打开控制台，我们创建CommentFixtures，类名输入CommentFixtures，在load()方法中我们也添加一个for循环，在循环体中我们使用new关键字来创建Comment对象。

设置Comment对象的作者，我们需要创建一个$faker对象，在构造方法中使用Factory的create()方法来创建$faker对象。create()方法也可以传一个参数，可以为不同国家生成不同语言的数据，这里我们使用中文地区。

作者我们使用`$this->faker->name()`方法来创建一个姓名，然后设置Comment的邮箱地址，setEmail()同样使用`$this->faker->email()`方法来生成邮件地址。然后设置Comment对象的消息正文setMessage()，使用`$this->faker->paragraph()`方法来生成正文。

现在我们想让这些假的数据中有一些子评论，我们可以在for循环的上面添加一个数组。每次创建一个对象后，我们把对象添加到数组中，然后再从数组中随机的挑选一个Comment对象，当做父级评论对象。

如果$i大于0，并且`$this->faker->boolean()`方法产生了一个真值，我们从数组中挑选一个Comment对象作为$parentComment，我们可以使用`$this->faker->randomElement()`方法，随机的从数组中挑选一个元素，数组我们这里输入$commentArray，现在我们设置评论对象的父级评论为$parentComment。

然后我们设置一下子评论的层级setLevel()，它等于`$parentComment->getLevel() + 1`。如果当前评论它不是一个子级评论的话，我们需要设置评论的文章对象，else，$comment->setPost()，我们需要设置到上节课创建的最后一篇文章上。

我们可以在PostFixtures类中添加一个常量，`LAST_POST`等于last_post。然后在load()方法中，我们添加一个变量，$lastPost等于null。如果循环到最后一篇文章，我们让$lastPost变量为最后一篇文章的数据，$lastPost等于$post。

在循环体外我们可以使用一个方法，将$lastPost对象设置为可以供其他Fixtures共享使用的数据。使用`$this->addReference()`，第一个参数我们需要传递一个字符串，第二个参数就要传递一个对象，字符串我们传入`self::LAST_POST`。第二个参数传入$lastPost。

```
#src/DataFixtures/PostFixtures.php

class PostFixtures extends Fixture
{
    public const LAST_POST = 'last_post';

    public function __construct(PostFactory $postFactory)
    {
        $this->postFactory = $postFactory;
        $this->faker=Factory::create();
        $this->faker = Factory::create();
    }

    public function load(ObjectManager $manager)
    {
        $lastPost = null;
        for ($i = 0; $i < 20; $i++) {
            $post = $this->postFactory->create($this->faker->sentence(),
$this->faker->paragraph());
            if ($this->faker->boolean()){
```

```
            if ($this->faker->boolean()) {
                $post->setStatus('published');
            }

            $image = '00'.$this->faker->randomDigit().'.jpg';
            $image = '00' . $this->faker->randomDigit() . '.jpg';
            $post->setPostImage($image);

            if ($i == 19){
            if ($i == 19) {
                $post->setStatus('published');
                $lastPost=$post;
            }

            $manager->persist($post);
        }

        $this->addReference(self::LAST_POST, $lastPost);

        $manager->flush();
    }
}
```

这样在CommentFixtures中，我们可以通过$this->getReference()来获取最后一篇文章，参数的名称就是PostFixtures类下的常量LAST_POST，它的返回值是个Post对象。我们设置评论的post属性为$lastPost对象，当然为了提升一下性能，我们可以将32行代码添加到for循环上面。最后我们使用$manager的persist()方法保存一下$comment数据。

打开控制台，我们复制doctrine:fixtures:load命令行，来加载这些假数据，粘贴回车。它提示我们数据库将会被清除，我们yes。现在它提示出错了，查看一下错误提示，还是在(清空数据)删除父级评论时触发了外键约束，在后面课程我们会彻底解决这个问题。

我们回到管理端，打开评论列表，手动的删除一下所有的评论，再次执行命令行，输入yes。在加载数据时出错了，提示没有找到last_post这个数据引用。我们查看一下Fixtures目录，Fixtures的加载，它是按照文件名称的先后顺序进行加载的。所以它会先执行AppFixtures，然后是CommentFixtures，最后是PostFixtures。

我们需要先加载文章再加载评论，回到CommentFixtures类，我们让CommentFixtures类实现一个接口DependentFixtureInterface。把鼠标移动到这条红线上，然后我们点击添加方法，点击OK，现在在CommentFixtures中添加了一个getDependences()方法。我们在getDependences()方法中返回当前类所依赖的所有Fixtures，return一个数组，数组中是依赖的Fixtures全类名。

```
#src/DataFixtures/CommentFixtures.php

class CommentFixtures extends Fixture implements DependentFixtureInterface
{
    private $faker;
    public function __construct()
    {
```

```
        $this->faker = Factory::create('zh_CN');
    }

    public function load(ObjectManager $manager)
    {
        $lastPost = $this->getReference(PostFixtures::LAST_POST);

        $commentArray = [];
        for ($i = 0; $i < 50; $i++) {
            $comment = new Comment();
            $comment->setAuthor($this->faker->name());
            $comment->setEmail($this->faker->email());
            $comment->setMessage($this->faker->paragraph());

            if ($i > 0 && $this->faker->boolean()){
                $parentComment = $this->faker-
>randomElement($commentArray);
                $comment->setParent($parentComment);
                $comment->setLevel($parentComment->getLevel() + 1);
            }else{
                $comment->setPost($lastPost);
            }

            $commentArray[] = $comment;

            $manager->persist($comment);
        }

        $manager->flush();
    }

    public function getDependencies()
    {
        return [
            PostFixtures::class
        ];
    }
}
```

打开控制台，再次执行命令，yes。现在命令就没有出错了，我们打开浏览器查看一下博客的首页，通过Fixtures添加了很多文章。

我们打开最新的一篇文章，在文章的下面就生成了一些评论信息。在下节课我们将完成首页文章列表的分页显示。