事件框架总线 Otto

一、Otto 是什么？

Otto 是基于 Guava 项目的 Android 系统的一个 Event Bus 模式类库，如果你在 Android 程序开发的过程中想要不同的组件之间进行有效的通信可以使用这个库。通过 Otto 库可以降低程序之间的耦合性。

二、Otto 解决了哪些问题？

1、用来简化应用程序组件之间的通讯，例如复杂的界面跳转 ActivityA 跳转到 ActivityB，ActivityB 跳转到 ActivityC，此时需要从 ActivityC 中获取到用户的操作数据返回到 ActivityA 和 ActivityB 中显示，或者在 Activity 中有多个 Fragment，其中一个 Fragment 里的数据变化需要同步更新其它 Fragment 的数据变化，这时候如果使用 Interface 的方式进行它们之间的交互则比较复杂，耦合度也高。

2、简化了程序，使代码变得更加优美。

三、Otto 如何使用？

Otto 采用注解定义订阅/发布者角色的方式：

@Subscribe：标明其为订阅者。通过此注解来告诉 Bus 总线，该方法订阅了一个事件，同时 Bus 总线可以通过此注解来找到需要调用的方法；

@Produce：标明其为发布者；

register()方法：通过 Bus 类的此方法来实现某个类的注册，这样 Bus 总线便可以通过此方法来查找注册的类；

unregister()方法：通过此方法实现从 Bus 事件总线中注销。

四、使用示例：

这里假设某个 Activity 发布消息数据给其 Fragment，具体使用步骤如下：

1）添加依赖

```
dependencies {
    ……
    / /Otto 所需要依赖的包
    compile 'com.squareup:otto:1.3.8'
}
```

2） 创建自定义 Bus 类

```
//使用单例模式创建 MyBus 类方便后面使用
public class MyBus extends Bus {
    private static MyBus bus;
    public static MyBus getInstance() {
        if (bus == null) {
            bus = new MyBus();
        }
        return bus;
    }
}
```

3）创建 TransferData 类作为组件间通信传递数据的载体

```
public class TransferData {
    public String content;
```

```
    public TransferData(String content) {
        this.content = content;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }
}
```

4）在接收数据端 Fragment 里

```
@Override
public void onStart() {
    super.onStart();
    //使用 Bus 类的 register()方法注册到 bus 事件总线中
    MyBus.getInstance().register(this);
}

//定义订阅者并接收来自 Activity 中的消息，参数类型为 3）中定义的类型
@Subscribe
```

```
public void setContent(TransferData data) {
    System.out.println(data.getContent());
}

//接收到数据的变化
@Subscribe
public void onMyScrollChange(ScrollEventData data) {
    System.out.println(data);
}

@Subscribe
public void onDataChange(String str) {
    System.out.println(str);
}

@Override
public void onStop() {
    super.onStop();
    //从 Bus 事件总线中注销
    MyBus.getInstance().unregister(this);
}
```

5） 在发布数据 Activity 端，通过 Bus 将 TransferData 类型的数据发布到

Fragment

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    MyBus.getBusInstance().register(this);
}
//注明发布者
@Produce
public TransferData produceFragmentData() {
    return new TransferData("The data comes from activity");
}
//通知所有@Subscribe 匹配 TransferData 类型参数的方法执行
public void postDataToFragment(){
    MyBus.getInstance().post(produceFragmentData());
}

@Override
public void onDestroy() {
    super.onStop();
    //从 Bus 事件总线中注销
    MyBus.getInstance().unregister(this);
}
```

五、总结：

Otto 是针对事件提供统一订阅和发布以达到组件间通信的解决方案。作用就是简化 android 应用内组件的通信。

使用时需要注意如下两点：

1）被@Subscribe，@Produce 注解的方法必须定义在直接的作用类上，而不能定义在基类。

2）发布者和订阅者都需要 register()和 unregister()。

推荐 Otto 官方网站：

http://square.github.io/otto/