目前文件的上传流程里，文件的保存，文件的重命名，都在FileManagedType类中，这个表单类有些太复杂了。我想保证类功能的纯粹性，表单类只做和表单有关的事情。我们让FileManagedType表单处理文件的上传与回显。

文件的移动和重命名这些功能，我们交给一个单独的服务类，在src目录下新创建一个Handler目录，创建UploadHandler类，类里面现在只有一个方法upload()，有一个参数是Symfony自动封装的UploadedFile对象，为了更灵活，我们将参数类型设置为Symfony\Component\HttpFoundation\File\File。

在upload()方法里，把文件移动到上传目录就可以了。

这就好办了，我们定义Handler的构造方法，然后把目标目录添加到构造方法里，然后再重命名新的文件名，然后移动文件到目标目录。 这没有问题!

但是，我们在开发一套可复用的Bundle，灵活性很重要。

目前，所有的文件都是在本机保存的，直接使用move()方法就可以了，但是很有可能在实际项目里，你的文件会存在第三方的CDN或者对象存储平台。我们需要定义一个中间层，这个中间层提供一个统一的方法，然后不同的平台实现这个方法就可以完成文件的上传了。目前可以确定必须有的方法就是upload方法，

```
interface StorageInterface
{
    /**
     * 定义一个接口用于上传文件
     * @param UploadedFile $file
     * @param string $distDir 文件上传的目标目录
     * @param string $fileName 新的文件名称
     */
    public function upload(UploadedFile $file, string $distDir, string
$fileName): void;

}
```

目前我们只是简单的移动文件，可以创建一个类FileSystemStorage实现StorageInterface

```
class FileSystemStorage implements StorageInterface
{
    public function upload(UploadedFile $file, string $distDir, string
$fileName): void
    {
        $file->move($distDir, $fileName);
    }
}
```

回到Handler类，在构造方法里，添加用于存储文件的参数

```
    public function __construct(StorageInterface $storage,
                                string $distDir)
    {
        $this->storage = $storage;
        $this->distDir = $distDir;
    }
```

现在文件的命名方式是这样的，但是将来你的领导可能让你更换其他的命名方式，这就麻烦了。

我们可以创建一个用于文件命名的类，添加到Handler的构造方法里，当然为了灵活性，我们定义一个命名器的接品NamerInterface， 接口中有一个方法rename()，他接收一个Symfony封装的File对象做为参数。

> 在设计代码的时候，尽量让类做纯粹的工作，这样可以最大程度的解耦合代码。

```
interface NamerInterface
{
    public function rename(File $file): string;
}
```

为我们先前的重命名规则创建一个服务类，

```
class PhpNamer implements NamerInterface
{
    public function rename(UploadedFile $file): string
    {
        //优化下这里的代码
        $fileName = pathinfo(htmlspecialchars($file-
>getClientOriginalName()), PATHINFO_FILENAME) . '-' . $file-
>getFilename();

        if ($extension = $file->getClientOriginalExtension()) {
            $fileName = \sprintf('%s.%s', $fileName, $extension);
        }

        return $fileName;
    }
}
```

最终Handler类如下：

```
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)
    {
        $fileName = $this->namer->rename($uploadedFile);

        $this->storage->upload($uploadedFile, $this->distDir, $fileName);
    }
}
```

下节课内容： 我们需要把刚刚写的服务类添加到容器里，可以编写xml配置文件，但是Symfony还提供了通过代码把服务类添加到容器的方法。