当我们点击回复按钮时，回复评论表单是通过Ajax请求追加到页面上的，所以jquery并不能查找到回复表单的DOM元素。同样的它也查找不到回复表单中的添加文件按钮，我们可以在添加文件按钮中添加一个属性`onclick`。

`onclick`指向一个方法，方法内部来处理按钮的事件。回到app.js文件，我们在window域添加一个方法，方法名称叫做`addFileUpload`。function传递一个参数element，方法内部就是按钮的事件处理，我们复制前面的代码，粘贴。

```
window.addFileUpload = function (element) {
        //查询input-wrapper DOM
        var fileInputWrapper = $(element.target).closest('fieldset.form-
group').find('div.input-row-wrapper');
        //获取当前input行计数
        var inputCount = fileInputWrapper.children().length;
        //获取input prototype并进行序号修改
        var inputCode = fileInputWrapper.data('prototype');
        inputCode = inputCode.replace(/__name__/g, inputCount);

        fileInputWrapper.append(inputCode);
    })
})
```

打开templates目录，我们打开form目录。打开模板文件，在button标签这里我们添加`onclick`属性，`onclick`属性它的值就是我们刚刚编写的方法。需要传一个参数我们输入this。 回到浏览器刷新，再次点击回复按钮，我们点击添加文件，添加文件按钮并没有生效。在添加文件上点击右键检查，再次点击添加文件按钮，这里报错了。

我们点击这个错误信息，它提示replace()方法错了，app.js67行。回到项目，replace()方法出错的话，那肯定是没有找到对应的元素，我们看61行，它在查找父元素时使用`element.target`，我们的参数element现在就是button标签，并不需要使用target属性删除target，我们需要把之前的按钮事件处理代码注释掉，注释掉42行代码。

```
window.addFileUpload = function (element) {
        //查询input-wrapper DOM
        var fileInputWrapper = $(element).closest('fieldset.form-
group').find('div.input-row-wrapper');
        //获取当前input行计数
        var inputCount = fileInputWrapper.children().length;
        //获取input prototype并进行序号修改
        var inputCode = fileInputWrapper.data('prototype');
        inputCode = inputCode.replace(/__name__/g, inputCount);

        fileInputWrapper.append(inputCode);
    })
})
```

回到浏览器刷新，点击添加文件，现在就可以了。点击回复按钮，再次点击添加文件，同样的没有问题。

我们添加一个子评论，点击提交，我刚刚提交的评论已经显示了，我们打开数据库客户端查看一下数据库表中的数据，查询一下comments_files表。我们刚刚上传了两张图片，这里只显示了一条数据，我们刚刚上传的文件并没有成功的插入数据库，回复评论表单的提交地址是在CommentController的方法中。我们打开CommentController，在CommentController的replyComment()方法中，我们并没有处理文件的上传。

我们复制PostController中的代码，回到CommentController，在表单提交这里进行一下处理，同样的把表单的数据放在文件上传之前，在评论数据插入到数据库之后，我们来添加一个flash消息。

```
#src/Controller/CommentController.php

class CommentController extends AbstractController
{
    #[
        Route('/post/{post_id}/comment/{comment_id}/reply',options:
['expose' => true], name: 'reply_comment'),
        ParamConverter('post', options: ['id' => 'post_id']),
        ParamConverter('parentComment', options: ['id' => 'comment_id']),
    ]
    public function replyComment(Request $request, Post $post, Comment
$parentComment, EntityManagerInterface $em): Response
    {
        $maxLevel = $this->getParameter('max_comment_level');
        if ($parentComment->getLevel()==$maxLevel){
            return new Response('<p class="max-level-info">当前评论已达到最大
层级，不允许添加子评论。😄 </p>');
        }
        $replyComment = $this->createForm(CommentType::class, null, [
            'action' => $request->getUri()
        ]);
        $replyComment->handleRequest($request);
        if ($replyComment->isSubmitted() && $replyComment->isValid()){
            /**@var Comment $data**/
            $data = $replyComment->getData();

            $files = $request->files->all();
            /**@var UploadedFile $file**/
            foreach ($files['comment']['files'] as $file){
                $originName = $file->getClientOriginalName();
                $fileName = pathinfo(htmlspecialchars($originName),
PATHINFO_FILENAME) . '-' . $file->getFilename() . '.' . $file-
>getClientOriginalExtension();
                $uploadPath = $this->getParameter('base_path');
                $mimeType = $file->getMimeType();
                $filesize = $file->getSize();

                $file->move($uploadPath, $fileName);

                $fileManaged = new FileManaged();
                $fileManaged->setOriginName($originName);
                $fileManaged->setFileName($fileName);
                $fileManaged->setMimeType($mimeType);
                $fileManaged->setPath($uploadPath . '/' . $fileName);
```

```
                $fileManaged->setFileSize($filesize);

                $data->addFile($fileManaged);
            }

            $data->setParent($parentComment);
            $data->setLevel($parentComment->getLevel() + 1);
//            $data->setPost($post);
            $em->persist($data);
            $em->flush();

            $this->addFlash('success', '您的评论已成功提交！');

            return $this->redirectToRoute('post_show', ['id1' => $post-
>getId()]);
        }
}
```

回到浏览器，我们再次提交一个子评论，这次我们上传008文件，009。点击提交按钮，它提示评论已经成功提交了。我们再次查看数据库，查询一下，现在数据库中新增了两条数据。文件为id4，文件为id5。我们查看一下file表，文件id4是008，文件id5是009。现在子评论的提交就没有问题了。

回到项目，我们看到在CommentController和PostController中有大量的重复代码。我们打开ConmentType类，Symfony支持自定义表单类型，在files表单行中我们可以自定义一个表单类型，在自定义的表单类型中来处理上传的文件，来将上传的文件自动的转化为FileManaged对象。这样就不用在Controller中手动的进行转换了。

在下节课我们来自定义一个表单类型。