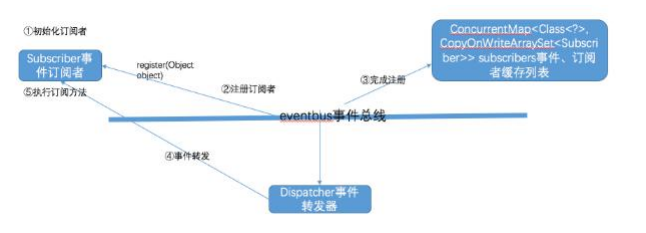
1. 事件总线的具体功能

用户使用EventBus来注册或者注销监听器。EventBus通过post函数广播某类事件，再通过register函数将把此类事件对应的监听器的事件处理器注册到一个multimap容器中，该容器的建是事件类型，值是该事件类型的事件处理器。



通俗的例子：验证码

用户: 软件

事件：验证码

监听器：软件用户

事件处理器：用户在软件上填写验证码

//验证码

public class Verifi\_code{

public String message;

public Verifi\_code(String message){

this.message = message;

}

//软件用户

public class User{

@Subscribe

public void Fill(Verifi\_code code){

}

}

//软件代码

public class Main{

public static void main(String[] args){

//创建事件总线

EventBus eventBus = new EventBus(“verification”);

//创建监听器

User user = new User();

//注册监听器

event.register(user);

//广播事件

eventBus.post(new Verifi\_code(“654358”);

}

}

在这里，事件总线EventBus被软件用来实现给指定用户发送验证码。

重要类：

1. EventBus: 事件总线，也有通过post函数广播事件的功能
2. AsyncEventBus: 继承自EventBus, 实现的是异步进程的事件总线
3. Subscriber: 这个类字面上是订阅者，但是还包括了事件处理器等信息。
4. SubscriberRegistry: 订阅注册表，通过一对多的map映射来实现某个类的事件对应到多个Subscriber。
5. Dispatcher: 事件分发器，工作是分发事件给Subscriber。
6. DeadEvent: 死信对象(没有订阅者关注的事件)
7. SubscribeExceptionHandler: 订阅者抛出异常的处理器
8. SubscribeExceptionContext: 订阅者抛出异常的上下文对象

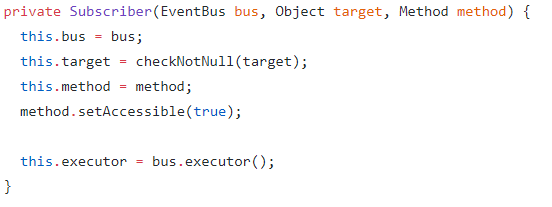
两个注释：

1. @Subscribe: 用于标识事件处理器的注解，当EventBus通过post函数广播一个事件后，相应的Subscriber将会得到通知并执行事件处理器。
2. @AllowConcurrentEvents: 标识当前的Subscriber的处理方法是线程安全的。
3. 具体功能分析之监听器的注册过程

EventBus通过一个一对多的映射容器Multimap<Class<?>, Subscriber>建立事件和监听器之间的映射关系。监听器实例user的注册event.register(user)，其实是根据user所对应的事件类型实例把user放到这个容器里。维护事件与事件监听器之间的对应关系，如果某类事件发生，则可以从该对应关系里查找出应该将该类事件派发至哪些事件监听器。

事件监控器Subscriber实质上是一个方法method，方法需要目标对象target和目标方法method。Subscriber的UML图的部分内容如下：

|  |
| --- |
| Subscriber |
| -bus: EventBus  target: Object  -method: Method  -executor: Executor |
| -Subscriber(): void  ... |



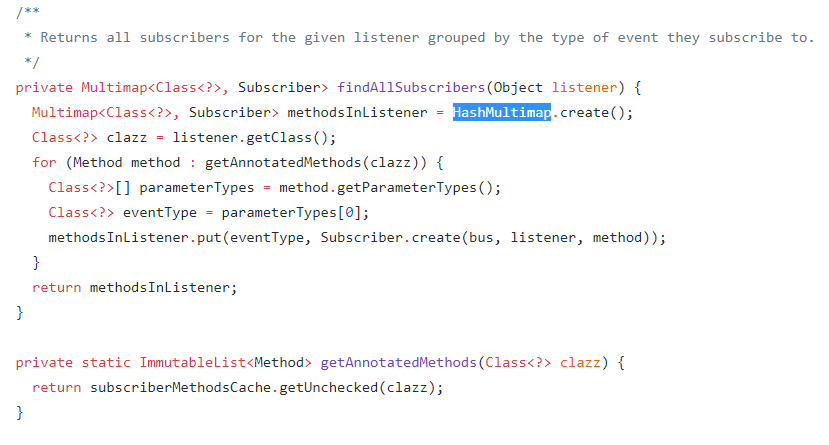
|  |
| --- |
| EventBus |
| -logger: Logger  -identifier: String  -executor: Executor  -exceptionHandler: SubscriberExceptionHandler  -subscribers: SubscriberRegistry  -dispatcher: Dispatcher |
| +identifier(): String  +register(): void  +unregister(): void  +post(): void  +toString(): String  +handleException(): void  -logger(): Logger  -message(): Strirng |

从图中可以看出Subscriber还包括了EventBus类型的bus和Excutor类型，从构造函数中的

this.executor = bus.excutor()可以看出bus的作用是为了获取bus中的executor。executor是Eventbus中对监听器事件处理方法的执行。

接下来看register函数，





findAllSubscribers(listener)的作用是从传入的类实例对象listener中找出所有事件(Event)和事件监听器Subscriber的对应关系。该函数内有一个反映射的过程，原本是clazz映射到listenner。通过Class<?> clazz = listener.getClass();可以得到该事件监听器被哪个事件映射。

在for循环里，每循环一次，找到事件实例clazz的一种处理方法，再判断该处理方法是否合理，如果合理则eventType的值为该处理方法method的参数类型。最后通过put函数将事件类型实例与事件监视器实例放入multimap类型的容器里。事件监视器的注册完成。