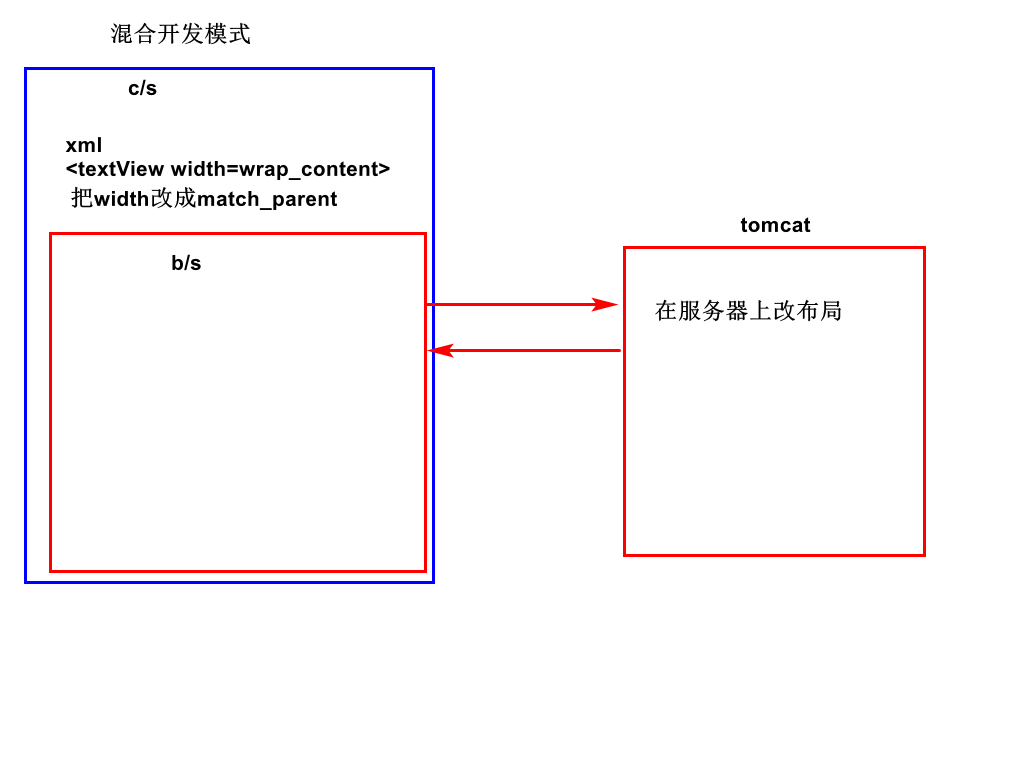
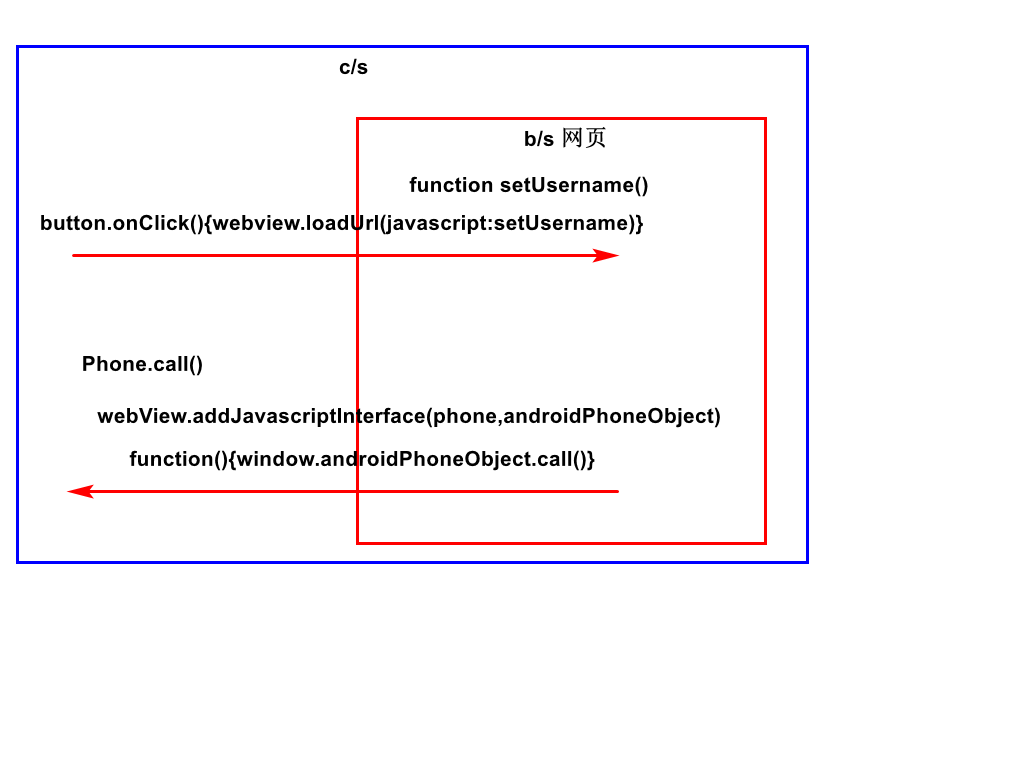
# 复习





# 拼多商城

ServerResult

{

Status

Msg

data

}

权限

全局变量

PDApplication{

serverIP

}

RegisterActivity

registerBtn.onClick(){

找到5个输入框

接收数据

}

## 联网注册用户

### 测试服务器接口

1. 查看源码 pdServer\src\main\java\com\pd\controller\UserController.java

@RequestMapping("/user/register.html")

@ResponseBody

//http://ip/user/register.html

//?username=180501&password=123456&phone=13691481001&email=1@qq.com

**public** PdResult register(PdUser ajiaUser)

**throws** Exception

{

PdResult ajiaResult=**new** PdResult();

ajiaResult.setStatus(500);

String password=ajiaUser.getPassword();

password=DesUtil.*encrypt*(password, "88889999");

password=MD5Encrypt.*MD5Encode*(password);

ajiaUser.setPassword(password);

String username=ajiaUser.getUsername();

PdUser dbPdUser=userService

.selectByUsername(username);

**if** (dbPdUser==**null**)

{

**int** row=userService.insert(ajiaUser);

**if** (row>=1)

{

ajiaResult.setStatus(200);

}

}

**return** ajiaResult;

}

2. 浏览器测试

<http://ip/user/register.html?username=180501&password=123456&phone=13691481001&email=1@qq.com>

