我们可以查看一下容器中所有的事件监听器，输入`symfony console debug:container`，标签我们输入`kernel.event_listener`。这里就列出了当前项目中所有的事件监听器。如果同一个事件有多个监听器的话，我们可以设置一下监听器的优先级。比如说`kernel.exception`事件，默认情况下它的优先级是0，我们可以设置为负数来降低它的优先级，也可以设置为正数来提升优先级。

我们打开services.yaml文件，在我们编写的监听器类下，我们添加一个配置，我们配置`priority`，这里我们设置为10。再次打开控制台执行命令，现在我们看到我们的事件监听类，它的优先级为10。如果有其他的事件监听器来监听我们编写的事件，那么它的优先级将会高一些，同样的，我们也可以为事件订阅器来设置优先级。

```
#config/services.yaml

# ...
services:
    # ...
    App\Listener\CommentSubmitListener:
        tags:
            - {name: 'kernel.event_listener', event:
App\Event\AfterCommentSubmitEvent, method: 'onCommentSubmit', priority:
10}
```

回到文档，在事件对应的处理方法后面可以设置优先级，这里就不再设置了。

我们继续往下看，Symfony为我们提供了一个通用的事件对象，我们查看一下文档，通用的事件类名是`GenericEvent`。我们查看这个类，回到项目，双击shift键，粘贴。

我们来查看这个类，这个类有一个`$subject`属性，这个属性我们可以设置为任意的对象。我们查看它的构造方法，构造方法传入了一个对象和一个数组，我们再看它的父类。在父类中有个`$propagationStopped`属性，如果这个属性为真的话，我们可以在时间监听器或者订阅器中停止事件的处理。

我们来修改之前编写的代码，打开事件类我们让它继承`GenericEvent`，这里我们就不需要在类中设置属性了，注释类中的代码，现在getComment()方法已经被注释了，我们需要修改我们的监听器类和订阅器类。这里我们需要使用`getSubject()`方法来替代，同样的修改订阅器类，我们可以在订阅器或者监听器中对$event对象的`$propagationStopped`属性进行判断。

如果为真的话，停止对事件的处理。首先回到监听器，在监听器中我们使用`$event->stopPropagation()`方法来停止事件。然后在订阅器中我们来进行一个判断，如果`$event->isPropagationStopped()`，我们直接返回。

回到浏览器，我们再次提交一个评论，点击提交。我们还需要修改一下PostController中的代码，这里替换为`getSubject()`。刷新页面，我们查看一下最后的结果，最后评论的h1标签已经过滤掉了，但是敏感词并没有被替换。这就是使用GenericEvent类的一个好处。

Symfony并没有其他系统所提供的一些钩子的概念，但是我们可以定义一些自己的事件，然后通过事件来对某些方法进行处理，来作为钩子。Symfony定义了很多事件，也使用了很多事件处理类，可以说事件系统是Symfony的灵魂。

回到项目，还记得我们之前的handleRaw()方法吗？我们再来回顾一下，双击shift键输入HttpKernel，打开HttpKernel类。我们查看结构，找到handleRaw()方法，在handleRaw()方法中首先发送了一个RequestEvent。

如果你的网站正在停机维护，你可以监听这个事件，直接返回一个正在维护中的页面，这就很容易。然后是获取$controller，在获取$controller之后发送一个ControllerEvent，如果你监听ControllerEvent，你可以对controller方法修改。下面是ControllerArgumentsEvent可以对controler的参数进行修改。最后controller方法的返回值，如果不是Response对象发送一个ViewEvent，对controller方法的返回值进行处理。最后一定要返回一个Response对象。

我们继续往下看，在filterResponse()方法中，发送了一个响应事件。在ResponseEvent事件中我们依然可以修改响应的内容。我们查看finishRequest()方法，在finishRequest()方法中又发送了一个事件。

可以说Symfony处理请求的整个流程都是由事件系统来进行调度的，这样就对代码进行了解耦合，大大的增加了系统的扩展性。

在下节课，我们来继续使用订阅器来解决管理端的一些问题。