
0:
code: "T3"
customers: 0
description: null
id: 3
seats: 2
1:
code: "T4"
customers: 0
description: null
id: 4
seats: 2
2:
code: "T6"
customers: 0
description: null
id: 6
seats: 2

1. 登录验证

难点：主要是发送 GET/POST 请求的的问题，老师给的案例代码中发送请求时是直接在 UI 界面中发送出去的，看看

它的 androidmenifest.xml 配置文件中 android 版本为 7~17，这是在很早以前所允许的。但如今在 android 高版本（我的是 14~25）中是禁止的，只能再开启一个线程中执行发送请求（即子线程）。

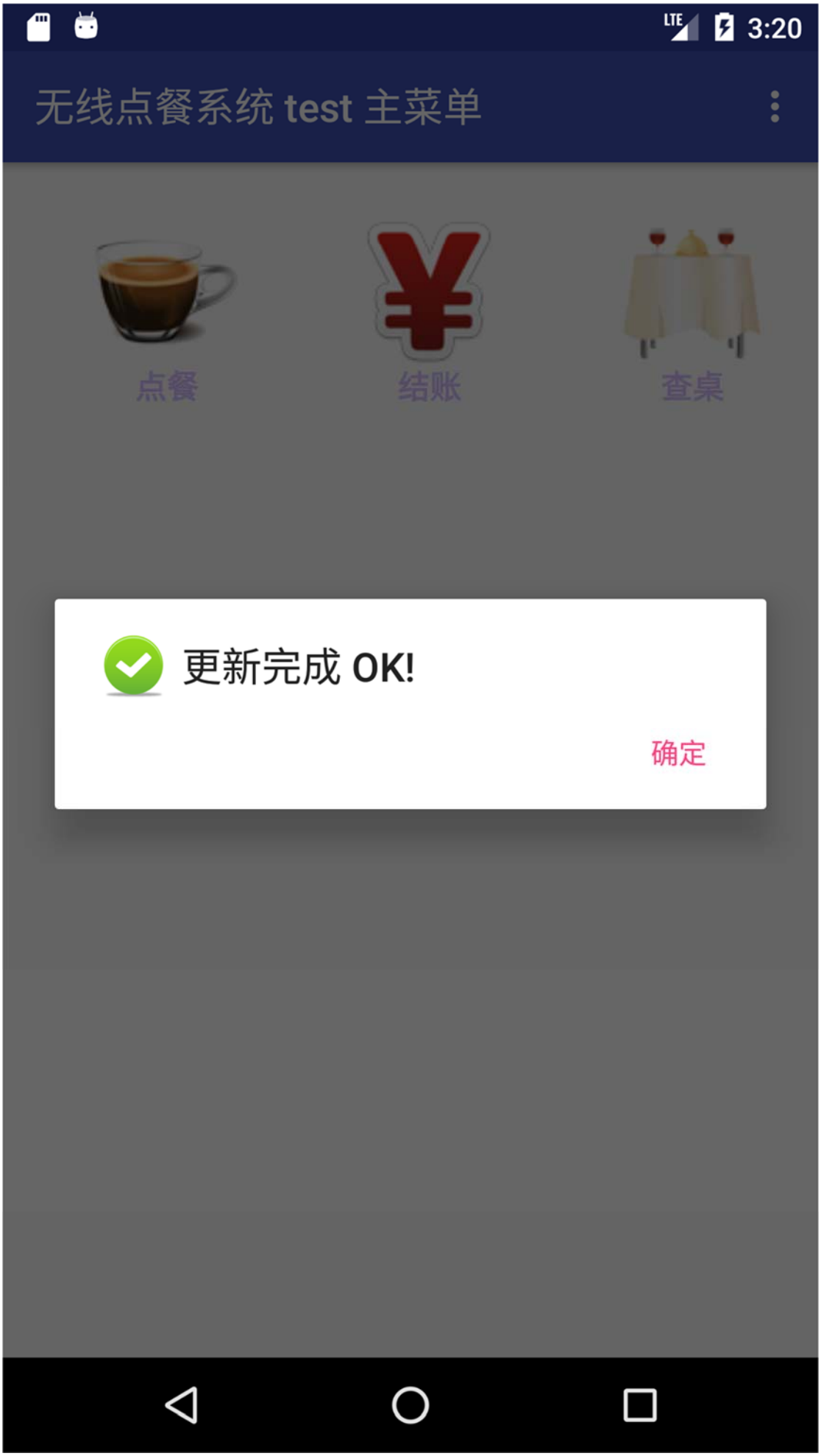
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "28.0.3"
    defaultConfig {
        applicationId "com.livejq.flyrestaurant"
        minSdkVersion 14
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.jakewharton:butterknife:5.1.1'
    testCompile 'junit:junit:4.12'
    compile files('libs/volley.jar')
}
```

