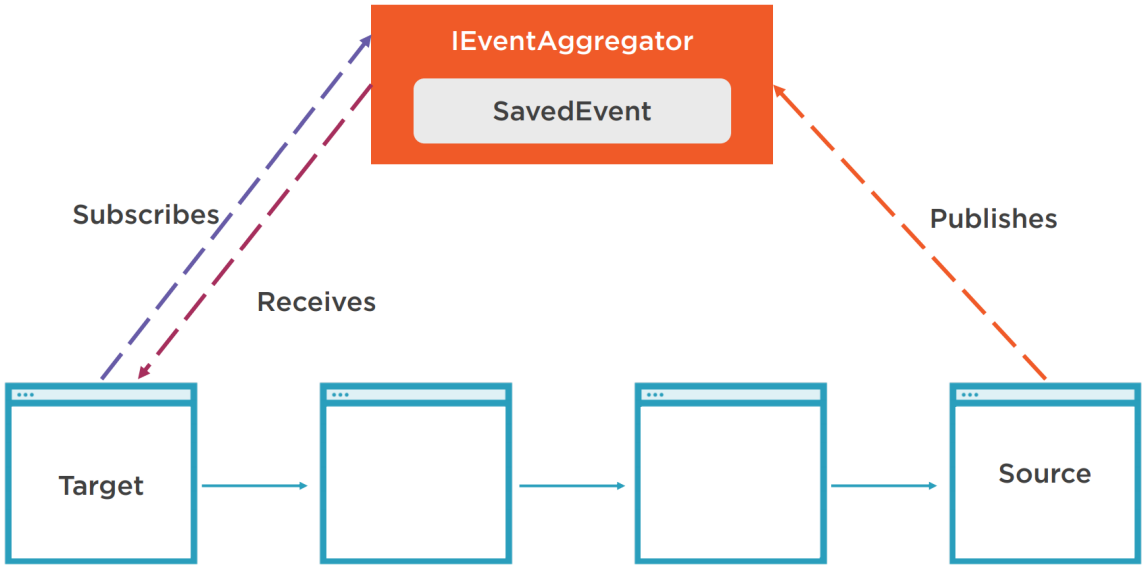
Event Aggregator事件聚合器

IEventAggregator

- 松耦合基于事件通讯
- 多个发布者和订阅者
- 微弱的事件
- 过滤事件
- 传递参数
- 取消订阅

该功能主要作用为, 事件聚合器负责接收订阅以及发布消息。订阅者可以接收到发布者发送的内容。

例如: AViewModel订阅了一个消息接收的事件, 然后BViewModel当中给指定该事件推送消息,此时AViewModel接收BViewModel推送的内容。

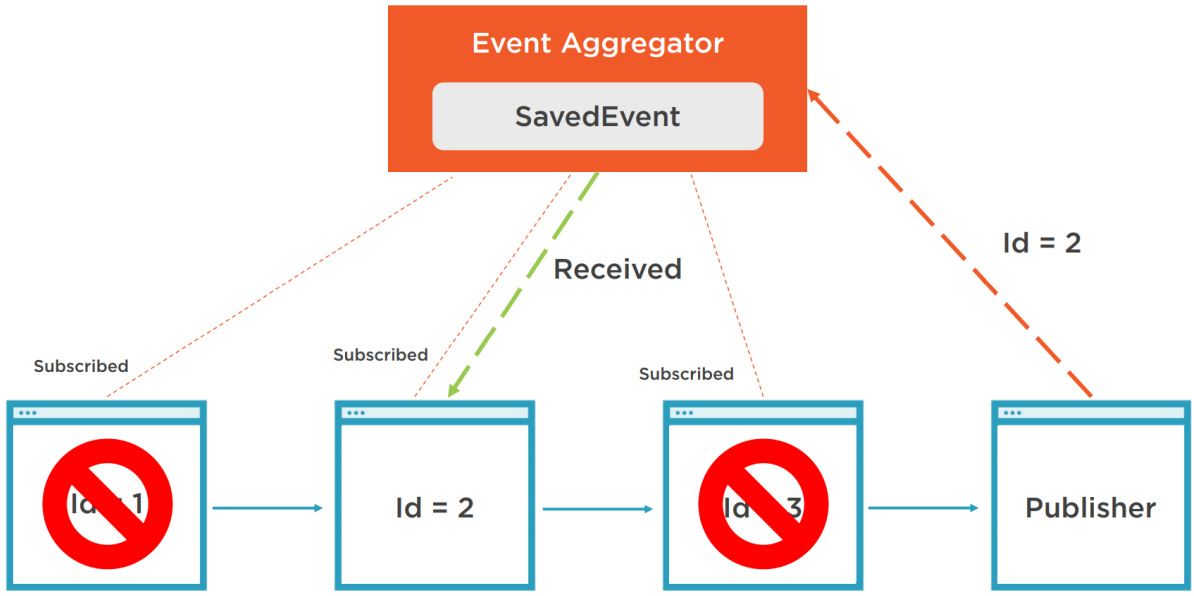



```
//创建事件
public class SavedEvent : PubSubEvent<string> { }
//发布
IEventAggregator.GetEvent<SavedEvent>().Publish("some value");
```

```
//订阅
IEventAggregator.GetEvent<SavedEvent>().Subscribe(.Subscribe(message=>
{
    //do something
});
```

Filtering Events

在实际的开发过程当中,我们往往会在多个位置订阅一个事件, 但是对于订阅者而言, 他并不需要接收任何消息



在Prism当中, 我们可以指定为事件指定过滤条件

```
eventAggregator.GetEvent<MessageSentEvent>()
               .Subscribe(arg =>
               {
                   //do something
               },
               ThreadOption.PublisherThread,
               false,
               //设置条件为token等于“MessageListViewModel” 则接收消息
               message => message.token.Equals(nameof(MessageListViewModel)));
```

关于Subscribe当中的4个参数, 详解:

1.**action**: 发布事件时执行的委托。

2.**ThreadOption**枚举: 指定在哪个线程上接收委托回调。

3.**keepSubscriberReferenceAlive**: 如果为true，则Prism.Events.PubSubEvent保留对订阅者的引用因此它不会收集垃圾。

4.**filter**: 进行筛选以评估订阅者是否应接收事件。

Unsubscribe

为注册的消息取消订阅, Prism提供二种方式取消订阅

1.通过委托的方式取消订阅

```
var event = IEventAggregator.GetEvent<MessageSentEvent>();
event.Subscribe(OnMessageReceived);
event.Unsubscribe(OnMessageReceived);
```

2.通过获取订阅者token取消订阅

```
var _event = eventAggregator.GetEvent<MessageSentEvent>();
var token = _event.Subscribe(OnMessageReceived);
_event.Unsubscribe(token);
```