第一次访问，服务器返回

{"status":200,"msg":null,"data":null}

再刷新url,服务器返回

{"status":500,"msg":null,"data":null}

3. 查看数据库有没有注册成功

执行下面select查询最新用户

SELECT \* FROM pd\_user ORDER BY id desc

### 联网过程分析，如图-86所示。

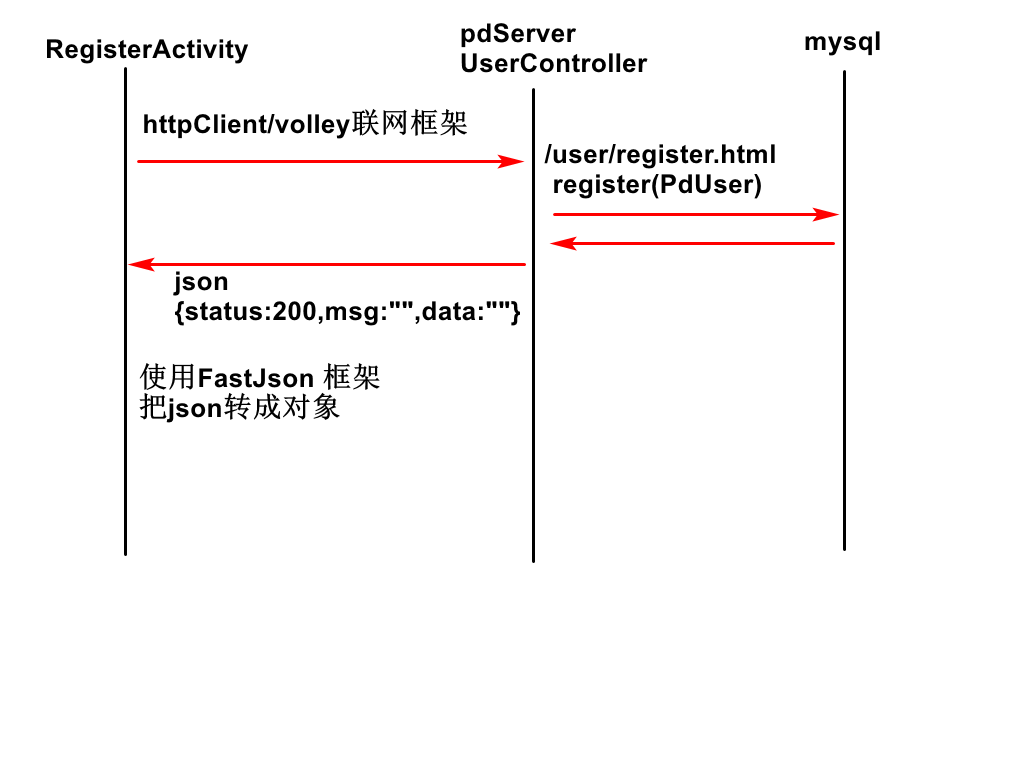


图- 86

Volley框架原理，如图-87所示。

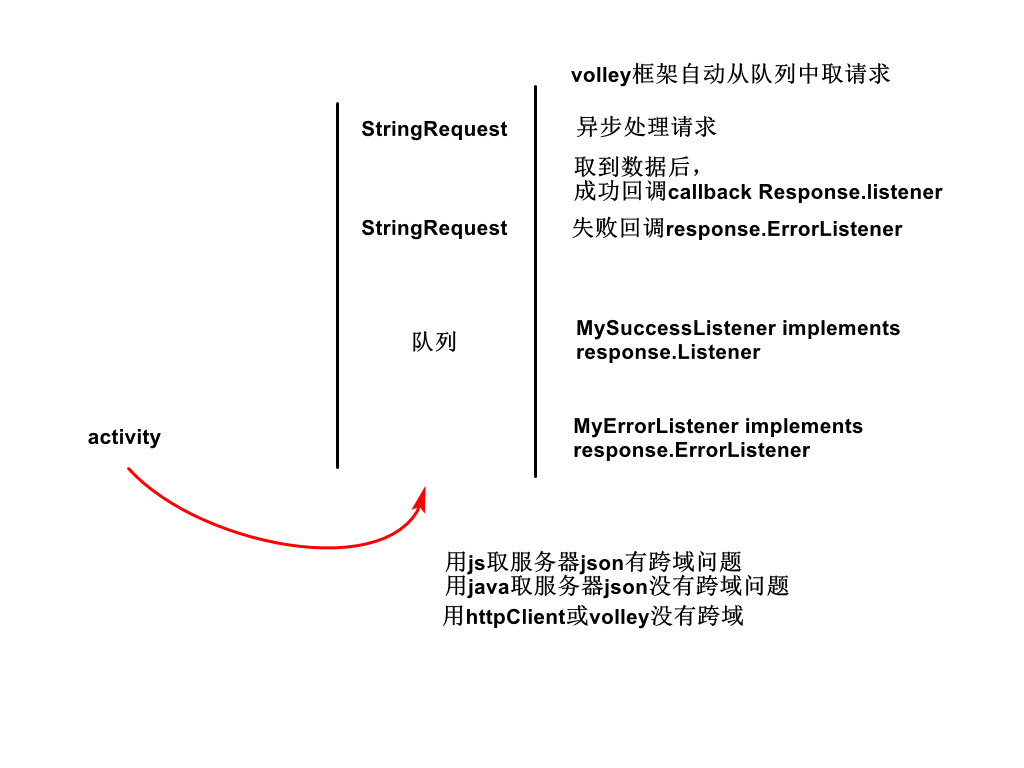


图- 87

### 拷贝jar

从“\亿发,移动端课前资料\android”拷贝两个jar

文件到app/libs文件夹，如图-88所示。

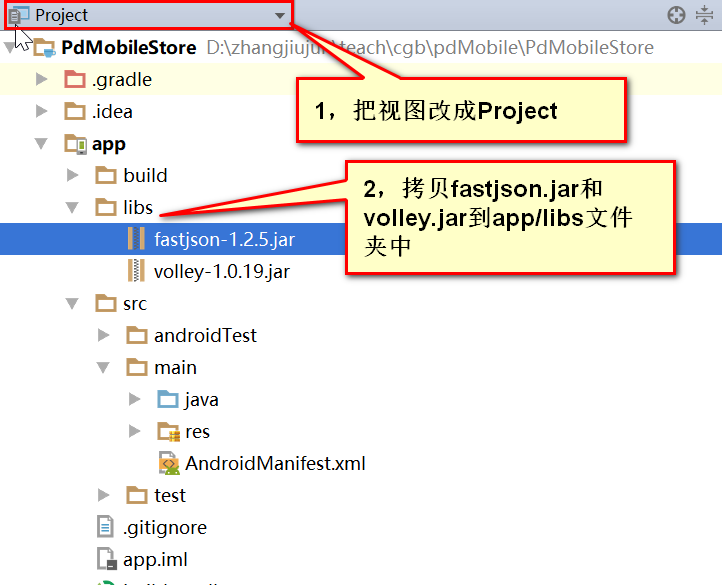


图- 88

把两个jar加到编译路径中有两种方 法

第一种方法，如图-89所示。

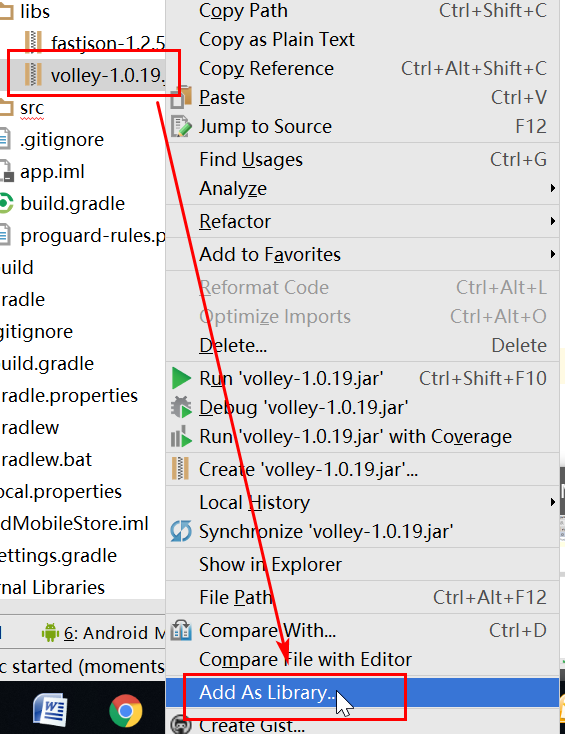


图- 89

在图-90中点击ok

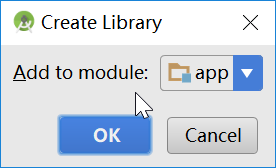


图- 90

第二种方法，如图-91所示。

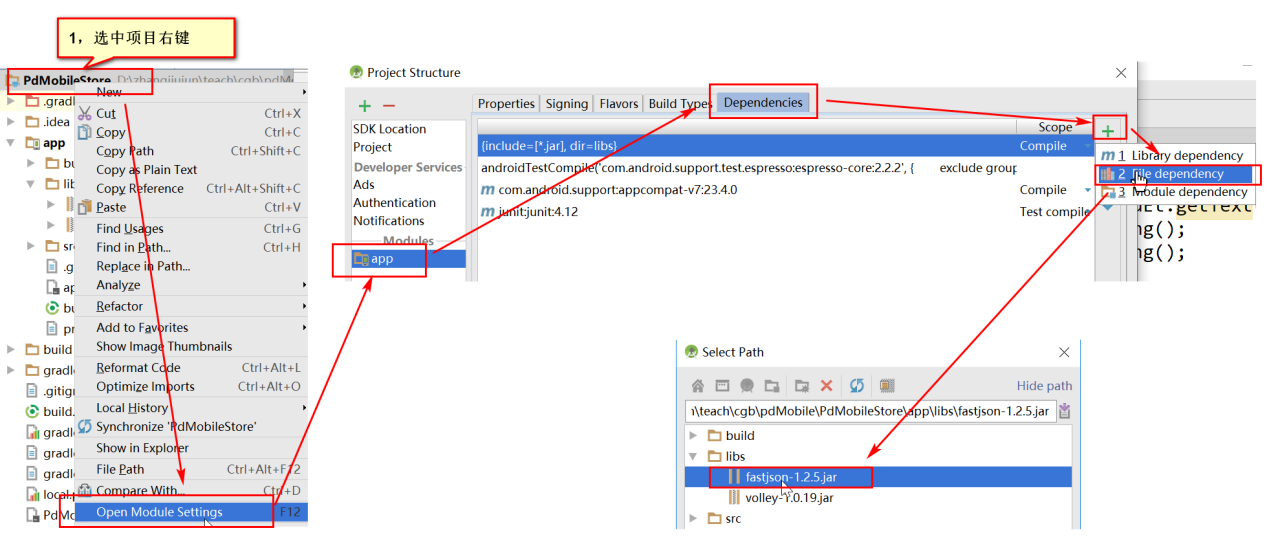


图- 91

### 代码实现

1,创建实体类

*//{"status":200,"msg":null,"data":null}***public class** ServerResult **implements** Serializable {  
 **private int status**;  
 **private** String **msg**;  
 **private** Object **data**;  
  
 **public** String getMsg() {  
 **return msg**;  
 }  
  
 **public void** setMsg(String msg) {  
 **this**.**msg** = msg;  
 }  
  
 **public** Object getData() {  
 **return data**;  
 }  
  
 **public void** setData(Object data) {  
 **this**.**data** = data;  
 }  
  
 **public int** getStatus() {  
 **return status**;  
 }  
  
 **public void** setStatus(**int** status) {  
 **this**.**status** = status;  
 }  
}

定义常量

public class PdApplication {  
 final static String *serverIP*="http://192.168.221.194";  
}

在androidManifest.xml中添加联网权限