```
        if(URL.length() == 0) {//设置默认服务器地址
            URL = "http://192.168.123.124:8080/flyserver";
//          URL = "http://192.168.43.232:8080/flyserver";
        }
        if(code.length() == 0 || password.length() == 0){
            showMessageDialog("请输入登录名或密码!", R.drawable.warning, null);
            return;
        }

        new Thread(new Runnable() {
            @Override
            public void run() {
                String url = URL + "/user/get?code=" + code + "&password=" + password;
//              String url = "http://192.168.123.124:8080/flyserver/user/get?code=test&password=test";
                String str = HttpUtils.get(url);
                User user = HttpUtils.parseJSONToUser(str);
                Message msg = handler.obtainMessage();
                msg.obj = user;
                handler.sendMessage(msg);
            }
        }).start();
```

```
handler = new Handler(Looper.getMainLooper()) {
    @Override
    public void handleMessage(Message msg) {
        User user = (User)msg.obj;
        App app = (App)getApplicationContext();
        if(user != null) {
            app.user = user;
            app.URL = URL;
            config.setProperty("URL", URL);
            Intent intent = new Intent(LoginActivity.this, MainActivity.class);
            startActivity(intent);
        }else {
            showMessageDialog("不存在此用户！", R.drawable.warning, null);
        }
    }
};
```

2. 更新数据


3:37
无线点餐系统 test 主菜单
点餐
结账
查桌
更新需要较长时间，确定要更新吗?
取消
确定



默认进入主界面时自动调用更新代码（因为可以将不会频繁改变的数据从服务器中获取过来，然后保存到手机的

sqlite 数据库中，以降低服务器的压力），

UpdateDataService

```
    * 从数据库中下载数据，并更新到本地数据库中
    * @throws InterruptedException
    */
   protected void updateData() throws InterruptedException {
       App app = (App)getApplicationContext();
       DBHelper dbHelper = new DBHelper(getApplicationContext());
       dbHelper.deleteDb();

       String url = app.URL + "/data/update";
       List data = null;
       try {
           data = HttpUtils.doUpdate(url);
       } catch (Exception e) {
           // TODO Auto-generated catch block
           e.printStackTrace();
       }

       List<Food> foods = (ArrayList<Food>) data.get(0);
       List<FoodType> foodTypes = (ArrayList<FoodType>) data.get(1);
       List<Table> tables = (ArrayList<Table>) data.get(2);

