我们的评论数据一般情况下是不需要编辑的，但是我们现在研究的是文件的上传功能，因为后台我们使用了EasyAdminBundle，编辑评论的表单使用的是EasyAdminBundle生成的表单。如果要显示为我们需要的文件编辑表单，就需要自定义EasyAdmin Field，这不是我们课程的内容。

课程里，我将在Teebblog项目中创建一个编辑评论的页面，在这个页面，我们学习Symfony表单的内容。

在Teebblog项目`CommentController`类中添加一个新的controller方法：

```
//CommentController.php

    #[Route('/comments/{id}', name: 'edit_comment')]
    public function editComment(Request $request, Comment $comment,
EntityManagerInterface $em): Response
    {
        $commentForm = $this->createForm(CommentType::class, $comment);


        return $this->render('comment/edit_comment.html.twig', [
            'comment_form' => $commentForm->createView(),
        ]);
    }
```

页面模板文件

```
{#comment/edit_comment.html.twig#}

{% extends 'base.html.twig' %}
{% import 'marco.html.twig' as macros %}
{% block title %}Edit Comment{% endblock %}

{% block content %}
    {# read and display all flash messages #}
    {% for label, messages in app.flashes %}
        {% for message in messages %}
            <div class="alert-{{ label }} mt-4 alert">
                {{ message }}
            </div>
        {% endfor %}
    {% endfor %}

    <!-- Comments Form -->
    {% if is_granted('ROLE_ADMIN') %}
        <div class="card my-4">
            <h5 class="card-header">Edit Comment:</h5>
            <div class="card-body">
                {#    {{ form(comment_form) }}#}
                {{ form_start(comment_form) }}
                <div class="my-custom-class-for-errors">
                    {{ form_errors(comment_form) }}
```

```
                </div>

                {{ form_row(comment_form.author) }}
                {{ form_row(comment_form.email) }}
                {{ form_row(comment_form.message) }}
                {#            <button type="submit">提交</button>#}
                {{ form_end(comment_form) }}
            </div>
        </div>
    {% else %}
        {{ 'login_for_edit_comment'|trans({'%login_path%':
path('app_login')})|raw }}
    {% endif %}

{% endblock %}
```

添加编辑评论链接

```
{#macro.html.twig#}

{% macro show_comments(comment, post) %}
    {% import _self as macro %}
    <div class="media mb-4">
        <img class="d-flex mr-3 rounded-circle"
src="http://placehold.it/50x50" alt="">
        <div class="media-body">
            <h5 class="mt-0">{{ comment.author }}</h5>
            {{ comment.message }}
            <div>
                {% for file in comment.files %}
                    <a href="{{ asset(file.name, 'simple_file_upload') }}"
data-lightbox="{{ comment.id }}">
                        <img src="{{ asset(file.name,
'simple_file_upload')|imagine_filter('my_thumb') }}" alt="{{
file.originName }}"
                             class="img-thumbnail">
                    </a>
                {% endfor %}
            </div>

            <button class="btn btn-sm btn-link js-reply-comment-btn" data-
post-id="{{ post.id }}"
                    data-parent-id="{{ comment.id }}">回复
            </button>
            <a class="btn btn-sm btn-link js-reply-comment-btn"
               href="{{ path('edit_comment', {id: comment.id}) }}">编辑</a>

            {% for childComment in comment.children %}
                {{ macro.show_comments(childComment, post) }}
            {% endfor %}
        </div>
```

```
        </div>
    {% endmacro %}
```

随便编辑一条评论，出错了，现在上传的文件类型是SimpleFile，但是表单行显示时，需要一个File类型。提示我们需要创建一个View Transformer把`SimpleFile`转换为`File`类型。

`FileManagedType`表单行是从`FileType`继承来的，查看`FileType`源码，`configureOptions()`方法中定义了默认的数据类型为 File 类型，所以会提示出错了。

```
    public function getBlockPrefix()
    {
        return 'file';
    }
```

`getBlockPrefix`方法返回的值是表单widget的名称前缀，我们找到表单的twig模板，先找到`file_widget`区块，再`file_widget`区块中，重要的是`{{- block('form_widget_simple') -}}`区块，继续找，最终在`form_div_layout.html.twig`模板中找到了区块的定义，是一个普通的文件上传input。

下节课，修改表单的代码，修复错误提示。