在文章详情页，我们右键检查一下表单的代码。在form标签上我们添加一个属性来取消html前端验证。现在我们点击提交按钮，我们点击提交按钮后，表单的数据会转化为Comment对象，然后将Comment对象的数据插入到数据库中。但是现在Comment的作者、email地址，还有评论消息，它们都是null值，我们需要在后端对表单的数据进行验证。

Symfony支持表单验证，我们搜索`symfony form validation`，打开Validation文档，Validation组件提供了很多约束。我们使用这些约束可以对Entity类的各个属性添加验证。

往下看，文档中有表单类的约束，在表单类中我们可以在各个表单行中添加`constraints`设置。在`constraints`设置中，我们可以对当前的表单行添加约束。示例代码这里有个长度约束，最小长度为3。

回到项目，我们打开CommentType类，文章的作者，我们可以添加一个约束`constraints`，作者不能为空。我们可以新建一个NotNull()，`NotNull`类就是validation组件提供的一个约束。

我们查看`NotNull`类所在的文件夹，这里提供很多约束。我们继续添加，我们还可以限制表单行的文本长度，使用`new Length()`，限制最小长度为1，最大长度为20。

```
#src/Form/CommentType.php

class CommentType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array
$options)
    {
        $builder
            ->add('author', TextType::class, [
                'row_attr' => [
                    'class' => 'form-inline'
                ],
                'label_attr' => [
                    'class' => 'mr-3'
                ],
                'attr' => [
                    'class' => 'form-control-sm w-50'
                ],
                'required' => true,
                'constraints' => [
                    new NotNull(['message'=>'评论作者不能为空']),
                    new Length(['min'=> 1, 'max'=>20])
                ]
            ])
            ;
            // ...
}
```

回到浏览器，再次访问文章详情页，我们再次检查表单代码，在form标签上，我们添加`novalidate`属性。点击提交按钮，现在表单并没有提交，它会在表单行前显示一个表单行的错误，我们还可以修改错误提示。

回到代码，在`NotNull`构造方法中，我们可以添加一个设置，message评论作者不能为空。回到浏览器再次检查代码，添加`novalidate`属性，点击提交按钮。

现在错误的信息就变成了我们设置的信息，同样的我们可以对其他的表单行进行设置。回到代码，constraints，我们设置email表单行，它必须是个email格式，选择Constraints目录下的Email。然后同样的是NotNull，回到文章详情页，我们右键检查表单代码，在form标签上添加novalidate 。填写作者，填写邮件地址，但是邮件地址我们使用一个错误的邮件地址，点击提交按钮。现在提示我们不是一个有效邮件地址，这就是表单验证的一个使用方法。

我们继续查看文档，我们还可以在类的属性上添加表单验证，首先引入Contraints目录，然后在需要验证的属性上添加对应的约束注解。回到项目，我们注释表单行的约束代码。打开Comment类，首先我们引入约束文件夹，在作者属性上，我们添加注解NotNull()，最小长度为1，最大为30。

在$email属性上添加注解，NotNull()然后是Email()，回到浏览器，打开文章详情页，再次访问文章详情页，检查表单代码，添加novalidate属性。点击提交按钮，我们使用注解的方式添加的表单验证也生效了，Symfony还提供了另外一种更简单的使用注解的方式，我们打开Constraints目录，在Constraints目录中有一个`EnableAutoMapping`注解。

使用这个注解的话，Symfony将会对类中的所有属性都会进行自动验证，首先Symfony会读取各个属性的ORM注解，然后根据注解的类型和设置来添加自动验证。比如作者属性，它是文本类型，长度为30，就自动的添加长度限制，并且作者配置不能为空就会自动添加`NotNull`约束。

我们删除之前编写的注解，在Comment类前我们添加注解。回到浏览器，我们检查表单代码，在form标签上添加novalidate属性，点击提交按钮。现在我们看到作者和邮件地址都提示了错误信息，如果我们不想为某个属性添加自动验证，我们可以在属性前添加另外一个注解`DisableAutoMapping`。

```
#src/Entity/Comment.php

/**
 * @ORM\Entity(repositoryClass=CommentRepository::class)
 * @ORM\HasLifecycleCallbacks
 * @ORM\EntityListeners({"App\Listener\CommentListener"})
 * @Assert\EnableAutoMapping
 */
class Comment
{

    /**
     * @ORM\Column(type="string", length=30)
     * @Assert\DisableAutoMapping()
     */
    private $author;

    // ...

}
```

我们再次回到浏览器检查代码，在form标签上添加novalidate，再次点击提交按钮，现在作者的自动验证就取消了，为了更灵活，我们通常在各个属性上手动的添加验证注解。

为了让系统更安全，我们只允许用户在评论表单中上传图片类型的文件，Symfony提供的表单验证发生在表单提交之后和数据库插入之前，我们的文件表单行在表单提交之后会自动的转化为FileManaged对象，所以

Symfony内置的文件约束和图片约束`File`注解和`Image`注解，它们并不能对FileManaged对象进行验证。

我们就需要添加一个自定义的验证器了，在下节课我们将手动的创建一个验证器来限制文件上传的格式。