我们先来重置一下数据库中的数据，输入`symfony console doctrine:fixtures:load`，现在数据库中的数据就已经被重置了。另外我们删除一下文件上传目录中的所有文件，我这里已经删除过了。

回到浏览器，我们打开博客首页，打开第一篇文章。在第一篇文章后面，我们来提交一个评论，上传一个文件，提交。回到管理端，打开评论列表，我们来删除刚刚提交的评论。回到项目，数据库中的数据我们已经删除了，但是我们上传的文件没有删除，我们打开数据库客户端。

查看一下file表，file表中我们刚刚上传的文件的数据也还有。我们希望在删除评论时，如果评论中有上传的文件，一块把文件表中的数据进行删除，同时把上传目录中的文件进行删除。

我们查看一下管理端的Controller，打开CommentCrudController类，CommentCrudController类继承于AbstractCrudController。我们打开这个类，我们查看一下类中的方法，在类中有个`delete()`方法，`delete()`方法就是我们删除评论时的controller方法。

首先它会发送一个`BeforeCrudActionEvent`事件，我们继续往下看，然后会发送`BeforeEntityDeleteEvent`事件。在try块中真正的删除对应的评论对象，在删除完之后会发送一个`AfterEntityDeleteEvent`。我们可以使用订阅器来处理这两个事件，在评论对象删除之前，我们来获取评论中所有的文件对象。

当数据删除之后，如果没有异常的话，我们来循环遍历所有的文件对象，来删除数据库的数据，同时来删除上传的文件。打开控制台，我们输入`symfony console make:subscriber`，类的名称我们叫做`DeleteCommentSubscriber`。我们要处理的事件`BeforeEntityDeleteEvent`，我们需要复制全类名，按着command键点击`BeforeEntityDeleteEvent`。复制全类名，粘贴。

我们打开订阅器类，我们还要处理另外一个事件`AfterEntityDeleteEvent`，按着command键复制这一行，修改一下。同时修改对应的方法名。然后创建对应的方法，按着command键复制上面的方法进行修改，我们需要修改一下参数。参数是`AfterEntityDeleteEvent`，我们先来看一下`BeforeEntityDeleteEvent`类的源码。按着command键点击`BeforeEntityDeleteEvent`，在事件类中有一个实例属性。这个实例属性就是我们的Comment对象的实例，对应的有一个`getEntityInstance()`方法。

回到我们的订阅器，我们通过事件来获取评论对象，$comment等于`$event->getEntityInstance()`，然后我们来进行一个条件判断。如果$comment对象是Comment类的一个实例，我们来获取评论中的所有文件`$files=$comment->getFiles()`。我们获取到了评论对象的文件集合。我们需要在评论对象删除之后再次使用，可以在订阅器类中添加一个属性来保存文件对象，我们可以定义一个数组属性，名称叫做`$needDeleteFiles`。

然后我们对所有的文件进行遍历，将遍历到的文件对象保存到数组中。foreach，当删除完评论对象之后，我们对数组进行遍历，在数组前我们添加一个注释。数组是FileManageed对象的集合，在这里添加for循环，`$this->needDeletedFiles as $deletedFile`。

要删除FileManaged对象，我们需要使用EntityManager对象，添加构造方法。在构造方法中，我们通过依赖注入的方式来使用EntityManager对象，点击参数，按照alt键点回车我们初始化属性。在下面循环中我们使用`$this->em->remove($deletedFile)`。同时我们要删除硬盘上的文件，我们要获取文件在硬盘上的绝对路径，首先我们要获取项目目录在硬盘上的路径。

打开config目录下的services.yaml文件，在参数这里我们定义一个参数`project_dir`，我们可以使用Symfony内置的参数，使用双百分号`kernel.project_dir`。

```
#config/services.yaml

parameters:
    # ...
    project_dir: '%kernel.project_dir%'
```

