我们删除$children属性的级联设置，现在我们来考虑一下，当删除父级评论时会出现的一些问题。当删除父级评论时，在数据库中子级评论对象的parent列，它指向的是父级对象的id。删除父级评论时，子级评论对象对父级对象还有引用，这时就会触发外键约束，删除就会失败。

回到浏览器，刷新评论列表，这里有一条评论，我们在这条评论下面增加一条评论。回到详情页刷新，点击回复按钮，点击提交。刷新评论列表，我们来删除id为103的评论，触发了外键约束。我们可以在数据库级别来添加一个级联设置，当删除父评论时，把子评论的parent_id列设置为null。

回到代码，在$parent属性前，我们添加注解，`ORM\JoinColumn`，我们让parent列可以为空，添加`onDelete`设置，`onDelete`的值设置为`SET NULL`。这个设置是在数据库级别的，我们需要创建一个migration文件。

打开控制台，输入`symfony console make:migration`，查看生成的migration文件，SQL语句中添加了`ON DELETE SET NULL`关键字。执行数据库的更改，复制命令行粘贴回车。

我们回到浏览器再次刷新，现在父评论就删除了，子级评论就保留了下来，现在子级评论就变成了顶级评论，但是子级评论并没有post属性。它的level属性还是之前的数据，我们就需要在CommentListener中进行一些修改。

打开CommentListener，在CommentListener类中我们添加一个方法。回到浏览器，我们打开`doctrine events`文档，在文档中我们找到UserListener这个案例，在案例中找到`preRemoveHandler()`这个方法。这个方法还是有两个参数，第一个参数是要删除的对象，第二个参数是生命周期事件对象。

我们复制这两行代码，回到项目粘贴，修改一下代码。User对象改为Comment对象。在CommentListener类中，我们新增一个方法，方法的名称叫做`fixCommentTitleAndLevelWhenRemove()`。需要传一个参数参数我们传入$comment对象，我们在方法中获取当前评论对象的所有子评论对象，对所有子评论对象进行遍历。

如果当前评论对象它是顶级评论对象，那么我们就需要将当前评论对象的post属性设置到子评论中，`if($post = $comment->getPost())`，我们将获取到的$post对象设置到子评论中，`$child->setPost($post);`，然后我们调整一下子评论对象的level。将子评论对象的level设置为父级评论对象的$level属性`$child->setLevel($comment->getLevel())`。

我们还需要设置一下子评论的$parent属性，如果当前评论对象不是顶级评论，那么它就有个$parent属性，我们需要将$parent属性设置到子评论中 `if($parent = $comment->getParent())``$child->setParent($parent);`参数为$parent对象，最后来判断当前的子级评论对象是否还有下一级评论对象。

如果有的话，再次调用这个方法来实现一个递归调用。调用`$this->fixCommentTitleAndLevelWhenRemove($child);`参数传入$child，最后在preRemoveHandler()方法。我们调用这个方法，参数传入$comment对象。

```
#src/Listener/CommentListener.php

class CommentListener
{
    /** @ORM\PostLoad */
    public function postLoadHandler(Comment $comment, LifecycleEventArgs
$event)
    {
```

```
            $rootComment = $this->getParent($comment);
            $comment->setPost($rootComment->getPost());
        }
        public function getParent(Comment $comment)
        {
            $parent = $comment->getParent();

            if ($parent) {
                return $this->getParent($parent);
            }

            return $comment;
        }

        /** @ORM\PreRemove */
        public function preRemoveHandler(Comment $comment, LifecycleEventArgs
    $event)
        {
            $this->fixCommentTitleAndLevelWhenRemove($comment);
        }

        public function fixCommentTitleAndLevelWhenRemove(Comment $comment)
        {
            $children = $comment->getChildren();
            foreach ($children as $child) {
                if ($post = $comment->getPost()) {
                    $child->setPost($post);
                }
                $child->setLevel($comment->getLevel());
                if ($parent = $comment->getParent()) {
                    $child->setParent($parent);
                }

                $this->fixCommentTitleAndLevelWhenRemove($child);
            }
        }
    }
```

回到浏览器，我们再次访问文章详情页，我们在杜依琳这条评论中添加一个子评论。点击回复按钮，点击提交，我们在第一条子评论中再添加下级评论，点击提交。回到管理端刷新评论列表，我们先来删除杜依琳这条评论，现在我们看到删除杜依琳评论之后，我们新增的两条评论，他们都有post属性。

本节课和上节课重点在讲解doctrine生命周期事件的处理，以及它们的使用场景，我们的代码会造成一些bug，在本套课程中这些bug作为边缘情况，我们不会处理这些bug。

回到文章详情页，当我们在提交评论时，html代码会进行一次前端验证。比如验证email地址格式是否正确，我们还需要进行后端验证。Symfony的表单系统提供了表单验证的功能，在下节课我们开始学习表单验证。