文件的上传是在表单提交过程中，如果其他表单行或者数据在插入数据库时抛了异常，那么文件就会多上传，造成服务器资源浪费。

我们写代码时保持类功能的纯粹性，`FileManagedType`类应该只负责文件的上传并进行类型转换就好了。我们把文件的`upload()`方法，放到文件对象已经插入数据库之后。

优化代码：

1. 在`File`Entity类中添加一个属性，仅用于保存`UploadedFile`对象，不插入到数据库，不用配置ORM信息

```
class File
{
    ...

    /**
     * @var UploadedFile|null
     */
    private $uploadedFile;

    ...

    /**
     * @return UploadedFile|null
     */
    public function getUploadedFile(): ?UploadedFile
    {
        return $this->uploadedFile;
    }

    /**
     * @param UploadedFile|null $uploadedFile
     */
    public function setUploadedFile(?UploadedFile $uploadedFile): void
    {
        $this->uploadedFile = $uploadedFile;
    }

}
```

2.修改Handler类，添加一个方法用于生成新的文件名，并且修改`upload()`方法

```
class UploadHandler
{
    /**
     * @var NamerInterface
     */
    private $namer;
    /**
```

```
     * @var StorageInterface
     */
    private $storage;

    /**
     * @var string
     */
    private $distDir;

    /**
     * @param StorageInterface $storage 用于操作文件
     * @param string $distDir 文件的目标目录
     * @param NamerInterface $namer 文件的命名器
     */
    public function __construct(StorageInterface $storage,
                                string $distDir,
                                NamerInterface $namer)
    {
        $this->namer = $namer;
        $this->storage = $storage;
        $this->distDir = $distDir;
    }

    public function upload(UploadedFile $uploadedFile, string $fileName)
    {
        $this->storage->upload($uploadedFile, $this->distDir, $fileName);
    }

    public function getFileName(UploadedFile $file): string
    {
        return $this->namer->rename($file);
    }

}
```

3. 修改FileManagedType类，

```
class FileManagedType extends AbstractType
{
class FileManagedType extends AbstractType
{
//    /**
//     * @var ParameterBagInterface
//     */
//    private $parameterBag;
//
//    public function __construct(ParameterBagInterface $parameterBag)
//    {
//        $this->parameterBag = $parameterBag;
//    }

// 这里不需要上传目录属性了
```

```
//    /**
//     * @var string
//     */
//    private $uploadDir;
    /**
     * @var EventDispatcherInterface
     */
    private $eventDispatcher;

// 这里也不需要handler属性了
//    /**
//     * @var UploadHandler
//     */
//    private $uploadHandler;

    public function __construct(
//                                string $uploadDir,
                               EventDispatcherInterface $eventDispatcher
//,
//                                UploadHandler $uploadHandler
                               )
    {
//        $this->uploadDir = $uploadDir;
        $this->eventDispatcher = $eventDispatcher;
//        $this->uploadHandler = $uploadHandler;
    }
    ...

    // 注意这里转换为文件对象时 没有设置`fileName`属性，但是把Symfony封装的
`UploadedFile`对象设置到了我们的文件对象里
    private function fillFileObject(UploadedFile $file, File $fileObject)
    {
        $originName = $file->getClientOriginalName();

//        $fileName = pathinfo(htmlspecialchars($originName),
PATHINFO_FILENAME) . '-' . $file->getFilename() . '.' . $file-
>getClientOriginalExtension();

        $mimeType = $file->getMimeType();
        $filesize = $file->getSize();

        //这里需要修改，文件名是在Handler类里自动生成的，我们需要在Handler类里添加一
个方法，这里的代码不够优雅，我们后面会重构出更优雅的代码。
//        $this->uploadHandler->upload($file);
        //这里不设置新的文件名，在插入数据库之前再生成文件名
//        $fileName = $this->uploadHandler->getFileName($file);
//        $file->move($this->uploadDir, $fileName);

//        $fileObject->setName($fileName);
        $fileObject->setMimeType($mimeType);
        $fileObject->setOriginName($originName);
        $fileObject->setSize($filesize);

        //在这里把Symfony封装的文件对象设置到我们自己的文件对象里
```

```
            $fileObject->setUploadedFile($file);
        }

        ...
    }
```

4.修改FileManagedTypexml文件和Extension类，取消构造参数的对象的设置。

```
<?xml version="1.0" encoding="utf-8" ?>
<container xmlns="http://symfony.com/schema/dic/services"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://symfony.com/schema/dic/services
http://symfony.com/schema/dic/services/services-1.0.xsd">
    <services>
        <service id="teebb.upload.form.file_managed_type"
class="Teebb\UploadBundle\Form\FileManagedType">
<!--            <argument type="service" id="parameter_bag"/>-->
<!--            <argument type="string"/>-->
            <argument type="service" id="event_dispatcher"/>
<!--            <argument type="service"
id="Teebb\UploadBundle\Handler\UploadHandler"/>-->
            <tag name="form.type"/>
        </service>
    </services>
</container>
```

```
class TeebbUploadExtension extends Extension
{
    public function load(array $configs, ContainerBuilder $container)
    {
        ...

//        $fileManagedTypeDefinition = $container-
>getDefinition('teebb.upload.form.file_managed_type');
//        $fileManagedTypeDefinition->setArgument(0,
$config['upload_dir']);

        ...
    }
}
```

上传一个文件，测试一下自动生成的文件对象。 下节课，创建Doctrine事件订阅器，在SimpleFile 插入数据库之前，生成一个文件名，在插入数据库之后上传文件。