回到订阅器，我们通过依赖注入的方式来使用ParameterBag对象，按着alt键点回车，在方法中我们来获取项目的绝对路径，使用unlink方法来删除文件。

首先$projectDir加上/public目录，再加上$deletedFile->getPath()，最后在循环体外，我们需要使用$this->em->flush()方法来提交一下删除。我们来删除一下文件上传目录中的所有文件，然后删除一下file表中的所有数据行。

打开浏览器，再次访问详情页，我们重新提交一个评论。添加几个文件，点击提交，回到评论列表，刷新。我们来删除这条评论，现在我们来查看一下文件上传目录，文件上传目录中已经没有我们上传的文件了。我们查看一下file表，现在file表中，也没有我们刚刚上传的文件数据了，这就是使用Symfony的事件系统来实现的一个钩子的案例。

回到项目，我们使用了EasyAdminBundle，这个Bundle的action方法中提供了很多的事件，我们可以通过对这些事件来进行处理，来实现不同的功能。回到订阅器类，我们使用$parameterBag对象来获取到了项目的路径。我们还可以使用另外一种方法，直接的在构造方法中注入项目的路径。

打开services.yaml文件，在_defaults配置键下，我们添加一个bind配置。在bind配置中我们配置一些参数的名称，这里配置$projectDir。它指向的是项目的目录kernel.project_dir。再回到订阅器类，这次我们不注入ParameterBag对象，我们直接使用刚刚设置的$projectDir参数，注释代码，按着alt键初始化这个属性。

```
#config/services.yaml

services:
    # default configuration for services in *this* file
    _defaults:
        autowire: true      # Automatically injects dependencies in your
services.
        autoconfigure: true # Automatically registers your services as
commands, event subscribers, etc.
        bind:
            $projectDir: '%kernel.project_dir%'
```

```
#src/EventSubscriber/DeleteCommentSubscriber.php

class DeleteCommentSubscriber implements EventSubscriberInterface
{
    /**
     * @var FileManaged[]
     */
    private array $needDeletedFiles = [];
```

```
    /**
     * @var EntityManagerInterface
     */
    private EntityManagerInterface $em;
//    /**
//     * @var ParameterBagInterface
//     */
//    private ParameterBagInterface $parameterBag;
    private $projectDir;

    public function __construct(EntityManagerInterface $em, $projectDir)
    {
        $this->em = $em;
//        $this->parameterBag = $parameterBag;
        $this->projectDir = $projectDir;
    }

    public function onBeforeEntityDeletedEvent(BeforeEntityDeletedEvent
$event)
    {
        $comment = $event->getEntityInstance();
        if ($comment instanceof Comment) {
            $files = $comment->getFiles();
            foreach ($files as $file){
                $this->needDeletedFiles[] = $file;
            }
        }
    }

    public function onAfterEntityDeletedEvent(AfterEntityDeletedEvent
$event)
    {
//        $projectDir = $this->parameterBag->get('project_dir');

        foreach ($this->needDeletedFiles as $deletedFile){
            $this->em->remove($deletedFile);

            unlink($this->projectDir . '/public/' . $deletedFile-
>getPath());
        }
        $this->em->flush();
    }

    public static function getSubscribedEvents()
    {
        return [
            BeforeEntityDeletedEvent::class =>
'onBeforeEntityDeletedEvent',
            AfterEntityDeletedEvent::class => 'onAfterEntityDeletedEvent',
        ];
    }
}
```

在下面的方法中我们来进行修改，这里$projectDir替换为`$this->projectDir`。回到浏览器，我们再次提交评论，添加几个文件，点击提交，查看项目的文件上传目录。这里就是我们刚刚上传的文件。

回到管理端，刷新评论列表，再次点击删除，现在我们的文件就已经删除了，我们刚刚配置的参数就生效了，我们可以使用这种方式来为项目注入参数，现在我们已经有了一个不错的管理端。

在下节课，我们将创建用户类来将管理端的页面保护起来。