# Day02 短信验证 订单推送

**短信验证**

1.支持ARR包

repositories{  
 flatDir{  
 dirs 'libs'  
 }  
}

2.添加依赖

compile files('libs/MobCommons-2016.0714.1402.jar')  
compile files('libs/MobTools-2016.0714.1402.jar')  
compile name: 'SMSSDK-2.1.1', ext: 'aar'

3.初始化

SMSSDK.initSDK(this, "1cfd8fdbf1d00", "f35256dbe2e17fa453af8738b3d6534f");

SMSSDK.registerEventHandler(mEventHandler); //注册短信回调

4.获取和提交验证码

SMSSDK.getVerificationCode("86", mPhone);

SMSSDK.submitVerificationCode("86", mPhone, code);

5.获取和提交后的回调

EventHandler mEventHandler = new EventHandler(){  
  
 @Override  
 public void afterEvent(int event, int result, Object data) {  
  
 if (result == SMSSDK.RESULT\_COMPLETE) {  
 //回调完成  
 if (event == SMSSDK.EVENT\_SUBMIT\_VERIFICATION\_CODE) {  
 //2.提交验证码成功  
 Log.e("sms","提交验证码成功");  
 }else if (event == SMSSDK.EVENT\_GET\_VERIFICATION\_CODE){  
 //1.获取验证码成功  
 Log.e("sms","获取验证码成功");  
 }else if (event ==SMSSDK.EVENT\_GET\_SUPPORTED\_COUNTRIES){  
 //返回支持发送验证码的国家列表  
  
 }  
 }else{  
 ((Throwable)data).printStackTrace();  
 }  
 }  
 };

6.增加倒计时

private class CutdownTask implements Runnable{  
 @Override  
 public void run() {  
 for(;time>0;time--){  
 try {  
 Thread.sleep(999);  
 mHandler.sendEmptyMessage(TIME\_MINUS);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 mHandler.sendEmptyMessage(TIME\_IS\_OUT);  
 }  
}

**登录外卖服务器**

1.在TakeoutService里添加接口

@GET("login")  
Call<ResponseInfo> loginByPhone(@Query("phone") String phone, @Query("type")int type);

2.创建LoginActivityPresenter调用接口

public class LoginActivityPresenter extends NetPresenter{  
  
 private LoginActivity mLoginActivity;  
  
 public LoginActivityPresenter(LoginActivity loginActivity) {  
 this.mLoginActivity = loginActivity;  
 }  
  
 public void loginByPhone(String phone, int type){  
 Call<ResponseInfo> loginCall = mTakeoutService.loginByPhone(phone,type);  
 loginCall.enqueue(mCallback);  
 }

3.缓存到内存中

@Override  
protected void onSuccess(String json) {  
 boolean isLoginOk = false;  
 Gson gson = new Gson();  
 User user = gson.fromJson(json, User.class);  
 //1.内存缓存  
 **TakeoutApp.sUser = user;**

4.登录成功后回调

public void afterLogin(boolean isLoginOk) {  
 if(isLoginOk){  
 //关闭页面  
 finish();  
 }else{  
 Toast.makeText(this, "请仔细检查验证码", Toast.LENGTH\_SHORT).show();  
 }  
}

@Override  
public void onStart() {  
 super.onStart();  
 //登录成功后更新Ui  
 int userId = TakeoutApp.sUser.getId();  
 if(userId!=-1){  
 mLogin.setVisibility(View.GONE);  
 mLlUserinfo.setVisibility(View.VISIBLE);  
 //展示用户名和电话号码  
 mUsername.setText("欢迎你，" + TakeoutApp.sUser.getName());  
 mPhone.setText(TakeoutApp.sUser.getPhone());  
 }else{  
 mLogin.setVisibility(View.VISIBLE);  
 mLlUserinfo.setVisibility(View.GONE);  
 }  
}

**OrmLite的使用**

1.Orm是JavaBean和Sqlite表的映射

@DatabaseTable(tableName = "t\_user")  
public class User {  
 @DatabaseField(id = true)  
 private int id; //auto increment自动增长(generatedId = true)，第二种可以任意指定id = true  
 @DatabaseField(columnName = "name")  
 private String name;  
 @DatabaseField(columnName = "balance")  
 private float balance;  
 @DatabaseField(columnName = "discount")  
 private int discount;  
 @DatabaseField(columnName = "integral")  
 private int integral;  
 @DatabaseField(columnName = "phone")  
 private String phone;