       SQLiteDatabase db = dbHelper.getWritableDatabase();
       String sql1 = "insert into food(id, code, type_id, name, price, description) values(?, ?, ?, ?, ?, ?)";
       String sql2 = "insert into food_type(id, name) values(?, ?)";
       String sql3 = "insert into tables(id, code, seats, customers, description) values(?, ?, ?, ?, ?)";
       db.beginTransaction();

       try {
           for(Food f : foods) {
               db.execSQL(sql1, new Object[] {f.getId(), f.getCode(), f.getTypeId(), f.getName(), f.getPrice(), f.getDescription()});
           }
```

3.点餐功能


3:21
点餐
选择酒菜
热菜
凉菜
烧烤
酒类
其他
水煮鱼 价格：40/每份
1
hello
取消
确定
Jello
hello
bello

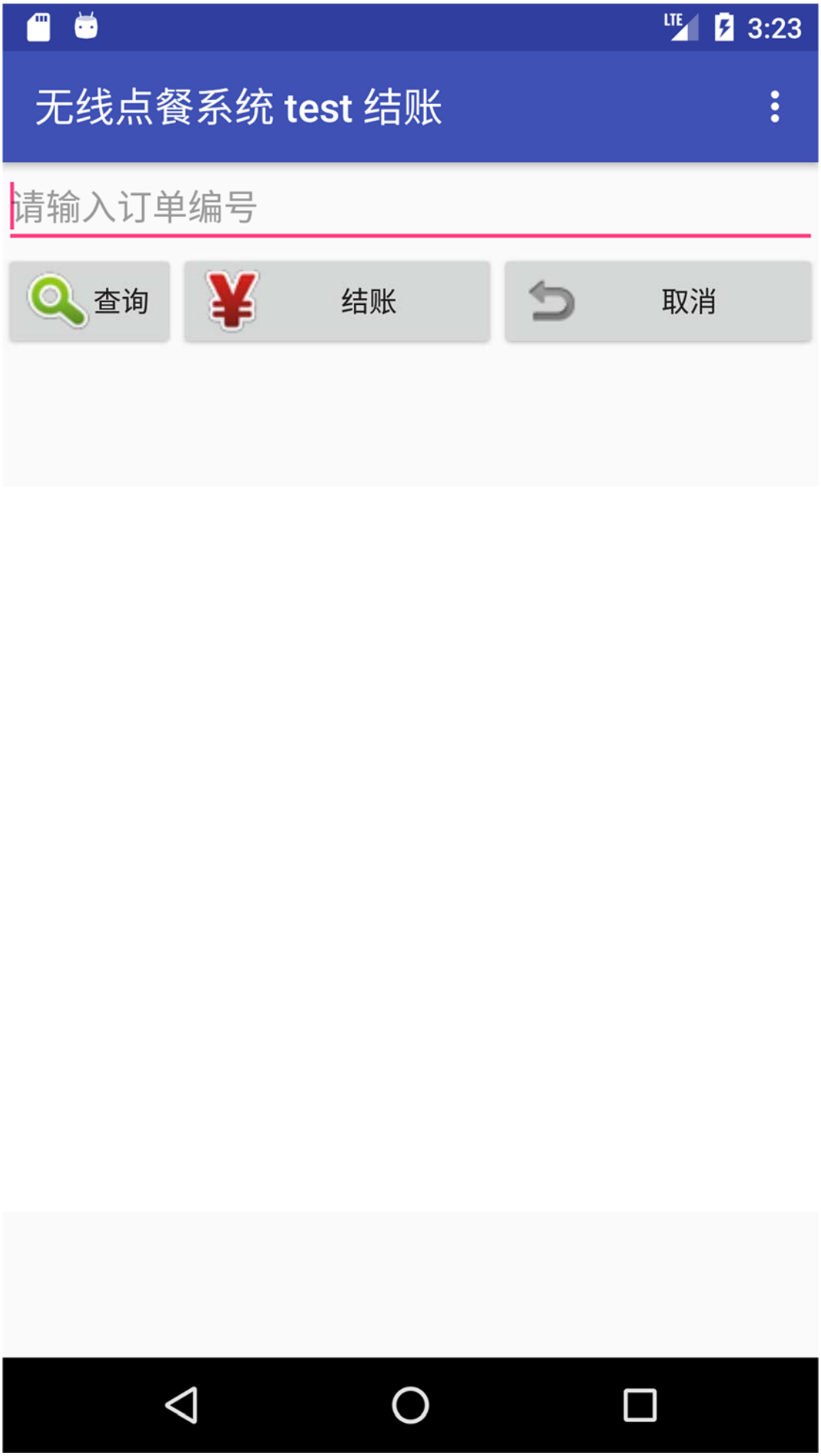
由于直接在 UI 中 new 一个线程是没有得到有效管理的（即野线程），所以在 JDK5.0 中引出了 ExecutorService 这个线

程管理服务，可以开启单线程或一个线程池并对开启的线程的生命周期进行有效管理，在这个管理中的 Call 或 Callable 中重写了 Runnable 接口方法，可以在执行完任务后返回一个 result 结果。

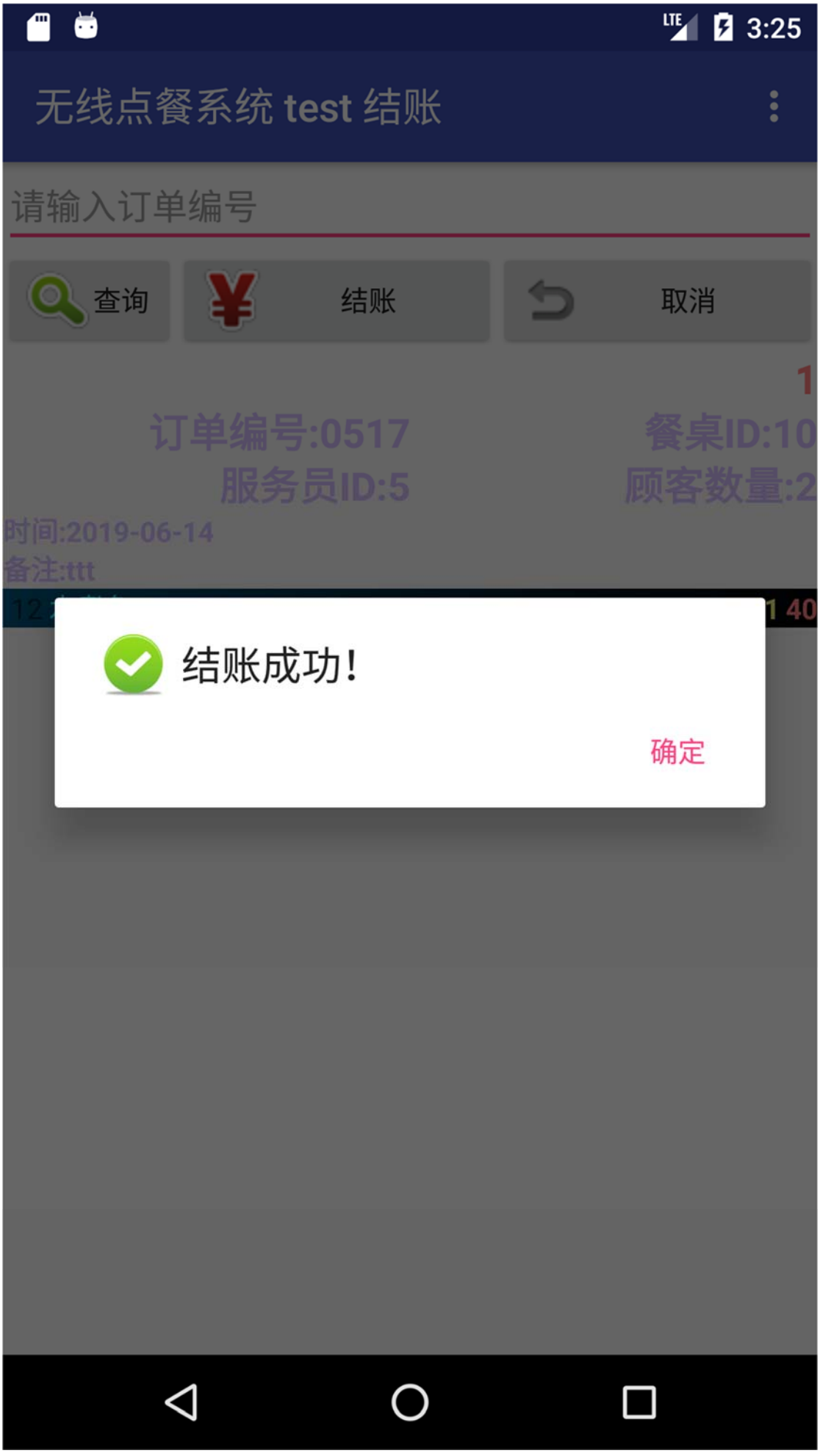
OrderActivity | onCreate()

