在回复评论的controller方法中，我们设置了子评论对象的post属性，这样文章对象和子评论之间也存在了关联关系。在show.html.twig文件中，所有的子评论对象也都作为了顶级评论进行了显示。

我们需要修改一下Comment类，我们设置评论的$post属性可以为空。如果为空的话，那么这个评论就是一条子评论，文章下方的评论框提交的评论对象都设置了post属性，这些评论对象都可以认为是顶级的评论对象。

在show.html.twig文件中，我们只获取文章对象所有顶级的评论，然后使用macro的show_comments()方法来嵌套显示所有的评论。我们修改Comment类，在JoinColumn注解后添加一个nullable设置为true。

```
#src/Entity/Comment.php
class Comment
{
    // ...

    /**
     * @ORM\ManyToOne(targetEntity=Post::class, inversedBy="comments")
     * @ORM\JoinColumn(nullable=false, nullable=true)
     */
    private $post;

    // ...
}
```

打开底部的控制台，输入symfony console make:migration来创建一个数据库更改。我们查看一下命令行生成的文件，文件中修改post_id它的默认值可以为空，执行这个数据库更改，yes。

我们回到CommentController方法中，注释掉34行setPost()方法。回到浏览器，打开博客系统的管理端，打开评论列表，我们从后往前删除后三条子评论。回到文章详情页刷新，在第三条评论上，我们点击回复按钮，添加一条评论。评论信息输入hello 1，提交一下。再次查看评论列表，我们在hello 1下方再次点击回复按钮，再次提交。现在评论列表显示的就没有问题了。

我们来解决一下，回复按钮，不断追加评论框的bug。我们在回复按钮上右键点击检查，在button后方添加了两个reply-comment-card。我们可以通过reply-comment-card的数量来进行判断，如果button后面没有replay-comment-card元素，我们发送Ajax请求来追加表单。如果有的话，就不发送Ajax请求了，回到项目修改app.js文件，在ajax()方法上方我们添加一个判断。

如果$(this)按钮的后方没有找到，div.reply-comment-card.length等于0，那么就发送Ajax请求。否则的话就不发送Ajax请求。回到浏览器等待刷新，我点击回复按钮，再次点击回复按钮现在就起作用了，多次点击回复按钮，只显示一个评论框。

```
#assets/app.js
$('button.js-reply-comment-btn').on('click', function (element) {
    let postId = $(this).data('post-id');
    let parentId = $(this).data('parent-id');

    if($(this).nextAll('div.reply-comment-card').length === 0){
```

```
        $.ajax({
            url: Routing.generate('reply_comment', {post_id: postId,
comment_id: parentId}),
            type: 'POST'
        }).done(function (response){
            $(element.target).after(response)
        }).fail(function (jqXHR){

        })
    }
})
```

我们再次回到管理端，刷新评论列表，hello1和hello2是第三条评论的子评论，现在我们删除第三条评论，删除时就出错了。因为第三条评论它下方有子评论，删除父评论时子评论仍然引用了父评论的数据，就出现了删除的bug。

我们可以设置子评论的级联操作，当删除父评论时，子评论也一并删除。回到项目，我们打开Comment类，在$children属性上，按着command键鼠标移动到OneToMany上，点击OneTwoMany它提供了一个$cascade属性。

我们搜索doctrine cascade，文档中提示我们使用cascade可以进行级联操作。比如保存删除，我们想在删除父评论时同时删除子评论，我们可以设置cascade属性为remove。回到Comment类，cascade属性它是数组，我们需要使用大括号来包裹一下，我们再次回到浏览器打开管理端。

```
#src/Entity/Comment.php
class Comment
{
    // ...

    /**
     * @ORM\OneToMany(targetEntity=Comment::class, mappedBy="parent",
cascade={"remove"})
     */
    private $children;

    // ...
}
```

回到评论列表，我们删除第三条评论，这时第三条评论和它所有的子评论也都一块删除了。再回到文章详情页刷新，现在我们的回复按钮可以无限的添加子评论。我们想对，我们想对文章的评论添加一个层级限制，当达到某个层级后，它就不允许再添加子评论了。

在下一节我们将解决评论的层级限制。