<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.pd.pdmobile">

*<!--权限-->* <uses-permission android:name="android.permission.INTERNET">  
  
 </uses-permission>

<application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme">  
 <activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <activity android:name=".LoginActivity" />  
 <activity android:name=".RegisterActivity"></activity>  
 </application>  
  
</manifest>

2．联网发数据

**public class** Register **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_register***);  
 Button registerBtn= (Button) findViewById(R.id.***register\_register***);  
 registerBtn.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//1,找到5个输入框* EditText usernameEt= (EditText)  
 findViewById  
 (R.id.***register\_username***);  
  
 EditText passwordEt= (EditText)  
 findViewById  
 (R.id.***register\_password***);  
  
 EditText confirmPasswordEt= (EditText)  
 findViewById  
 (R.id.***register\_confirmPassword***);  
  
 EditText emailEt= (EditText)  
 findViewById  
 (R.id.***register\_email***);  
  
 EditText phoneEt= (EditText)  
 findViewById  
 (R.id.***register\_phone***);  
  
 *//2,从输入框得用户输入的内容* String username=usernameEt  
 .getText().toString();  
 String password=passwordEt  
 .getText().toString();  
 String confirmPassword=confirmPasswordEt  
 .getText().toString();  
 String email=emailEt.getText().toString();  
 String phone=phoneEt.getText().toString();  
  
