我们在获取文章数据时，文章的作者数据它是一个IRI字符串。现在我们有个需求，我们先让文章作者的数据显示详细的作者信息而不是显示一条IRI。回到项目，在作者属性前，我们添加了`post:item:get`这个组。

我们可以在对应的User类中为需要显示的属性也设置这个组名，复制这一行。打开User类，我们获取作者的id，粘贴组名，点击OK，这里只保留post:item:get。然后获取作者的用户名，获取作者的角色。

回到浏览器，我们刷新文档页面，我们往下看，我们查看一下这个schema 展开。现在我们看到作者的值，它不再是`IRI`字符串了，它是另一个scheme。我们展开这个schema，在schema中就显示了作者的详细信息，就获取到了我们刚刚配置了组的属性的数据。

往上看，我们来测试一下接口，展开第三个接口。点击Try it out，文章的id我们输23，点击Execute。现在我们看到作者的数据进行了展开。这就是嵌到关系的显示，我们可以查看Api Platform文档。

点开API组件的文档，查看序列化过程文档。往下看，这里有嵌套关系的文档，我们点击，往下看。在文档中嵌套关系的显示也是使用了Groups注解，继续往下看。

在有些情况下，我们希望嵌套关系的数据必须使用`IRI`，我们可以添加另外一个注解`ApiProperty`，在`ApiProperty`中我们设置`readableLink`为`false`。这样具有嵌套关系的数据将会强制的显示为`IRI`字符串。

回到项目打开Post类，现在我们不希望获取作者的详细信息，可以添加`ApiProperty`注解。在注解的参数中，我们设置`readableLink`设置为false。回到文档，再次执行Execute，现在作者的数据就强制的显示为了`IRI`字符串。

回到项目，我们查看一下`ApiProperty`注解的源码，按着command键点击`ApiProperty`，在构造方法中，除了$readableLink还有一个$writableLink。

$readableLink就是当我们在获取资源数据时，可以控制嵌套关系的数据是显示IRI字符串还是显示详细的数据，那么对应的$writableLink，用来控制当我们在创建资源时提交的嵌套关系数，是使用IRI字符串还是使用展开的详细数据。

回到Post类，我们设置readableLink为true，将author数据进行展开，然后我们设置writableLink，也设置为true。

回到浏览器刷新文档，我们展开post.item.get这个schema。在scheme中作者的数据格式是另一个schema，当我们在创建文章资源时使用的是post.write这个schema，我们展开这个scheme，现在我们看到作者数据就不再是IRI字符串了。

打开User类，我们来为必要的属性来设置post:write组，我们为$username设置post:write，然后我们为用户的密码也设置post:write组。复制31行注解，粘贴，只保留post:write组。回到浏览器刷新文档页面，我们来尝试使用API接口来创建一篇文章，展开第二个接口，现在我们来看数据的格式。

当我们在创建文章时，需要文章的标题摘要，正文，还需要上传文章的作者，现在我们来看作者需要上传的数据，我们需要上传用户名和密码。点击Try it out，标题，这里我们输入hello 123，摘要也是hello 123，body也是hello 123，用户名这里我们输入hello，然后是密码，密码我们不能直接输入明文，我们需要对密码进行加密，我们需要提交一个加密过的密码。

回到项目，我们打开控制台，输入`symfony console`，查看所有可用的命令行。这里有个`security:hash-password`命令行，可以对用户的密码进行加密，我们复制命令行，输入`symfony console` 粘贴，后面我们添加密码123，现在我们看到加密过的密码是这个字符串。

复制字符串，回到浏览器，粘贴删除最后的空格。点击excute，我们看到出现了错误，这里提示，当我们在创建文章对象时，它也同时的创建了一个新的用户对象。

当EntityManager在保存文章对象时，新创建的用户对象并没有提前保存到数据库中，就出现了这个错误。回到项目，打开Post类，我们在$author属性上添加一个级联设置cascade。这里我们设置为数组添加persist。

这样当我们在保存文章对象时，新增的作者对象也会同时进行保存，回到项目再次点击Execute，现在我们看到我们新增了一篇文章。回到博客管理端。刷新文章列表，这就是我们刚刚新增的文章，但是我们看作者，我们在新增文章时创建了一个新的用户。

回到项目，通常我们使用API发表文章时，并不需要重新创建一个用户。在后面的课程中，我们将会使用另外一种方法来为提交的文章数据指定文章的作者，如果我们将readableLink和writableLink设置为false，那么我们在获取数据时和创建文章资源时，作者的数据将会强制使用IRI字符串。

回到浏览器刷新，我们查看文章的第二个接口，点击schema，作者被强制使用了IRI字符串，我们再来看文章类的第三个接口，查看schema，作者数据也是IRI字符串。

回到项目恢复一下，如果readableLink和writableLink都为true，那么我们可以省略这个注解，在使用API创建文章资源时。我们仍然让作者数据使用IRI字符串。打开User类，我们删除post:write这个组。

回到浏览器刷新文档，查看第二个接口，查看schema，现在作者的数据格式是IRI字符串，我们再来看第三个接口。我们看获取到的数据的scheme，作者数据仍然是详细的用户数据。

回到项目，如果我们要获取到文章的所有评论，我们可以添加组。Api Platform还提供了另外一个注解来生成子资源，添加注解`ApiSubresource`。回到浏览器刷新，往上看，我们看到在文章类中新增加了一个API，URL地址中有个参数就是文章的id。

我们测试一下，点击Try it out，id我们输入20点击Execute。现在我们就获取到当前文章的所有评论数据，在后面的课程我们将会使用另外一个方法来获取文章的所有评论数据。查看Api Platform文档，打开序列化过程的文档，当我们在获取资源数据或者创建资源数据时，分别进行了`normalize`和`denormalize`过程。

我们可以自定义Normalizer(没有自定义Denormalizer)，在序列化的过程中或者反序列化的过程中对数据进行更改。在下节课我们来自定义一个Normalizer。