```
     * 在子线程中执行代码
     * @param r
     */
    ExecutorService executor = Executors.newSingleThreadExecutor();
    Future<String> future = executor.submit((Callable) () → {
            App app = (App) getApplicationContext();
            String url = app.URL + "/order/put";
            boolean flag = false;
            try {
                flag = HttpUtils.sendObject(url, ordered);
            } catch (Exception e) {
                e.printStackTrace();
            }
            return flag == true ? "1" : "0";
    });
    String result = null;
    try {
        result = future.get();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
    }finally {
        executor.shutdown();
    }
    if(result != null && result.equals("1")) {
        showMessageDialog("下单成功！",
                R.drawable.successful, null);
    }else if(result != null && result.equals("0")){
        showMessageDialog("下单失败！", R.drawable.warning, null);
    }
```

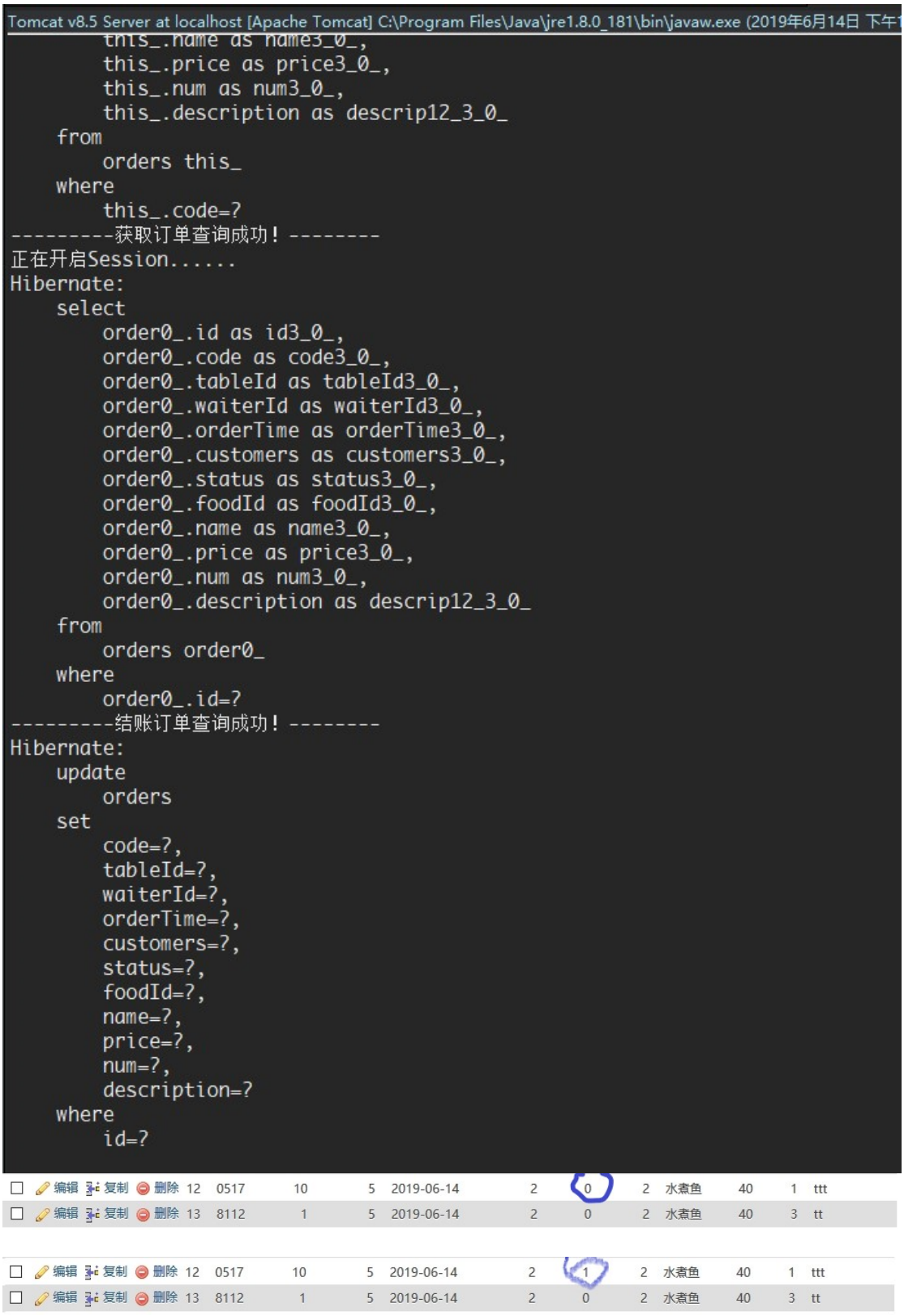
4.结账功能


3:23
无线点餐系统 test 结账
请输入订单编号
