Comment类和Post类都有createdAt和updatedAt属性，我们可以使用doctrine的生命周期事件在将对象保存到数据库之前，自动的为createdAt属性和updatedAt属性进行初始化。这两个属性在类中都有对应的get()方法和set()方法。

我们查看一下对应的方法，我们来修改一下setCreatedAt()方法，我们直接使用new \DateTime()，来为createdAt属性赋值。同样的在setUpdatedAt()方法中，我们一样使用new \DateTime()，来为updatedAt属性赋值。

查看文档，我们需要在类前添加一个注解。回到Post类，我们使用ORM\HasLifecycleCallbacks。同样的在set方法前，我们也添加注解，创建时间属性是在保存时创建的，添加PrePersist注解。更新时间它是在文章创建时和更新时进行赋值的，同样添加PrePersist注解，再额外添加一个PreUpdate注解。

回到浏览器，我们测试一下，打开管理端，打开文章列表。我们新增一篇文章，直接点击添加，现在我们刚刚添加的文章自动的添加了创建时间和更新时间。我们来编辑一下这篇文章，修改一下正文，再点击保存。现在更新时间、创建时间不一致了。说明我们的注解已经起作用了。

回到代码，我们可以为Comment类也添加注解，修改对应的set方法，修改更新时间方法，添加注解。回到浏览器，我们在博客的详情页提交一篇评论 直接点击提交，回到管理端，我们查看一下评论列表。我们刚刚提交的评论现在就有了创建时间，那同样的也会有更新时间。

回到代码，我们可以把创建时间和更新时间属性抽取成一个单独的Trait文件，来增加代码的可复用性。在src目录下，我们新增一个目录，目录名称叫做Utils，在Utils目录中我们新增一个PHP类，名称叫做DateTimeTrait，类的模板我们选择Trait模板，点击ok。

回到Post类，我们复制一下两个属性，粘贴，复制对应的方法，粘贴。我们再把注解引用进来，回到Post类，我们注释掉两个属性，再注释掉对应的get方法和set方法。在类中我们直接使用刚刚创建的Trait，回到Comment类，同样的注释两个属性也注释对应的方法，在类中使用刚刚创建的Trait。

```
#src/Utils/DateTimeTrait.php

trait DateTimeTrait
{
    /**
     * @ORM\Column(type="datetime", nullable=true)
     */
    private $createdAt;

    /**
     * @ORM\Column(type="datetime", nullable=true)
     */
    private $updatedAt;

    public function getCreatedAt(): ?\DateTimeInterface
    {
        return $this->createdAt;
    }

    /**
     * @ORM\PrePersist
     */
```

```
    public function setCreatedAt(): self
    {
        $this->createdAt = new \DateTime();

        return $this;
    }

    public function getUpdatedAt(): ?\DateTimeInterface
    {
        return $this->updatedAt;
    }

    /**
     * @ORM\PrePersist
     * @ORM\PreUpdate
     */
    public function setUpdatedAt(): self
    {
        $this->updatedAt = new \DateTime();

        return $this;
    }

}
```

回到浏览器再次提交一个评论，点击提交。回到管理端刷新评论，我们刚刚提交的评论，它也自动的初始化了创建时间，我们的Trait已经起作用了。如果我们有其他的Entity类需要使用这两个属性，可以直接在类中使用`DateTimeTrait`。

回到管理端的评论列表，现在子评论不会显示文章属性，我们可以在doctrine的生命周期事件中获取子评论的顶级父评论，然后获取顶级父评论的Post属性，再将post属性设置到子评论中，来显示子评论所属于的文章。

在下节课我们将学习doctrine的`postLoad`事件。