 *//3,数据检查* **if** (password.equals(confirmPassword)){  
  
 *//4,用volley联网把数据发给服务器  
 //4.1 拼出url* String url=PDApplication.***SERVER\_IP***+**"/user/register.html"**;  
 *//4.2 准备发送的数据* url=url+**"?username="**+username  
 +**"&password="**+password  
 +**"&phone="**+phone  
 +**"&email="**+email;  
 *//4.3 发送：让请求放到队列* MySuccessListener mySuccessListener=**new** MySuccessListener();  
 MyErrorListener myErrorListener=**new** MyErrorListener();  
 StringRequest stringRequest=**new** StringRequest(url,mySuccessListener,myErrorListener);  
 *//创建队列  
 //使用队列的优点是可以处理很多请求  
 //不用队列，框架正在处理一个请求，  
 // 新的请求不能处理，* RequestQueue requestQueue=  
 Volley.*newRequestQueue*(Register.**this**);  
  
 requestQueue.add(stringRequest);  
 *//请求放到队列后，  
 //1,volley自动从队列中取出请求  
 //2,联网发送请求  
 //3,如果成功，volley框架调mySuccessListener  
 //4,如果失败，volley框架调myErrorListener* }  
 }  
 });  
  
 }  
 *//内部类  
 //联网成功后，回调* **class** MySuccessListener **implements** Response.Listener<String>{  
  
 @Override  
 **public void** onResponse(String s) {  
 *//4.4 接收服务器返回的数据  
 //4.5 显示结果* ServerResult serverResult=  
 JSON.*parseObject* (s,ServerResult.**class**);  
 String msg=**"注册失败"**;  
 **if** (serverResult.getStatus()==200)  
 {  
 msg=**"注册成功"**;  
 }  
 Toast.*makeText*(Register.**this**,  
 msg,Toast.***LENGTH\_LONG***)  
 .show();  
 }  
 }  
 *//联网失败后，回调* **class** MyErrorListener **implements** Response.ErrorListener{  
  
 @Override  
 **public void** onErrorResponse(VolleyError volleyError) {  
 *//把错误信息发给服务器，服务器收到后，保存到mysql* Toast.*makeText*(Register.**this**,volleyError.getMessage(),Toast.***LENGTH\_LONG***).show();  
 }  
 }  
}

## 出错后

创建新项目Pdapp 公司域名tedu.com

包名是com.tedu.pdapp

拷贝两个jar，执行add as library

删除app/src/main文件夹

拷贝main

改PdApplicaton中的serverIp

运行程序

## 联网登录

### 分析，如图-92所示。

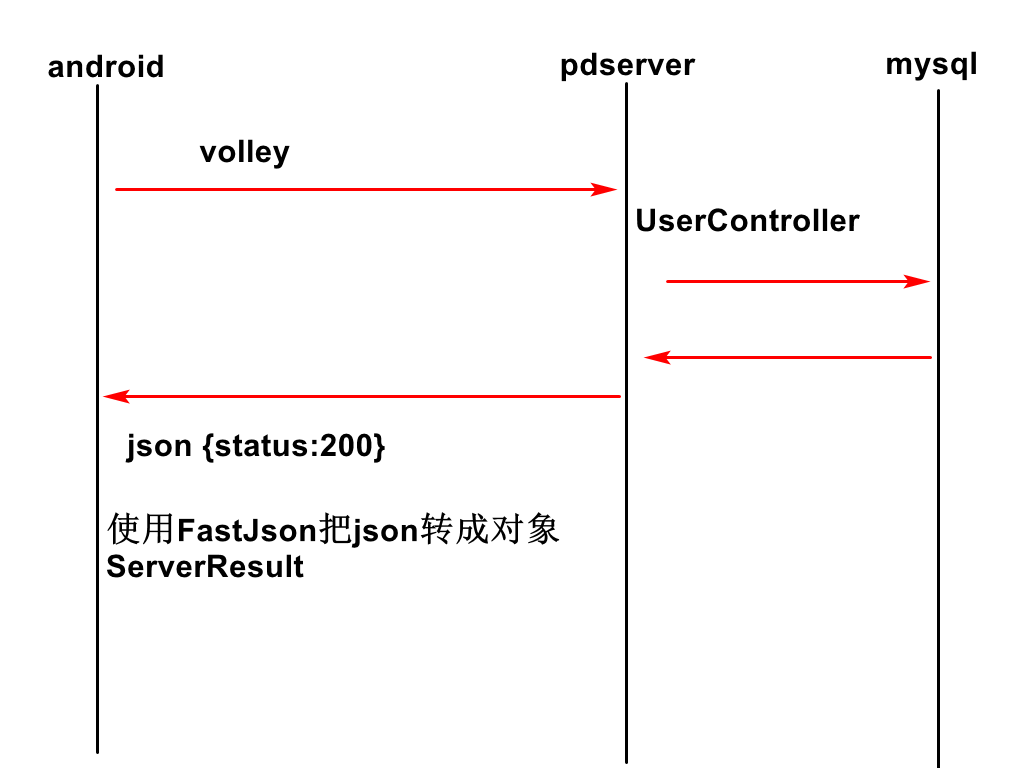


图- 92

### 测试服务器接口

1. 查看源码 pdServer\src\main\java\com\pd\controller\UserController.java

//http://localhost:8090/user/login.html?username=b&password=123457

@RequestMapping("/user/login.html")

@ResponseBody

**public** PdResult Login

(String username,String password) **throws** Exception

{

password=DesUtil.*encrypt*(password, "88889999");

password=MD5Encrypt.*MD5Encode*(password);

PdResult ajiaResult=**new** PdResult();

ajiaResult.setStatus(500);

PdUser dbPdUser=userService

.selectByUsername(username);

**if** (dbPdUser!=**null** &&

password.equals(dbPdUser.getPassword()))

{

ajiaResult.setStatus(200);