查询
结账
取消


3:25
无线点餐系统 test 结账
请输入订单编号
查询
结账
取消
1
订单编号:0517
餐桌ID:10
服务员ID:5
顾客数量:2
时间:2019-06-14
备注:ttt
12 水煮鱼
ttt
1 40


3:25
无线点餐系统 test 结账
请输入订单编号
查询
结账
取消
1
订单编号:0517
餐桌ID:10
服务员ID:5
顾客数量:2
时间:2019-06-14
备注:ttt
结账成功!
确定


3:36
无线点餐系统 test 结账
请输入订单编号
查询
结账
取消
此订单已结算， 合计：1
订单编号:0517
餐桌ID:10
服务员ID:5
顾客数量:2
时间:2019-06-14
备注:ttt
12 水煮鱼
ttt
1 40

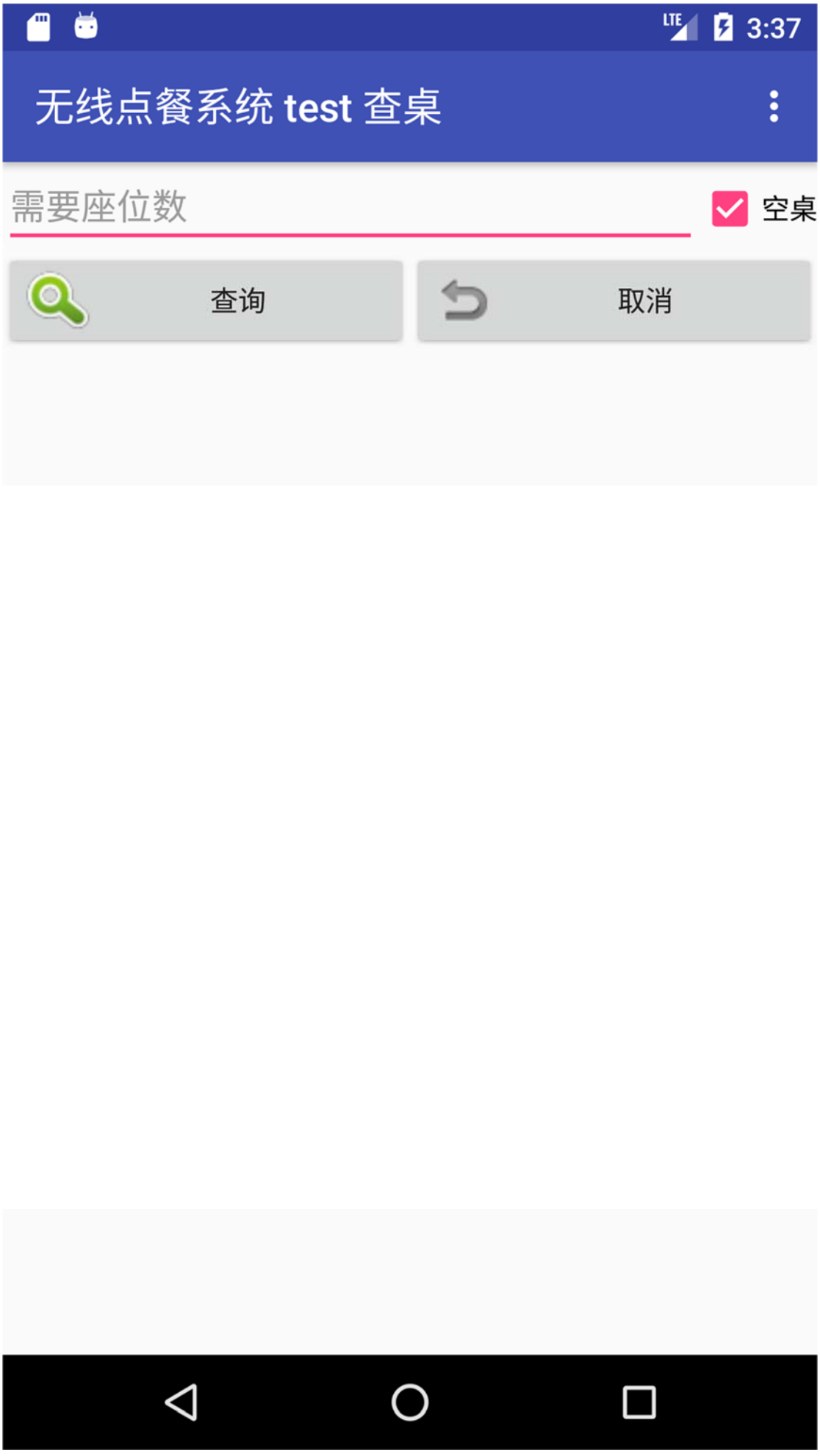


这个线程的用法跟上面差不多，先在服务器中查询订单编号，找到则设置改变 UI 一个属性值作为找到该订单的标志（因为当用户直接点击结账时得检查这个标志，即查到订单才可以结账），同时将该订单信息通过适配器 SimpleAdaptor 显示在屏幕上。

