第三方文件存储的平台有很多，国内阿里云、七牛云、腾讯云，国外亚马逊、谷歌都有自己的对象存储。 当文件向这些平台上传时，各个平台的接口不同，上传的方法也不同。 FlySystem提供了一个抽象层，使用统一的方法，可以操作文件的上传、删除、读取等等。 Symfony中要使用FlySystem，最简单的方法就是安装Bundle，这里有两个Bundle可以快速完成Symfony与FlySystem的集成。

```
thephpleague/flysystem-bundle  是Flysystem官方提供的一个包

1up-lab/OneupFlysystemBundle 是第三方的一个Bundle，我对这个Bundle很熟悉，所以课程里将会使用此Bundle与我们自己的Bundle集成
```

打开自己之前的项目：

```
oneup_flysystem:
    adapters:
        default_adapter:
            local:
                directory: '%kernel.cache_dir%/flysystem'
    filesystems:
        default_filesystem:
            adapter: default_adapter
            alias: League\Flysystem\Filesystem
```

安装完OneupFlysystemBundle进行上面的配置后，OneupFlysystemBundle会自动在容器中添加对应的服务类，然后我们就可以在项目里直接通过容器获取对应的服务类，或者在Controller方法里注入服务类对象，直接使用了。

服务类的id是：oneup_flysystem.default_filesystem_filesystem， default_filesystem是配置项的名称。

我们创建一个FlySystemStorage类，然后把对应的flysystem服务类作为构造参数注入FlySystemStorage类，然后实现上传方法。

先来安装1up-lab/OneupFlysystemBundle，我们仍然把文件上传到本地，但是使用使用Flysystem来完成上传。

```
teebb_upload:
#    upload_dir: '%kernel.project_dir%/public/teebb_upload'
#    namer:
#        service: Teebb\UploadBundle\Namer\HashNamer
#        options:
#            algorithm: sha1
#            length: 10
    handlers:
        # 为不同的类定义不同的Handler， 键值用于handler服务类的id，可以任意定义
        handler_name1:
            entity: App\Entity\SimpleFile # 文件数据的Entity全类名
```

```
            upload_dir: '%kernel.project_dir%/public/teebb_upload'  # 上传
的目标目录
            uri_prefix: 'teebb_upload' # 用于文件显示时添加前缀
#            storage:  # file_system 这里暂时这么写，这个名称对应
`FileSystemStorage`服务类
#                type: 'file_system'
#                service: 'serviceID'
#            namer:  # 和之前的一样就行
#                service: Teebb\UploadBundle\Namer\PhpNamer

        # 为不同的类定义不同的Handler， 键值用于handler服务类的id，可以任意定义
        handler_name2:
            entity: App\Entity\SimpleFile # 文件数据的Entity全类名
            upload_dir: 'string'  # 上传的目标目录
            uri_prefix: 'string' # 用于文件显示时添加前缀
            storage:  # file_system 这里暂时这么写，这个名称对应
`FileSystemStorage`服务类
                type: 'fly_system'
                service: 'oneup_flysystem.default_filesystem_filesystem'
# 这里可以是完整的id,也可以是flysystem的配置值`default_filesystem`，在Extension类
里添加判断就可以了
            namer:  # 和之前的一样就行
                service: Teebb\UploadBundle\Namer\HashNamer
                options:
                    algorithm: sha1
                    length: 10
```

当有两个handler处理相同的文件Entity时，后配置的handler会处理文件的上传。

创建FlySystemStorage服务类，使用FlySystem上传文件

```
class FlySystemStorage implements StorageInterface
{
    /**
     * @var Filesystem
     */
    private $filesystem;

    public function __construct(Filesystem $filesystem)
    {
        $this->filesystem = $filesystem;
    }

    public function upload(UploadedFile $file, string $distDir, string
$fileName): void
    {
        $stream = fopen($file->getPathname(), 'r');
        //这里以流的方式上传
        $result = $this->filesystem->writeStream(
            $distDir . \DIRECTORY_SEPARATOR . $fileName,
            $stream
        );
```

```
        if ($result === false) {
            throw new \Exception(sprintf('Could not write uploaded file
"%s"', $fileName));
        }

        if (is_resource($stream)) {
            fclose($stream);
        }
    }

}
```

```
<?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\UploadBundle\Storage\FileSystemStorage"
public="true"/>
        <service id="teebb.upload.storage.file_system_storage"
alias="Teebb\UploadBundle\Storage\FileSystemStorage"
                 public="true"/>

        <service id="Teebb\UploadBundle\Storage\FlySystemStorage"
public="true">
            <argument type="service"/>
        </service>
        <service id="teebb.upload.storage.fly_system_storage"
alias="Teebb\UploadBundle\Storage\FlySystemStorage"
                 public="true"/>
    </services>
</container>
```

修改Extension类，

```
class TeebbUploadExtension extends Extension
{
    ...
    private function registerHandlers(ContainerBuilder $container, array
$handlers)
    {
        $entityHandlerArray = [];
        foreach ($handlers as $handlerName => $values) {
            //storage
            if ($values['storage']['type'] == 'file_system') {
                $storageServiceId = $values['storage']['service'];
            } elseif ($values['storage']['type'] == 'fly_system') {
```

```
                $storageServiceId = FlySystemStorage::class;;
                //这里这样处理，如果service没有包含:`oneup_flysystem.` 那么拼接完
整的service id
                $flysystemServiceId = $values['storage']['service'];
                if (false == strpos($flysystemServiceId,
'oneup_flysystem.')) {
                    $flysystemServiceId = 'oneup_flysystem.' .
$flysystemServiceId . '_filesystem';
                }
                $flysystemServiceStorageDefinition = $container-
>getDefinition($storageServiceId);
                $flysystemServiceStorageDefinition->setArgument(0, new
Reference($flysystemServiceId));
            }

            //namer
            $namerServiceId = $values['namer']['service'];
            if ($options = $values['namer']['options']) {
                $namerServiceDefinition = $container-
>getDefinition($namerServiceId);
                $namerServiceDefinition->addMethodCall('configure',
[$options]);
            }

            $handlerDefinition = new Definition(UploadHandler::class);
            $handlerDefinition->setArgument(0, new
Reference($storageServiceId));
            $handlerDefinition->setArgument(1, $values['upload_dir']);
            $handlerDefinition->setArgument(2, new
Reference($namerServiceId));
            $handlerDefinition->setPublic(true);

            $handlerServiceId = sprintf('%s.%s', 'teebb.upload.handler',
$handlerName);
            $container->setDefinition($handlerServiceId,
$handlerDefinition);

            $entityHandlerArray[$values['entity']] = $handlerServiceId;
        }

        $container->setParameter('teebb.upload.handlers',
$entityHandlerArray);
    }
    ...
}
```

最后，上传一张图片，上传到了项目的cache\dev目录下。

表单的文件上传功能 就已经基本完成了，我们的课程主要功能也完成了，但是仍然遗留了很多问题，比如文件的删除、更新文件时旧文件是否要删除。这些功能，会做为Bundle的功能后续开发，如果你将来可能会用

到这个Bundle，你也可以在Github上Fork一下，然后提交PR来帮助我完善功能。

我们看Bundle的src目录下`Controller`目录、`Resources\public`目录仍然是空的。下节课，我们在Bundle里添加一个Controller，并且在`public`目录下添加一些资源文件。