}

**return** ajiaResult;

}

2. 浏览器测试

http://192.168.221.194/user/login.html?username=180601&password=123456

### 实现

**LoginActivity.oncreate()中增加如下代码**

**public class** LoginActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_login***);  
 *//有个注册按钮  
 //找到注册按钮* Button registerBtn= (Button)  
 findViewById(R.id.***login\_register***);  
 *//设置事件监听* registerBtn.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//跳到register* Intent intent=**new** Intent(LoginActivity.**this**,Register.**class**);  
 startActivity(intent);  
 }  
 });  
 *//找到登录按钮* Button loginBtn= (Button) findViewById  
 (R.id.***login\_login***);  
 *//设置事件监听* loginBtn.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//1,找到输入框* EditText usernameEt= (EditText)  
 findViewById(R.id.***login\_username***);  
 EditText passwordEt= (EditText) findViewById  
 (R.id.***login\_password***);  
 *//2,接收用户输入的内容* String username=usernameEt.getText().toString();  
 String password=passwordEt.getText().toString();  
 *//3,判断数据* **if** (!TextUtils.*isEmpty*(username)  
 && !TextUtils.*isEmpty* (password)){  
 *//4,把数据发给服务器的controller  
 //4.1 拼出url* String url=PDApplication.***SERVER\_IP*** +**"/user/login.html"**;  
 *//4.2 准备发送的数据* url=url  
 +**"?username="**+username  
 +**"&password="**+password;  
 *//4.3 发送* MySuccessListener mySuccessListener=**new** MySuccessListener();  
 MyErrorListener myErrorListener=**new** MyErrorListener();  
 StringRequest stringRequest=  
 **new** StringRequest(url,mySuccessListener,myErrorListener);  
 RequestQueue requestQueue=  
 Volley.*newRequestQueue*(LoginActivity.**this**);  
 requestQueue.add(stringRequest);  
 *//voller框架会完成下面的事情  
 //1,从队列中取出请求  
 //2,联网发送数据  
 //3,如果联网成功，调Listener的实现类  
 //4,如果联网失败，调errorListener的实现类  
  
 //4.4 接收数据  
 //4.5 显示结果* }  
 }  
 });  
 }  
 *//内部类  
 //联网成功后回调* **class** MySuccessListener  
 **implements** Response.Listener<String>  
 {  
 @Override  
 **public void** onResponse(String s) {  
 *//用fastJson把json转成对象* ServerResult serverResult=  
 JSON.*parseObject* (s,ServerResult.**class**);  
 String msg=**"登录失败"**;  
 **if** (serverResult.getStatus()==200)  
 {  
 msg=**"登录成功"**;  
 }  
 Toast.*makeText*(LoginActivity.**this**,  
 msg,Toast.***LENGTH\_LONG***)  
 .show();  
  
 }  
 }  
 *//联网失败(服务器没开，url写错了)后回调* **class** MyErrorListener **implements** Response.ErrorListener{  
  
 @Override  
 **public void** onErrorResponse(VolleyError volleyError) {  
 System.***out***.println(**"联网出错"**);  
 }  
 }  
}

## 显示主界面

### 分析，如图-93所示。

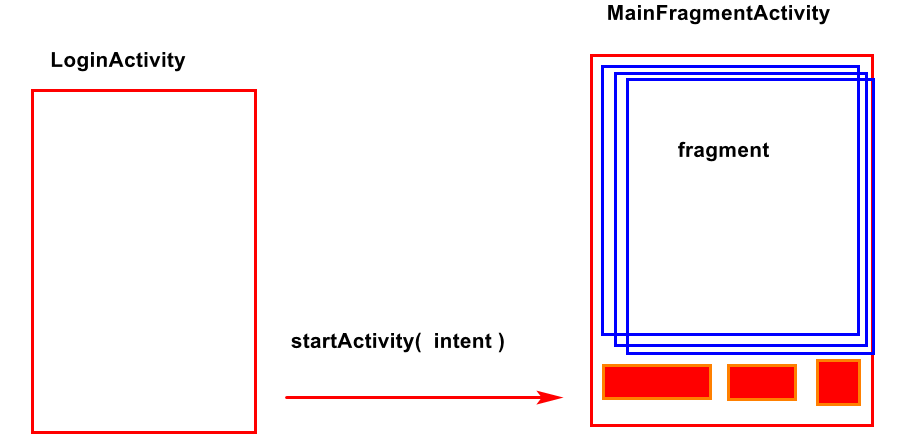




图- 93

### 显示三个按钮

创建java Class，类名为 MainFragmentActivity

加载main\_framgent,只出button

**public class** MainFragmentActivity  
 **extends** FragmentActivity{  
 *//只有一个参数* @Override  
 **protected void** onCreate  
 (@Nullable Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***main\_fragment***);  
 }  
}

清单文件注册,activity标签要放在application标签里面。

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.pd.pdmobile">  
*<!--权限-->* <uses-permission android:name="android.permission.INTERNET">  
  
 </uses-permission>  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme">  
 <activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <activity android:name=".LoginActivity" />  
 <activity android:name=".RegisterActivity">  
 </activity>

<activity android:name=".MainFragmentActivity"/>

</application>  
  
</manifest>

### 登录成功跳到主界面

**if** (result.getStatus() == 200) {  
 Toast.*makeText*(LoginActivity.**this**, **"登录成功"**, Toast.***LENGTH\_LONG***).show();  
 **try** {*//保存用户*   
 Intent intent = **new** Intent(LoginActivity.**this**, MainFragmentActivity.**class**);  
 startActivity(intent);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
  
}

## 数据保存

### sqlite分析，如图-94所示。



图- 94

数据保存过程，如图-95所示。

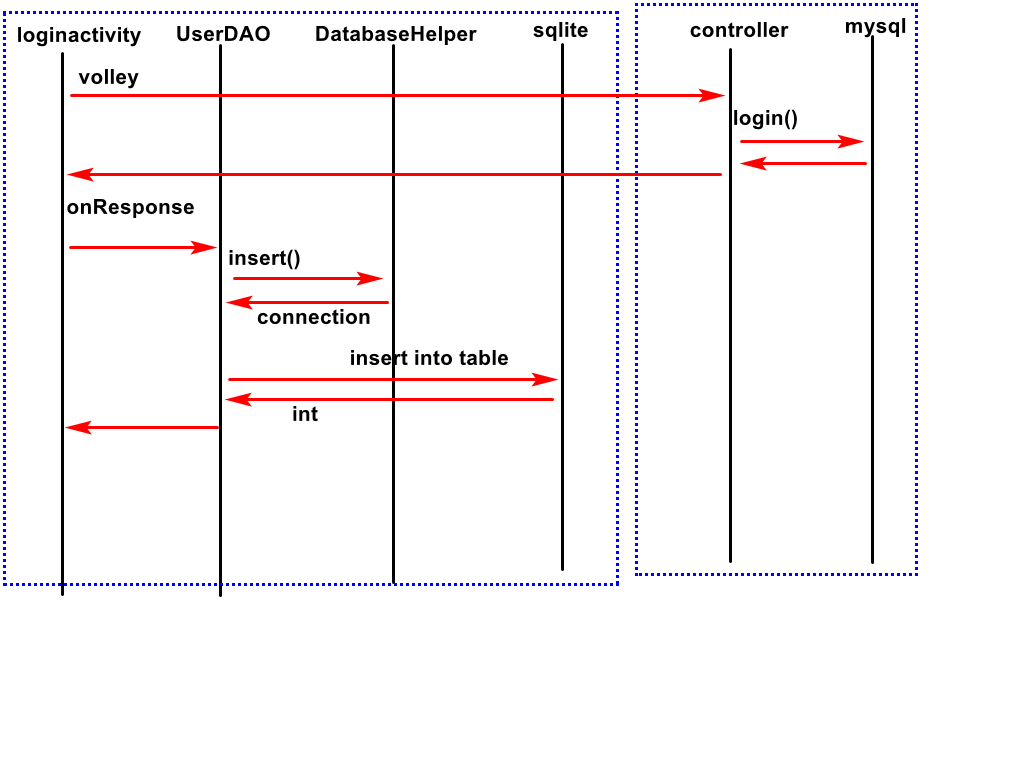


图- 95

### 分析

DatabaseHelper extends SqliteOpenHelper

建立连接

执行sql创建表

UserDAO

{

//相当于context上下文，场景，环境

Insert(activity,username)