```
public void query() throws InterruptedException {
    String orderCode = orderCodeEdt.getText().toString();
    clearDisplay();
    if(orderCode.length() == 0) {
        showMessageDialog("请输入订单编号", R.drawable.warning, null);
        return;
    }
    Order dto = null;
    final String code = orderCode;
    ExecutorService executor = Executors.newFixedThreadPool(1);
    CompletionService completionService = new ExecutorCompletionService(executor);
    completionService.submit(() -> {
            String url = app.URL + "/order/get?code=" + code;
            String str = HttpUtils.get(url);
            Order order = HttpUtils.parseJSONToOrder(str);
            return order;
    });

    try {
        dto = (Order)completionService.take().get();
    } catch (ExecutionException e) {
        e.printStackTrace();
    }

    if(dto == null) {
        showMessageDialog("未查找到订单!", R.drawable.warning, null);
        return;
    }
    orderCodeTxv.setText("订单编号:" + dto.getCode());
    tableCodeTxv.setText("餐桌ID:" + dto.getTableId() + "");
    waiterCodeTxv.setText("服务员ID:" + dto.getWaiterId() + "");
    orderTimeTxv.setText("时间:" + sdf.format(dto.getOrderTime()));
    customersTxv.setText("顾客数量:" + dto.getCustomers());
    descriptionTxv.setText(dto.getDescription() == null ? "" : ("备注:" + dto.getDescription()));
    Map<String, Object> line = new HashMap<~>();
    line.put("foodId", dto.getFoodId());
    line.put("no", dto.getId());
    line.put("name", dto.getName());
    line.put("description", dto.getDescription());
    line.put("num", dto.getNum());
    line.put("price", dto.getPrice());
    line.put("checked", true);
    orderedList.add(line);
    SimpleAdapter sa = (SimpleAdapter)orderedLtv.getAdapter();
    sa.notifyDataSetChanged();
    orderedLtv.setVisibility(View.VISIBLE);
    if(dto.getStatus() == 1) {
        sumTxv.setText("此订单已结算， 合计：" + dto.getNum());
        PayActivity.this.orderStatus = 1;
    }else {
        sumTxv.setText(" " + dto.getNum());
        PayActivity.this.orderId = dto.getId();
        PayActivity.this.orderStatus = 0;
    }
```

```
public void pay() {
    if(orderId == -1) {
        showMessageDialog("请选择订单", R.drawable.warning, null);
        return;
    }
    if(orderStatus == 1) {
        showMessageDialog("此订单已结算", R.drawable.warning, null);
        return;
    }

    /**
     * 在子线程中执行代码
     * @param r
     */
    ExecutorService executor = Executors.newSingleThreadExecutor();
    Future<String> future = executor.submit((Callable) () -> {
            App app = (App) getApplicationContext();
            String url = app.URL + "/order/pay?orderId=" + orderId;
            int status = 0;
            try {
                String str = HttpUtils.get(url);
                status = HttpUtils.parseMsg(str);
            } catch (Exception e) {
                e.printStackTrace();
            }
            return status == 200 ? "1" : "0";
    });
    String result = null;
    try {
        result = future.get();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
    } finally {
        executor.shutdown();
    }

    if(result.equals("1")) {
        orderId = 1;
        showMessageDialog("结账成功！", R.drawable.successful, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                finish();
            }
        });
    }else {
        showMessageDialog("结账失败！", R.drawable.not, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                finish();
            }
        });
    }
}
```

5.查桌功能（存在问题）



暂时还没有找到方法可以在线程中执行完任务后返回一个集合或对象数据（这里是 List<Table>），可能得重写里面的

一个回调方法，有待研究。

TableActivity > onCreate() > new OnClickListener > onClick() > new Callable > call()

```
67              final int seats = num;
68              String result;
69              if(needEmptyCkb.isChecked()) {
70                  result = "true";
71              }else {
72                  result = "false";
73              }
74              final String needEmpty = result;
75              List<Table> tables = null;
76              try {
77                  ExecutorService executor = Executors.newSingleThreadExecutor();
78                  CompletionService completionService = new ExecutorCompletionService(executor);
79                  completionService.submit(() -> {
81                          String url = app.URL + "/table/query?seats=" + seats + "&needEmpty=" + needEmpty;
82                          Table tb = HttpUtils.getTable(url);
83                          return tb;
84                  });
86
87                  try {
88                      Table tab = (Table) completionService.take().get();
89                      tables.add(tab);
90                  } catch (ExecutionException e) {
91                      e.printStackTrace();
92                  }finally {
93                      executor.shutdown();
94                  }
95              }catch(Exception e) {
96                  e.printStackTrace();
97                  showMessageDialog("未知错误！", R.drawable.warning, null);
98                  return;
99              }
100             tableList.clear();
```