我们查看一下错误栈代码，在PostController show()方法中，评论表单对象会处理表单的请求，然后进行表单数据的转换。当表单处理到files属性时，发现表单提交的数据类型并不是FileManaged类型，所以就报了异常。

我们可以让表单不对files表单行提交的数据进行自动转换，我们可以在controller方法中通过$request对象来获取表单提交的文件，然后手动的将文件转换为FileManaged对象，再将FileManaged对象设置到评论对象的files属性中。

回到项目，打开CommentType类，在files表单行中我们添加一个设置，我们设置mapped选项为false。现在files表单行就不会自动映射到Comment类的$files属性上了。

打开PostController，在show()方法中我们查看一下表单提交的数据，输入dd($data);，回到浏览器刷新，我们再次提交表单，你现在表单的files属性，它是一个空的ArrayCollection对象。

我们检查一下表单提交时的$request对象。回到浏览器刷新提交表单，我们打开files属性。现在files属性中，我们看到了表单提交的文件对象，我们可以通过$request对象来获取表单提交的文件对象，然后转换为FileManaged对象。

回到代码，在表单提交的代码中，我们使用$request对象的files属性来获取所有上传的文件对象。定义一个返回值$files，我们查看一下$files变量，回到浏览器，再次刷新，$files变量是个数组，我们需要遍历获取数组中的所有文件。添加一个for循环，foreach $files。第一个键是comment，第二个键是files，$file对象。

我们添加一个注释，现在$file对象的类型是UploadedFile，再回到浏览器，我们展开UploadedFile，可以通过UploadedFile对象的方法来获取上传的文件信息，然后将文件移动到指定的目录中。回到代码，为了节省时间，我这里就直接粘贴代码了。

```
#src/Controller/PostController.php

class PostController extends AbstractController
{
    // ...

    #[Route('/post/{id1}', name: 'post_show', methods: ['GET', 'POST'])]
    #[ParamConverter('post', options: ['id' => 'id1'])]
    public function show(Request $request, Post $post,
EntityManagerInterface $entityManager,
                         PaginatorInterface $paginator, CommentRepository
$commentRepository): Response
    {
        $commentForm = $this->createForm(CommentType::class);
        $commentForm->handleRequest($request);
        if ($commentForm->isSubmitted() && $commentForm->isValid()) {
            if ($commentForm->get('submit')->isClicked()) {
                /**@var Comment $data * */
                $data = $commentForm->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);
                }

//                dd($data);
                $data->setPost($post);
                $entityManager->persist($data);
                $entityManager->flush();
            }
            $this->addFlash('success', '您的评论已成功提交！');
        }
        $query = $commentRepository->getPaginationQuery($post);
        $pagination = $paginator->paginate(
            $query, /* query NOT result */
            $request->query->getInt('page', 1), /*page number*/
            10 /*limit per page*/
        );
        return $this->render('post/show.html.twig', [
            'post' => $post,
            'pagination' => $pagination,
            'comment_form' => $commentForm->createView()
        ]);
    }

}
```

首先我们获取文件的原始名称，然后根据原始名称我们生成一个新的文件名称，然后我们获取之前设置的参数base_path，将上传的文件移动到uploads/images目录中。第61行，我们使用new关键字创建了一个FileManaged对象，然后我们为FileManaged对象设置所有的属性，最后我们将FileManaged对象添加到评论对象中。这里我们调整一下代码的顺序，使用$data->addFile()方法，将FileManaged对象添加到files属性中。

回到浏览器再次刷新，继续，现在我们查看files属性，files属性中，现在就有一个FileManaged对象。回到Controller，我们注释75行代码，后面EntityManager对象会将评论对象保存到数据库中，回到浏览器再次刷新，现在提示出错了。

回到项目，评论对象和文件对象之间存在着关联关系，但是我们只把评论对象保存到了数据库中，文件对象并没有保存到数据库中，所以就抛出了这个异常。我们查看一下异常的信息，我们可以添加级联设置，这样

在存储评论对象的数据时，也可以一块把文件对象的数据插入到数据库中。

回到代码，我们打开Comment类，在$files属性的注解中，我们添加cascade设置。cascade设置的值是数组，我们使用大括号，添加persist，回到浏览器再次刷新。

```
#src/Entity/Comment.php
/**
 * @ORM\Entity(repositoryClass=CommentRepository::class)
 */
class Comment
{
    // ...

    /**
     * @ORM\ManyToMany(targetEntity=FileManaged::class, cascade=
{"persist", "remove"})
     * @ORM\JoinTable(name="comments_files",
     *      joinColumns={@ORM\JoinColumn(name="comment_id",
referencedColumnName="id")},
     *      inverseJoinColumns={@ORM\JoinColumn(name="file_id",
referencedColumnName="id", unique=true)}
     *      )
     */
    private $files;

    // ...
}
```

现在评论就提交成功了，我们查看一下数据库，在comment表中，我们新插入了一条评论，然后我们查看文件表，文件表中也成功的插入了一条数据，关联表中也成功的插入了数据。

回到文章详情页，我们向下拉，我们在评论列表中点击回复按钮，点击回复表单中的添加文件按钮。现在添加文件按钮无效了，在下节课我们来解决这个bug。