我们想让正文的内容不进行转义，我们搜索Twig文档，我们打开twig.symfony.com这个页面。打开文档页面，在文档中Twig提供了很多过滤器，用来对Twig中的变量进行过滤，过滤器的底层代码其实是PHP方法，通过对Twig模板中的变量使用方法，可以对变量进行一些处理，我们想取消转义。这里有个raw过滤器。

我们点击raw，这里有个使用的示例，使用竖线加上raw过滤器，可以取消var变量的转义，我们回到项目代码，在body后添加竖线，添加raw过滤器，回到项目首页刷新。现在我们正文中的html字符串就取消了转义，回看我们项目的代码，代码第31行32行文章的创建时间和更新时间，它们后面使用了date过滤器来进行时间格式化显示，date需要传递一个参数，参数就是日期的显示格式。

回到项目页面，我们查看封面图像，封面图像在数据库中是以文件名称的方式进行存储的，还记得我们文件上传的位置吗？回到项目，我们打开public目录，在uploads文件夹下的images目录下，我们找到了上传的图片，文章的封面图像，我们想让它显示出来，而不是显示一个文件名称。我们修改代码添加img标签，src等于文件名，当然我们需要在文件名前添加uploads/images前缀。

回到浏览器刷新，现在我们的封面图像就显示出来了，我们还需要进行一些判断，如果封面图像为空的话，就不让它显示了，我们可以使用Twig的if关键字来进行处理，修改代码{% if post.postImage is not null %}，那么就让它显示图片，最后使用endif来结束条件判断。

回到页面刷新，现在就显示正常了，再次回到项目，img标签的src属性，我们使用了硬编码的方式来显示了文章的封面图像。这样写当然没有什么问题，但是Symfony为我们提供了另外一个Twig方法asset()。使用asset()方法对页面的图片、JS或者CSS文件进行包裹，可以很方便的进行资源文件的版本管理，另外如果你的图片资源保存在CDN上，你可以通过简单的配置在资源路径前自动的添加CDN前缀，我们现在对封面图片进行一下处理，我们按着command键鼠标移动到asset()方法中，asset()方法需要两个参数。

第一个参数是资源文件的路径，第二个参数是包的名称，我们把图片的路径传入第一个参数。剪切这段代码，粘贴，删除双引号，asset()是Twig方法。Twig方法中的参数是不需要再使用双括号的，现在所有的字符串都使用了单引号进行包裹，单引号包裹的话它整个都是字符串，post.postImage就不是变量了。我们需要把它拿出来，字符串和变量的连接，我们需要使用波浪线(~)。

回到页面刷新，和之前显示的一样，我们的图片资源已经成功的使用了asset()方法进行了包裹。

回到项目，在asset()方法中，我们使用硬编码的方式为图像资源添加了前缀，asset()方法有第二个参数包名，我们可以对Symfony项目进行资源包的配置，每个包可以指定不同的basePath，就是包的前缀，打开底部控制台，我们输入symfony console config:dump framework assets，我们使用这个命令行来查看一下symfony framework配置下assets配置项。

在assets配置项下有个packages配置，在packages下我们可以配置不同的包名，然后为不同的包设置base_path或者base_urls。这样在包下的资源将会自动的添加base_path或者base_urls，另外我们还可以对资源的版本进行控制。

关闭控制台，我们打开config目录，在packages目录下我们打开framework.yaml文件，在framework配置项下，我们将assets配置项复制过来，粘贴。assets配置项下，前几行是对项目的所有资源进行配置，我们需要灵活一些，删除这几行，在packages下，我们创建一个包名称，包的名称我们叫做file_upload，版本策略和版本控制，这里我们就不需要了删除base_urls我们也不需要。我们设置base_path为uploads/images文件夹。

```
#config/packages/framework.yaml
framework:
    assets:
```

```
    packages:

        # Prototype
        file_upload:
            base_path: 'uploads/images'
```

回到index.html.twig文件，我们删除硬编码，添加第二个参数，第二个参数就是我们的资源包名， 我们输入引号file_upload，回到浏览器刷新首页，封面图像资源正常的显示了。

```
#templates/post/index.html.twig
<td>{{ post.id }}</td>
<td>{{ post.title }}</td>
<td>{{ post.summary }}</td>
<td>{{ post.body }}</td>
<td>{{ post.body|raw }}</td>
<td>{{ post.status }}</td>
<td>{{ post.createdAt ? post.createdAt|date('Y-m-d H:i:s') : '' }}</td>
<td>{{ post.updatedAt ? post.updatedAt|date('Y-m-d H:i:s') : '' }}</td>
<td>{{ post.postImage }}</td>
<td>
    {% if post.postImage is not null %}
    <img src={{ asset(post.postImage, 'file_upload') }}>
    {% endif %}
</td>
<td>
    <a href="{{ path('post_show', {'id': post.id}) }}">show</a>
</td>
```

我们检查源码，asset()方法自动的在图片名称前添加了base_path路径。再次查看framework配置文件，在base_path这里我们依旧使用了硬编码，我们打开之前编写的管理端代码，在PostCrudController类中，我们为图片字段basePath和uploadDir也设置了硬编码。

我们可以修改一下，打开config目录下的services.yaml文件，在services.yaml文件中有个parameters配置，我们可以在parameters配置下添加一些参数，我们设置base_path，指向public目录中的uploads/images目录。然后我们设置upload_dir，它指向项目的public目录下的uploads/images目录。我们已经设置了base_path，我们可以复用base_path，复制base_path，使用百分号进行包裹，这样upload_dir就指向了public目录下的uploads/images目录。

```
#config/services.yaml
parameters:
    base_path: 'uploads/images'
    upload_dir: 'public/%base_path%'
```

我们再修改framework.yaml文件，我们让file_upload包下的base_path，它也等于我们刚刚设置的参数，使用双百分号包裹base_path，这个base_path就是刚刚我们在service.yaml文件中设置的参数。

```
#config/packages/framework.yaml
framework:
    assets:
        packages:
            # Prototype
            file_upload:
                base_path: '%base_path%'
```

回到PostCrudController.php文件，我们可以在Controller类中，用$this->getParameter()获取对应的参数，这里输入base_path，我们修改上传文件夹，这里使用$this->getParameter('upload_dir');。

```
class PostCrudController extends AbstractCrudController
{
    // ...

    public function configureFields(string $pageName): iterable
    {
        return [
            IdField::new('id'),
            IdField::new('id')->onlyOnIndex(),
            TextField::new('title'),
            TextEditorField::new('description'),
            ImageField::new('postImage')
                ->setBasePath($this->getParameter('base_path'))
                ->setUploadDir($this->getParameter('upload_dir'))
                ->setUploadedFileNamePattern('[slug]-[contenthash].
[extension]'),
            TextareaField::new('summary'),
            TextEditorField::new('body'),
            ChoiceField::new('status')
                ->setChoices(fn() => ['draft' => 'draft', 'published' =>
'published']),
            TimeField::new('createdAt')
                ->setFormat('Y-MM-dd HH:mm:ss')->onlyOnIndex(),
            TimeField::new('updatedAt')
                ->setFormat('Y-MM-dd HH:mm:ss')->onlyOnIndex(),
        ];
    }

    // ...
}
```

我们回到管理端上传一张图片，打开文章列表，我们编辑第5个文章，选择一张图片，这里选择001，保存更改，第5张图片就成功的显示了，我们的001文件成功的上传到了uploads/images目录中。

回看项目首页，刷新，我们第5篇文章，它的封面图像也成功的显示了出来，我们已经完成了博客系统的首页的显示。在下节课我们将完成博客详情页的展示。