{

contentValues.put(username,181012)

//相当于sqlsession

SqliteDatabase.insert(contentValue)

}

}

LoginActivity.onResponse{

调用dao.insert()

}

### 保存数据

#### databaseHelper

**public class** DatabaseHelper **extends** SQLiteOpenHelper {  
  
 **public** DatabaseHelper(Context context) {  
 //代码出错，每次修改后，改一下库名。

**super**(context, **"pd.db"**, **null**, 1);  
 }  
  
 @Override  
 **public void** onCreate(SQLiteDatabase db) {  
 String sql = **"create table pd\_user ("** + **"username varchar(50)) "**;  
  
 db.execSQL(sql);  
 }  
  
 @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
  
 }  
}

#### UserDAO

*//操作user表***public class** UserDAO {  
 **public long** insert  
 (Activity activity,  
 String username)  
 {  
 **long** row=0;  
 *//1，得到连接* DatabaseHelper databaseHelper=  
 **new** DatabaseHelper(activity);  
 *//执行sql语句* SQLiteDatabase sqLiteDatabase=  
 databaseHelper  
 .getWritableDatabase();  
  
  
 *//contentValues是key,value  
 //user 有两列username,password  
 //contentValues.put(username,1)  
 //contexntValues.put(password,a)  
 //2,创建contentValues* ContentValues contentValues=**new** ContentValues();  
 contentValues.put(**"username"**,username);  
 *//3,往数据库写数据* row=sqLiteDatabase.insert(**"user"**,**null** ,contentValues);  
 *//4,关闭连接* sqLiteDatabase.close();  
 databaseHelper.close();  
 **return** row;  
 }  
}

#### LoginActivity

**if** (result.getStatus() == 200) {  
 Toast.*makeText*(LoginActivity.**this**, **"登录成功"**, Toast.***LENGTH\_LONG***).show();  
 **try** {*//保存用户* UserDAO userDAO = **new** UserDAO();  
 userDAO.insert(LoginActivity.**this**, username);  
 Intent intent = **new** Intent(LoginActivity.**this**, MainFragmentActivity.**class**);  
 startActivity(intent);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
  
}

#### 查看数据

手机系统中 \data\data\com.tedu.pdapp\databases\pd1.db

使用夜神模拟器执行D:\Program Files\Nox\bin>文件夹下的adb

在 dos中切换到文件夹中执行下面的命令

adb shell

root@android:/ # ls

data

etc

root@android:/ # cd data

root@android:/data # ls

data

root@android:/data # cd data

root@android:/data/data # ls

com.pd.pdmobilestore(程序的包名)

root@android:/data/data # cd com.pd.pdmobilestore（改成自己程序的包名）

root@android:/data/data/com.pd.pdmobilestore # ls

app\_webview

cache

databases

lib

root@android:/data/data/com.pd.pdmobilestore # cd databases

root@android:/data/data/com.pd.pdmobilestore/databases # ls

pd.db

pd.db-journal

root@android:/data/data/com.pd.pdmobilestore/databases # sqlite3 pd.db

SQLite version 3.7.11 2012-03-20 11:35:50

Enter ".help" for instructions

Enter SQL statements terminated with a ";"

sqlite> .table

android\_metadata pd\_user

sqlite> select \* from pd\_user;