2.OrmLite是对SqliteOpenHelper的封装

public class TakeoutOpenHelper extends OrmLiteSqliteOpenHelper {  
  
 public TakeoutOpenHelper(Context context) {  
 super(context, "takeout36.db", null, 1);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase database, ConnectionSource connectionSource) {  
 //创建数据库的时候需要创建表  
 try {  
 TableUtils.createTable(connectionSource, User.class);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }

3.获取dao增删改查

Dao<User,Integer> userDao = takeoutOpenHelper.getDao(User.class);  
//步骤B:更新或者创建  
User oldUser = userDao.queryForId(36);  
if(oldUser!=null){  
 //已经有36这个用户，更新用户信息（老用户）  
 userDao.update(user);  
 //TecentStasticSDK.userAction(ACTION\_LOGIN, -1);  
 Log.e("login", "老用户登录");  
}else{  
 //新建一个用户,新用户登录  
 userDao.create(user);  
 //TecentStasticSDK.userAction(ACTION\_LOGIN, 0);  
 Log.e("login", "新用户登录");  
}

4.使用事务

AndroidDatabaseConnection connection = new AndroidDatabaseConnection(db, true);  
//保存点,操纵数据库的事务  
Savepoint startPoint = null;  
try {  
 startPoint = connection.setSavePoint("start");  
 connection.setAutoCommit(false); //手动提交事务，注意顺序

...  
 connection.commit(startPoint);  
 isLoginOk = true;  
} catch (SQLException e) {  
 e.printStackTrace();  
 isLoginOk = false;  
 Log.e("login", "保存用户缓存信息失败");  
 try {  
 connection.rollback(startPoint);  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 }  
}

**订单的快速实现**

1.在TakeoutService里添加接口

@GET("order")  
Call<ResponseInfo> getOrderList(@Query("userId") String userId);

2.创建LoginActivityPresenter调用接口

public class OrderFragmentPresenter extends NetPresenter{  
  
 OrderFragment mOrderFragment;  
  
 public OrderFragmentPresenter(OrderFragment orderFragment) {  
 mOrderFragment = orderFragment;  
 }  
  
 public void getOrderList(String userId){  
 Call<ResponseInfo> orderCall = mTakeoutService.getOrderList(userId);  
 orderCall.enqueue(mCallback);  
 }

3.解析数据

@Override  
protected void onSuccess(String json) {  
 //List<Order>  
 Gson gson = new Gson();  
 List<Order> orderList = gson.fromJson(json, new TypeToken<List<Order>>(){}.getType());  
 mOrderFragment.onOrderSuccess(orderList);  
}

4.获取订单成功后回调

public void onOrderSuccess(List<Order> orderList) {  
 mOrderRvAdapter.setOrderList(orderList);  
}

5.使用swipeRefreshLayout快速下拉刷新

public void onOrderSuccess(List<Order> orderList) {  
 mSrlOrder.setRefreshing(false);  
 mOrderRvAdapter.setOrderList(orderList);  
 mSrlOrder.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {  
 @Override  
 public void onRefresh() {  
 int userId = TakeoutApp.sUser.getId();  
 mOrderFragmentPresenter.getOrderList(userId + "");  
 }  
 });  
}

**极光推送**

1. 支持.so文件

Android{

...  
 sourceSets {  
 main {  
 jniLibs.srcDirs = ['libs']  
 }  
 }  
}

2. 添加权限

<!-- Required -->  
<permission  
 android:name="com.heima.takeout35.permission.JPUSH\_MESSAGE"  
 android:protectionLevel="signature" />  
  
<!-- Required -->  
<uses-permission android:name="com.heima.takeout35.permission.JPUSH\_MESSAGE" />  
<uses-permission android:name="android.permission.RECEIVE\_USER\_PRESENT" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.WAKE\_LOCK" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.VIBRATE" />  
<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.WRITE\_SETTINGS" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
  
