**android EventBus**

参考： <http://blog.csdn.net/harvic880925/article/details/40787203>

1. **onEvent:**如果使用onEvent作为订阅函数，那么该事件在哪个线程发布出来的，onEvent就会在这个线程中运行，也就是说发布事件和接收事件线程在同一个线程。使用这个方法时，在onEvent方法中不能执行耗时操作，如果执行耗时操作容易导致事件分发延迟。  
   **onEventMainThread:**如果使用onEventMainThread作为订阅函数，那么不论事件是在哪个线程中发布出来的，onEventMainThread都会在UI线程中执行，接收事件就会在UI线程中运行，这个在Android中是非常有用的，因为在Android中只能在UI线程中跟新UI，所以在onEvnetMainThread方法中是不能执行耗时操作的。  
   **onEventBackground:**如果使用onEventBackgrond作为订阅函数，那么如果事件是在UI线程中发布出来的，那么onEventBackground就会在子线程中运行，如果事件本来就是子线程中发布出来的，那么onEventBackground函数直接在该子线程中执行。  
   **onEventAsync：**使用这个函数作为订阅函数，那么无论事件在哪个线程发布，都会创建新的子线程在执行onEventAsync.
2. Subscription是不区分观察者是谁(这句话得意思是,一个Activity对象中可能有多个Subscription,另一个Activity也有多个,但是他们不区分)，即它只根据EventType去查找所有Subscription
3. EventBus流程：
4. 通过静态方法EventBus.getDefault()得到EventBus的全局单例（采用双重校验并加锁）；
5. 调用EventBus.register()，先通过Class名为KEY从缓存中查找是否有该类的EventBus事件处理方法的列表，有则直接返回该列表；否则new一个EventBus事件处理方法的列表，通过反射得到以onEvent开头的方法，将该方法的信息（类名、参数类型等）加到EventBus事件处理方法的列表；
6. EventBus.post()判断事件处理的类型，从map中拿到处理EventBus事件相关方法的信息，从而知道应该在哪个线程处理，并发送到相应的线程中；
7. 如果是发送线程和处理线程为同一线程则通过反射调用相应的onEvent开头的方法；
8. 发送线程和处理线程为不同线程，且处理线程不是UI线程，则直接通过线程池在子线程以阻塞的形式处理EventBus事件；
9. 发送线程和处理线程为不同线程，且处理线程是UI线程，则通过Handler消息机制处理EventBus事件。
10. 调用EventBus.unregister()通过Class名为KEY从缓存中清除该类的EventBus事件处理方法的列表。