180611

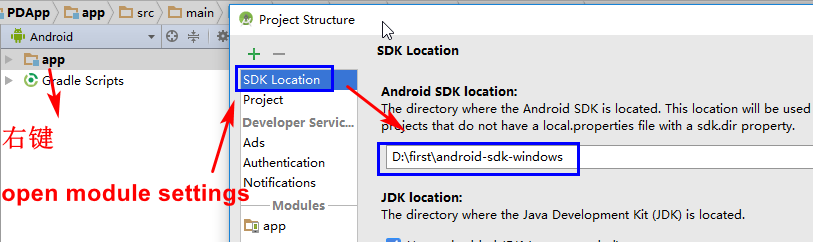
sqlite> .quit

root@android:/data/data/com.pd.pdmobilestore/databases # exit

D:\Program Files\Nox\bin>

#### 使用原生模拟器执行adb shell

查看android-sdk-windows



在dos下进入android-sdk-windows\platform-tools文件夹

执行adb shell

### 查询数据，如图-96所示。

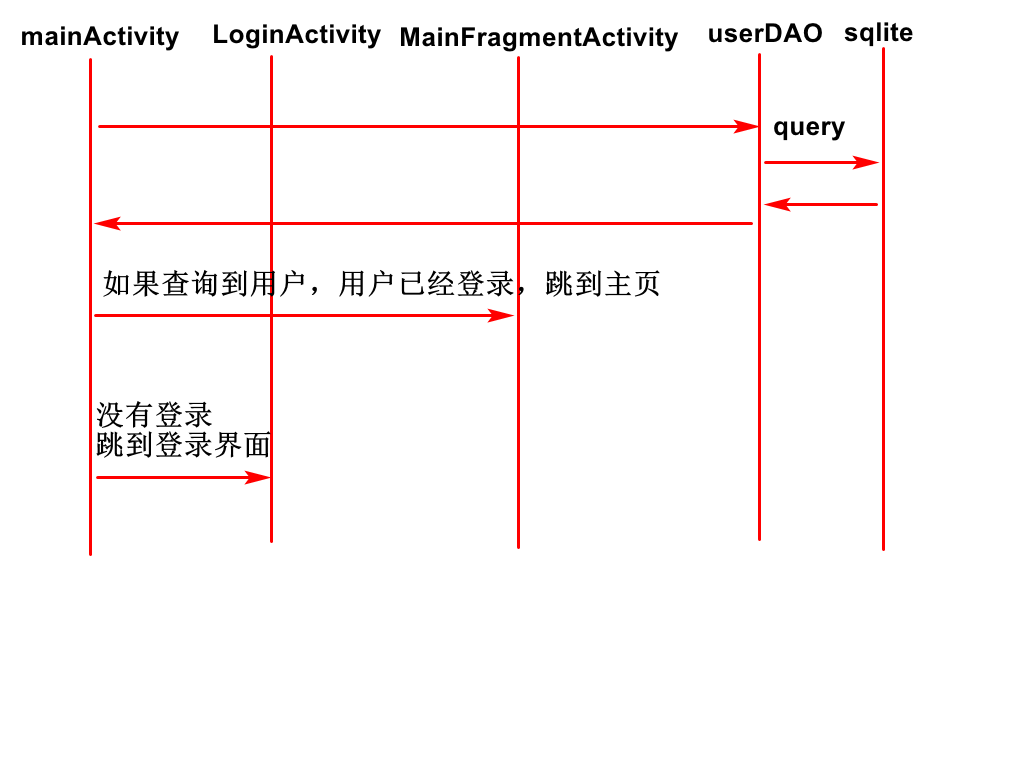


图- 96

1. userDao增加query

*//用户的个数***public int** query(Activity activity)  
{  
 **int** row=0;  
 *//1,得到连接* DatabaseHelper  
 databaseHelper=**new** DatabaseHelper(activity);  
 SQLiteDatabase db=databaseHelper  
 .getReadableDatabase();  
 *//2,执行select语句* String sql=**"select \* from user"**;  
 *//cursor游标，相当于jdbc中的结果集resultSet  
 //jdbc selct \* from user  
 // where username=? and password=?  
 //[a,1]* String[] args=**null**;  
 Cursor cursor=db.rawQuery(sql,args);  
 **if** (cursor!=**null**)  
 {  
 *//3,得到行数* row=cursor.getCount();*//数据的行数* }  
 *//4,关闭连接* db.close();  
 databaseHelper.close();  
 **return** row;  
}

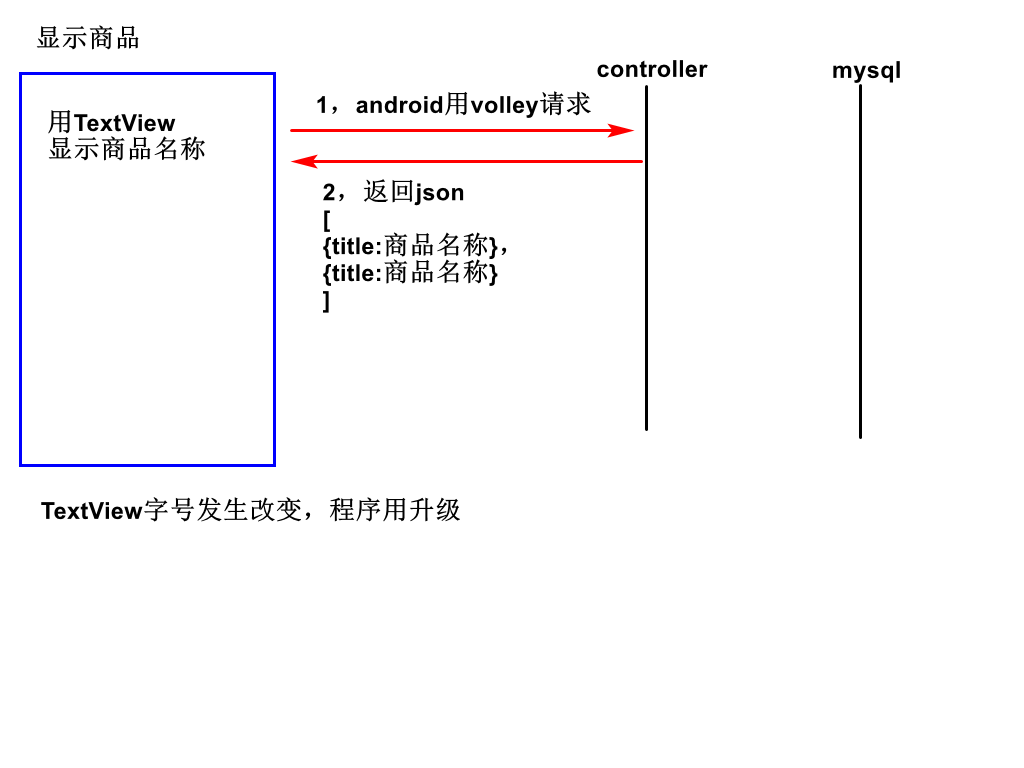
1. MainActivity修改代码

**public void** run() {  
 **super**.run();  
 **try** {  
 sleep(3000);  
 //跳到LoginActivity  
 //p1:从那  
 //p2:去那  
 //查询数据库，看用户有没有登录  
 //登录后，直接跳到mainFragmentActivty  
 //没登录跳到login  
 Class toActivity=LoginActivity.**class**;  
 UserDAO userDAO=**new** UserDAO();  
 **int** row=userDAO.query(MainActivity.**this**);  
 **if** (row>=1)  
 {  
 toActivity=MainFragmentActivity.**class**;  
 }  
 Intent intent=**new** Intent(MainActivity.**this** ,toActivity);  
 startActivity(intent);  
 } **catch** (Exception e) {  
 }  
}