<!-- Optional. Required for location feature -->  
<uses-permission android:name="android.permission.SYSTEM\_ALERT\_WINDOW" /> <!-- 用于开启 debug 版本的应用在6.0 系统上 层叠窗口权限 -->  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  
<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS" />  
<uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.GET\_TASKS" />  
  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

3. 添加必要的组件和JPUSH\_APPKEY

<!-- Required SDK 核心功能-->  
<!-- 可配置android:process参数将PushService放在其他进程中 -->  
<service  
 android:name="cn.jpush.android.service.PushService"  
 android:enabled="true"  
 android:exported="false" >  
 <intent-filter>  
 <action android:name="cn.jpush.android.intent.REGISTER" />  
 <action android:name="cn.jpush.android.intent.REPORT" />  
 <action android:name="cn.jpush.android.intent.PushService" />  
 <action android:name="cn.jpush.android.intent.PUSH\_TIME" />  
 </intent-filter>  
</service>  
  
<!-- since 1.8.0 option 可选项。用于同一设备中不同应用的JPush服务相互拉起的功能。 -->  
<!-- 若不启用该功能可删除该组件，将不拉起其他应用也不能被其他应用拉起 -->  
<service  
 android:name="cn.jpush.android.service.DaemonService"  
 android:enabled="true"  
 android:exported="true">  
 <intent-filter >  
 <action android:name="cn.jpush.android.intent.DaemonService" />  
 <category android:name="com.heima.takeout35"/>  
 </intent-filter>  
</service>  
  
<!-- Required SDK核心功能-->  
<receiver  
 android:name="cn.jpush.android.service.PushReceiver"  
 android:enabled="true" >  
 <intent-filter android:priority="1000">  
 <action android:name="cn.jpush.android.intent.NOTIFICATION\_RECEIVED\_PROXY" />  
 <category android:name="com.heima.takeout35"/>  
 </intent-filter>  
 <intent-filter>  
 <action android:name="android.intent.action.USER\_PRESENT" />  
 <action android:name="android.net.conn.CONNECTIVITY\_CHANGE" />  
 </intent-filter>  
 <!-- Optional -->  
 <intent-filter>  
 <action android:name="android.intent.action.PACKAGE\_ADDED" />  
 <action android:name="android.intent.action.PACKAGE\_REMOVED" />  
 <data android:scheme="package" />  
 </intent-filter>  
</receiver>  
  
<!-- Required SDK核心功能-->  
<activity  
 android:name="cn.jpush.android.ui.PushActivity"  
 android:configChanges="orientation|keyboardHidden"  
 android:theme="@android:style/Theme.NoTitleBar"  
 android:exported="false" >  
 <intent-filter>  
 <action android:name="cn.jpush.android.ui.PushActivity" />  
 <category android:name="android.intent.category.DEFAULT" />  
 <category android:name="com.heima.takeout35" />  
 </intent-filter>  
</activity>  
<!-- SDK核心功能-->  
<activity  
 android:name="cn.jpush.android.ui.PopWinActivity"  
 android:configChanges="orientation|keyboardHidden"  
 android:exported="false"  
 >  
 <intent-filter>  
 <category android:name="android.intent.category.DEFAULT" />  
 <category android:name="com.heima.takeout35" />  
 </intent-filter>  
</activity>  
  
<!-- Required SDK核心功能-->  
<service  
 android:name="cn.jpush.android.service.DownloadService"  
 android:enabled="true"  
 android:exported="false" >  
</service>  
  
<!-- Required SDK核心功能-->  
<receiver android:name="cn.jpush.android.service.AlarmReceiver" />  
  
<!-- User defined. 用户自定义的广播接收器-->  
<receiver  
 android:name=".utils.MyReceiver"  
 android:enabled="true">  
 <intent-filter>  
 <!--Required 用户注册SDK的intent-->  
 <action android:name="cn.jpush.android.intent.REGISTRATION" />  
 <!--Required 用户接收SDK消息的intent-->  
 <action android:name="cn.jpush.android.intent.MESSAGE\_RECEIVED" />  
 <!--Required 用户接收SDK通知栏信息的intent-->  
 <action android:name="cn.jpush.android.intent.NOTIFICATION\_RECEIVED" />  
 <!--Required 用户打开自定义通知栏的intent-->  
 <action android:name="cn.jpush.android.intent.NOTIFICATION\_OPENED" />  
 <!-- 接收网络变化 连接/断开 since 1.6.3 -->  
 <action android:name="cn.jpush.android.intent.CONNECTION" />  
 <category android:name="com.heima.takeout35" />  
 </intent-filter>  
</receiver>  
  
