# **[Android消息传递之组件间传递消息](http://www.cnblogs.com/whoislcj/p/5593056.html)**

#### **需求场景：**

      之前做图片社交App的时候，需要处理一个点赞数据的同步，比如在作品的详情页点赞 需要同时更新列表页该作品的点赞数量。

#### **方式一：通过动态注册BroadcastReceiver**

##### **1.）内部定义BroadcastReceiver**

[IMG_256](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

//同步数据广播

private BroadcastReceiver dataSynReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

if (intent.getAction().equals("com.whoislcj.broadcastReceiver.dataSynAction")) {

int count = intent.getIntExtra("count", 0);

//接下来执行同步操作 }

}

};

[IMG_257](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

##### **2.)在Activity对应的生命周期注册/解注册 onCreate/onStart/onResume 注册 onDestroy/onStop/onPause 解注册**

 注册

[IMG_258](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

//同步数据广播

private BroadcastReceiver dataSynReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

if (intent.getAction().equals("com.whoislcj.broadcastReceiver.dataSynAction")) {

int count = intent.getIntExtra("count", 0);

runOnUiThread(new Runnable() {

@Override

public void run() {

//接下来执行同步操作 }

});

}

}

};

[IMG_259](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

 解注册

//解除同步数据广播

private void unRegisterReceiver() {

unregisterReceiver(dataSynReceiver);

}

##### **3.）在触发数据同步的地方发送消息**

Intent intent = new Intent();

intent.setAction("com.whoislcj.broadcastReceiver.dataSynAction");//设置Action

intent.setPackage(getPackageName());//设置包名使广播只能被app内接收者接收

intent.putExtra("count", 5);//添加附加信息

sendBroadcast(intent);

##### **4.）分析优缺点**

   优点：可以设置不同页面接收消息的优秀级，而且也可以采用发送有序广播的方式终止将同步消息发给下一级，同时也可以修改消息传递给下一级。

   缺点：广播传递本身是有安全隐患的，需要设置权限，每一个Activity都要定义、注册，解注册广播无形中加大了工作量和维护成本。

#### **方式二：通过自己管理事件监听总线**

##### **1.）声明一个数据同步接口**

[IMG_260](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

/\*\*

\* 赞同步接口

\*/

public interface IDataSynListener {

void onDataSyn(int count);

}

[IMG_261](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

##### **2.）定义一个单例管理监听总线**

[IMG_262](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

public class DataSynManager {

private LinkedList<IDataSynListener> autoListeners = new LinkedList();//监听集合

private static DataSynManager mInstance;//单例引用

/\*\*

\* 获取单例引用

\*

\* @return

\*/

public static DataSynManager getInstance() {

if (mInstance == null) {

synchronized (DataSynManager.class) {

if (mInstance == null) {

mInstance = new DataSynManager();

}

}

}

return mInstance;

}

/\*\*

\* 添加同步数据监听

\*/

public void registerDataSynListener(IDataSynListener autoDataListener) {

if (autoListeners == null) {

autoListeners = new LinkedList<IDataSynListener>();

}

if (!autoListeners.contains(autoDataListener)) {

autoListeners.add(autoDataListener);

}

}

/\*\*

\* 移除同步数据监听

\*/

public void unRegisterDataSynListener(IDataSynListener autoDataListener) {

if (autoListeners == null) {

return;

}

if (autoListeners.contains(autoDataListener)) {

autoListeners.remove(autoDataListener);

}

}

/\*\*

\* 执行数据同步

\*

\* @param count

\*/

public void doDataSyn(final int count) {

if (autoListeners == null) {

autoListeners = new LinkedList();

}

new Handler().post(new Runnable() {

@Override

public void run() {

for (IDataSynListener dataSynListener : autoListeners) {

dataSynListener.onDataSyn(count);

}

}

});

}

/\*\*

\* 清除所有监听者

\*/

public void release() {

if (autoListeners != null) {

autoListeners.clear();

autoListeners = null;

}

}

/\*\*

\* 赞同步接口

\*/

public interface IDataSynListener {

void onDataSyn(int count);

}

}

[IMG_263](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

##### **2.)在Activity对应的生命周期添加监听/移除监听 onCreate/onStart/onResume 添加监听 onDestroy/onStop/onPause 移除监听**

添加监听

DataSynManager.getInstance().registerDataSynListener(dataSynListener);

移除监听

DataSynManager.getInstance().unRegisterDataSynListener(dataSynListener);

声明一个监听

[IMG_264](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

DataSynManager.IDataSynListener dataSynListener=new DataSynManager.IDataSynListener() {

@Override

public void onDataSyn(int count) {

//接下来执行同步操作 }

};

[IMG_265](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

##### **3.）在触发数据同步的地方发送消息**

DataSynManager.getInstance().doDataSyn(5);

##### **4.）分析优缺点**

 优点：相对广播传输安全一点，对于总线数量过大的时候效率可能会比较低。

 缺点：不能设置优先级，不能终止传递，不能修改消息。

小结：

  以上两种方式是在EventBus出现之前我所使用的实现方式，如果有更好的实现方式，也可以互相学习一下。接下来就是来学习一个EventBus是如何管理事件总线的，以及优缺点。