API Platform可以快速的帮助我们创建API，我们要创建操作文章的API接口。我们打开Post类，在Post类前，我们只需要添加一个注解`ApiResource`就可以了。

```
#src/Entity/Post.php

/**
 * @ORM\Entity(repositoryClass=PostRepository::class)
 * @ORM\HasLifecycleCallbacks
 */
#[ApiResource]
class Post
{
    // ...
}
```

我们打开config目录，在config目录中，flex组件自动的为我们添加了api_platform.yaml配置文件。我们打开配置文件，API Platform会自动的检索Entity目录中的所有类，读取类前的注解，然后根据注解的设置生成API接口。

回到浏览器，我们刷新Api文档页面，API Platform自动的创建了6个操作文章资源的接口。前两个接口使用了一个路径，后4个接口使用了一个路径。这些接口使用了相对统一的路径。但是相同路径的接口，它们的方法并不是一样的。

我们看第一个接口，它使用GET方法，用于检索文章资源的集合。第2个接口它使用POST方法，用于创建一个文章资源。第3个接口它是用于操作单个资源的接口，这里使用了GET方法，用于检索一个文章的资源。下面一个使用了PUT方法，它用于对文章资源进行替换。第5个接口，这个接口使用了DELETE方法，用于删除文章资源。最后一个使用PATCH方法，它用于更新文章资源。

其中PUT方法和PATCH方法都是对文章资源的修改，但是两个方法有一些区别，PUT方法是对文章资源的整体替换。而PATCH方法是对文章资源的某个属性或者某些属性进行更新。这就是API Platform遵循REST规范设计的一套API。

它们有相对统一的路径，接口的方法也很重要，按照REST规范设计的接口，每个接口都有对应的功能。这样我们设计的接口被其他用户使用的话，其他用户就可以快速的上手了。

我们可以直接在文档页面上操作接口，我们查看第三个接口，我们来检索一篇文章。查看文章列表，我们获取ID为21的文章。点击，点击Try it out按钮，我们输入id参数。这里输入21点击Execute，点击之后。

curl库会发送一个get请求，请求地址是`/api/posts/21`，请求头有个accept参数，接收响应的格式为`application/ld+json`格式。在下节课，我们来学习这个格式。

我们查看一下响应的结果，响应码为400出错了，出现了循环引用。当我们获取某个文章数据时，我们同样的也要获取文章的作者。回到项目，我们打开User类，在User类中，我们要获取当前作者的所有文章，然后又再次的获取了一次文章。然后在文章中又再次的获取了作者，这样就出现了循环引用。我们在User类前添加API注解，回到浏览器，我们再次使用这个接口。

```
#src/Entity/User.php
```

```
/**
 * @ORM\Entity(repositoryClass=UserRepository::class)
 */
#[ApiResource]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    // ...
}
```

这次我们就看到了当前的文章信息，作者属性并没有展开，它获取到了一条字符串。在下节课我们来详细的讲解一下响应的结果格式。