<!-- Required. For publish channel feature -->  
<!-- JPUSH\_CHANNEL 是为了方便开发者统计APK分发渠道。-->  
<!-- 例如: -->  
<!-- 发到 Google Play 的APK可以设置为 google-play; -->  
<!-- 发到其他市场的 APK 可以设置为 xxx-market。 -->  
<!-- 目前这个渠道统计功能的报表还未开放。-->  
<meta-data android:name="JPUSH\_CHANNEL" android:value="developer-default"/>  
<!-- Required. AppKey copied from Portal -->  
<meta-data android:name="JPUSH\_APPKEY" android:value="22aee046d2d27e81f591fe55"/>

4. 初始化极光推送

@Override  
public void onCreate() {  
 super.onCreate();  
 sInstance = this;  
  
 sUser = new User();  
 sUser.setId(-1); //空用户，未登录  
  
 JPushInterface.setDebugMode(true);  
 JPushInterface.init(this);  
}

5. 广播接收者接收推送数据

public class MyReceiver extends BroadcastReceiver {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 Bundle bundle = intent.getExtras();  
 String message = bundle.getString(JPushInterface.EXTRA\_MESSAGE);  
 Log.e("jpush", "自定义消息：" + message);  
 String extras = bundle.getString(JPushInterface.EXTRA\_EXTRA);  
 Log.e("jpush", "额外消息：" + extras);

**订单状态自动刷新（观察者模式）**

1. 广播告诉OrderObservable新数据来了

HashMap<String, String> data = processExtra(extras);  
  
OrderObservable.getInstance().newMsgComing(data);

2. OrderObservable 通知所有观察者具体变化

public class OrderObservable extends Observable {  
 ...  
 public void newMsgComing(HashMap<String, String> data) {  
 //b.通知所有观察者新消息来了  
 setChanged(); //发生了变化  
 notifyObservers(data); //具体什么变化  
 }  
}

3. 如果此数据源有观察者

public OrderRvAdapter(Context context) {  
 mContext = context;  
 OrderObservable.getInstance().addObserver(this);  
}

4. 观察者的响应

@Override  
 public void update(Observable observable, Object data) {  
 Log.e("jpush", "观察者收到新的消息。。。。");  
 HashMap<String,String> map = (HashMap<String, String>) data;  
 String pushOrderId = map.get("orderId");  
 String pushType = map.get("type");  
  
 int position = -1; //-1表示所有都不匹配  
 for(int i=0;i<mOrderList.size();i++){  
 Order order = mOrderList.get(i);  
 if(order.getId().equals(pushOrderId)){  
 //匹配上,设置新的状态  
 order.setType(pushType);;  
 //只刷新要修改的那一个  
 position = i;  
 notifyItemChanged(position);  
 }  
 }  
 }

**5.加入RxJava**

* 1. **配置环境module的build.gradle**

compile **'io.reactivex:rxjava:1.3.0'** compile **'io.reactivex:rxandroid:1.2.1'** compile **'com.squareup.retrofit2:adapter-rxjava:2.1.0'**

* 1. **Retrofit的Service加入Rx元素**

Retrofit retrofit = new Retrofit.Builder().baseUrl(ENDPOINT)  
 .addConverterFactory(GsonConverterFactory.create())  
 .addCallAdapterFactory(RxJavaCallAdapterFactory.create())

.build();  
  
 public interface TakeoutService {  
 @GET("order")  
 fun getOrderListByRxjava(@Query("userId") userId: String): Observable<ResponseInfo>

}  
}

* 1. **使用Rx接口访问数据**

orderObservable.subscribeOn(Schedulers.io()).observeOn(AndroidSchedulers.mainThread())  
 .subscribe(  
 {  
 val str = it.data  
 parseJson(str)  
 },  
 {  
 Log.e("order", it.message)  
 },  
 {  
 Log.e("order", "on Complete!")  
 }  
 )

## 每日作业

1. 能实现短信验证
2. 会使用OrmLite保存缓存
3. 会配置极光推送
4. 了解观察者模式