评论表单的样式有些丑，原因就是评论表单中没有使用到bootstrap的样式类，Symfony的表单原生是支持bootstrap样式的。回到项目，我们查看一下Twig组件可以使用到的一些配置，输入`symfony console config:dump`，可以查看各个组件的配置，然后我们输入参数twig。

Twig的表单是支持自定义样式的，我们查看一下Twig组件的源码，打开vendor目录，在symfony目录下找到`twig-bridge`，打开Resources目录，在views目录下打开Form目录。Form目录中，我们使用bootstrap4版本修改Twig样式。

打开config目录，我们找到twig.yaml文件，复制控制台上的`form_themes`。模板文件这里我们复制文件名。在views目录下，我们找到bootstrap_4_layout.html.twig文件。拷贝一下，粘贴。

回到浏览器刷新，现在评论表单的样式就进行了修改，但是这个评论表单样式还是有些奇怪，我们需要再次修改。我们让表单按列显示，回到项目，我们打开show.html.twig模板文件，在模板文件中使用了`row`样式类，让所有的表达输入行都显示到了一排。我们去掉这些样式类，删除html代码，再次回到浏览器刷新，现在表单行就按列显示了。

我们还需要再稍微调整一下这个样式，让样式更好看一些，比如说让表单行的标签和输入框成一行显示。我们打开CommentType类，在CommentType中，$builder通过add()方法添加了各个表单行，表单行的名称就是Comment类的属性。Form组件会根据属性它的注解，来自动的添加表单行的类型，这里我们手动的添加表单行的类型。

作者这里输入`TextType`，普通的文本框类型，我们搜索一下`TextType`。回到浏览器，搜索`symfony texttype`，`TextType`表单行可以有一些配置对表单行的样式进行修改，比如说添加表单行的html属性，或者修改表单行的标签，或者对整个表单行添加一个属性。

回到代码，`add()`方法的第三个参数就可以设置这些选项，首先我们让表单行的标签和文本框显示到一行，需要对整行进行显示设置，我们使用`row_attr`设置，在设置中我们添加一个样式类`class`，`class`的名称为`form-inline`。

回到浏览器刷新，现在标签和文本框就显示到了一行，我们让标签和输入框之间有一些距离。我们再次查看文档，这里有个`label_attr`设置，我们可以为`label`添加一个样式类，回到代码，输入`label_attr`。

再次回到浏览器刷新，这样就可以了，我们还可以让输入框显示的更美观一些，为输入框添加一些样式，输入框的样式直接使用`attr`设置。输入框添加一个样式类，刷新，现在样式显示的就可以了。

评论的作者是必填项，我们还可以对表单行添加设置，让作者框设置为必填项，回到代码，`required`设置设置为`true`。刷新。我们查看html代码，在input框中就添加了`required`属性。在`input`上一行，Symfony表单根据`required`设置，自动的在表单行的标签上添加了一个`required`样式类。

我们可以修改一下`required`样式类显示，为必填项添加一个星号，回到项目我们打开assets下的app.scss文件，我们在具有required样式类的label标签后方添加一个星号。星号颜色是红色。

```
#assets/styles/app.scss

form {
    label.required:after{
        content: '*';
        color: red;
    }
}
```

我们看到webpack已经编译成功了，回到浏览器刷新，在必填项的后方都添加了星号，为了节省时间，我直接将已经设置好的email和message样式直接粘贴过来，刷新。现在样式的显示就比较符合预期了。

我们再看评论列表的显示，评论列表现在还是硬编码，我们修改模板文件，打开`show.html.twig`文件。这里提供了两种评论的显示方式，一种是单独评论，一种是带有回复的嵌套评论显示。我们先注释掉带有嵌套的评论显示，我们打开Post类，在Post类中有$comments属性，$comments属性是个集合，我们使用`getComments()`方法来获取文章的所有评论。

回到`show.html.twig`文件，我们使用`post.comments`来获取文章的所有评论，按着command键把鼠标移动到`comments`上。它最终调用的是Post类的getComments()方法来获取所有的评论对象。我们再往前看，这里使用`post.body`来获取文章的正文，它最终调用的是getBody()方法。回到Post类，如果我们在Post类种自定义了一个get方法，那么在Twig模板中，我们可以直接使用`post.hello`来调用这个方法。

删除刚刚的代码，$comments属性它是一个集合，我们使用for循环来遍历这个集合。修改代码，使用`endfor`结束循环。这里是作者的名称，使用`comment.author`。下面是评论的正文，回到浏览器刷新，现在我们文章的评论就显示了出来。但是评论好像是按照正序来排列的，我们需要让评论列表按照id降序的方式来进行显示。

回到Post类，在$comments属性中我们可以使用ORM注解，`OrderBy`来调整评论对象的显示顺序，这里我们调整为id降序。回到浏览器刷新，现在评论列表，就按照id降序的方式进行显示了。我们手动提交一篇评论，评论提交成功了。

我们之前学习了功能测试。我们希望我们的系统能够自动化的进行测试。在下节课我们将使用功能测试代码来测试前台页面评论的提交