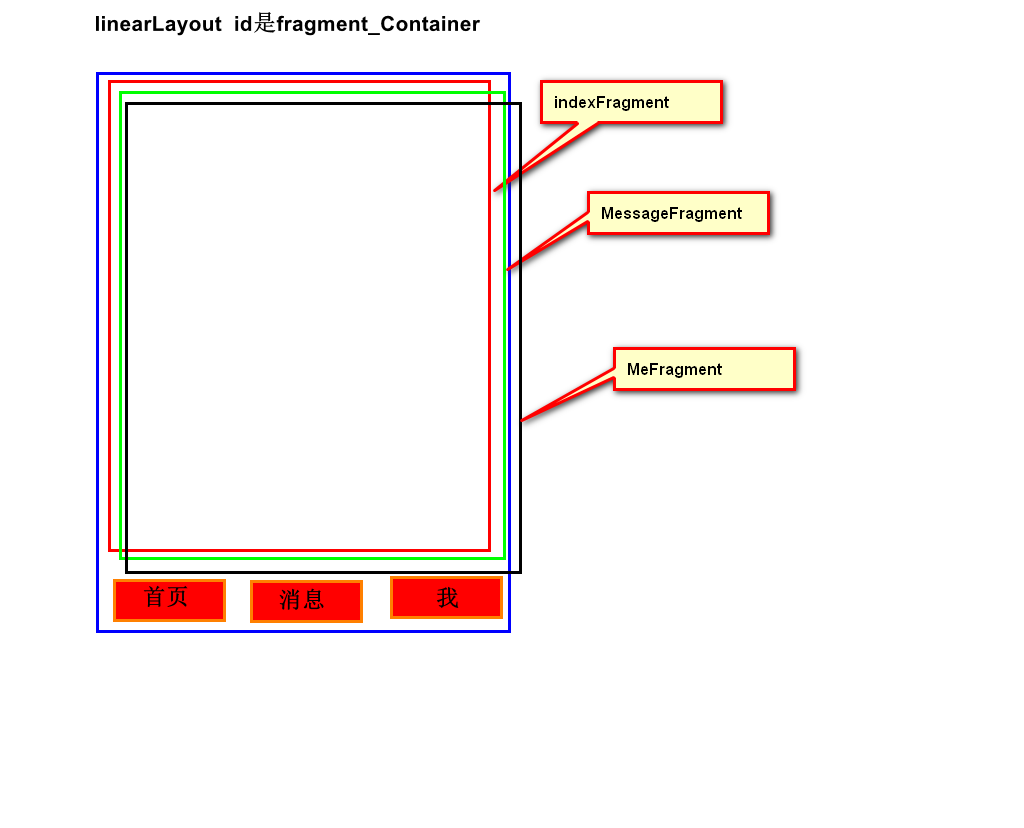
## 显示商品

### 显示商品有两种方式。

1，纯c/s



Fragment介绍



2,c/s+b/s

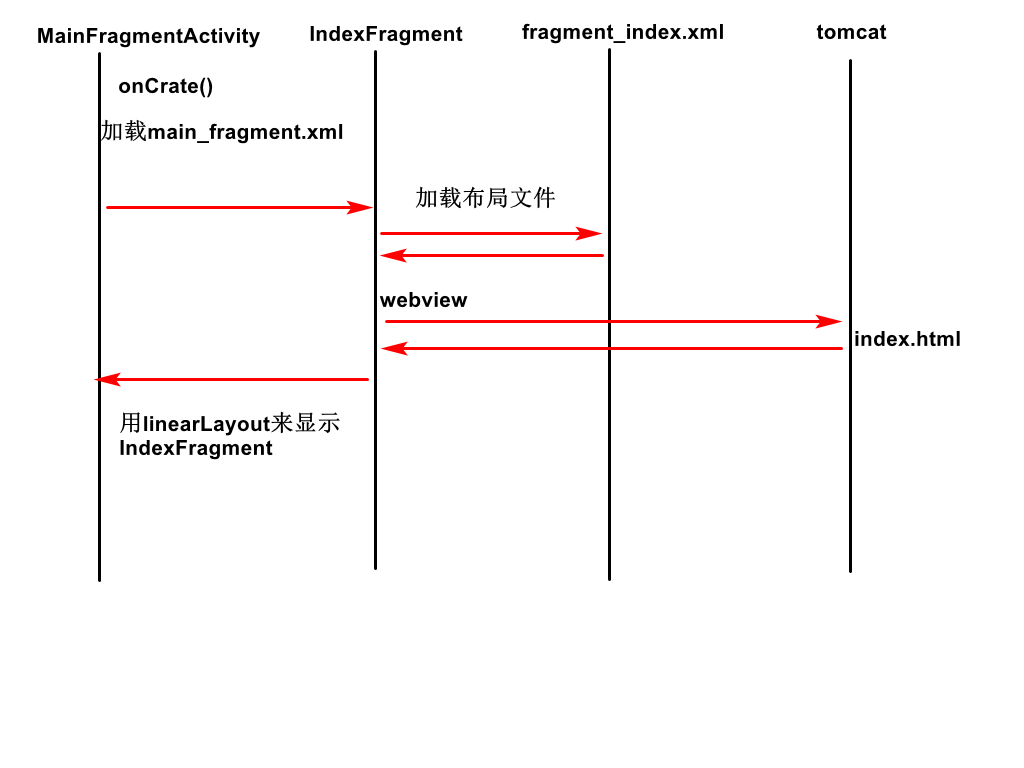


图- 97

mainFragmentActivity加载的布局与indexfragment关系，如图-98所示。

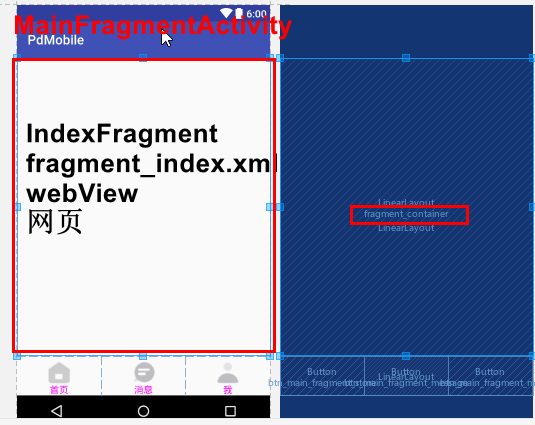


图- 98

### 显示首页

创建IndexFragment

*//fragment类是v4包***public class** IndexFragment **extends** Fragment {  
 @Nullable  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 *//把fragment\_index.xml转成view  
 /\*getActiviyt()返回的是mainFragmentActivity\*/* View view=View.*inflate*(getActivity(),R.layout.***fragment\_index***,**null**);  
 *//找到webView* WebView webView= (WebView) view.findViewById(R.id.***webView***);  
 *//单击超连接，打开的是手机的浏览器,不能打开浏览器* webView.setWebViewClient(**new** WebViewClient());  
 *//用webView加载网页* String url=PdApplication.***serverIP***+**"/mobileIndex.html"**;  
 webView.loadUrl(url);  
 WebSettings webSettings=webView.getSettings();  
 *//允许webview执行js* webSettings.setJavaScriptEnabled(**true**);  
 *//返回view* **return** view;  
 }  
}

MainFragmentActivity中加载IndexFragment

IndexFragment **indexFragment**;*//商城首页  
//只有一个参数*@Override  
**protected void** onCreate(@Nullable Bundle  
 savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *//加载res/layout/main\_fragment.xml* setContentView(R.layout.***main\_fragment***);  
 *//main\_fragment.xml中有一个linearLayout,  
 // id是fragment\_container  
 //显示fragment用的是事务  
 //开始事务* FragmentManager manager=  
 getSupportFragmentManager();  
 FragmentTransaction transaction=manager  
 .beginTransaction();  
 *//动作1：添加indexFragment* **indexFragment**=**new** IndexFragment();  
 **int** containerId=R.id.***fragment\_container***;  
 *//使用linearLayout来显示indexFragment* transaction.add(containerId,**indexFragment**);  
 *//动作2:显示indexFragment* transaction.show(**indexFragment**);  
 *//提交事务  
 //前面有两个动作，如果有一个失败，就不提交了* transaction.commit();  
  
}