继续查看工作流文档，往下看，这里有个BlogPost类。在BlogPost类中有$currentPlace属性，$currentPlace属性就对应着配置文件中的currentPlace属性。在类中有对应的get方法和set方法，和之前的set方法不同的是set方法有第二个参数$context，$context参数对应的是当前的工作流上下文。

我们可以参考示例代码来对我们自己的Post类进行修改。回到项目，之前$status属性只保存一个状态，现在要保存多个状态，我们要修改一下$status属性在数据库中的类型。这里修改为array类型数组类型，删除length，修改对应的get方法和set方法。

getStatus()方法，现在返回值就不是string类型了，是array类型。对应的我们修改setStatus()方法，这里类型我们修改为array，然后添加第二个参数$context，默认值是个空数组。

```
#src/Entity/Post.php

/**
 * @ORM\Entity(repositoryClass=PostRepository::class)
 * @ORM\HasLifecycleCallbacks
 */
class Post
{
    // ...

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

    public function getStatus(): ?array
    {
        return $this->status;
    }

    public function setStatus(?array $status, $context = []): self
    {
        $this->status = $status;

        return $this;
    }
    // ...
}
```

修改了Post类，我们要创建一个数据库的更改文件。查看数据库的更改文件，在migration文件中，status列类型修改为了LONGTEXT类型，执行数据库的更改，yes，关闭控制台。

我们修改了Post类的setStatus()方法和getStatus()方法，我们来搜索一下哪些地方使用了这两个方法。先来搜索setStatus()方法，右键点击Find usages，在Method Call这里有五次使用，先来修改PostFixtures类，setStatus()方法是个数组，这里修改为数组。然后是对PostFactory类进行修改，然后对单元测试的代码进行修改。

随着系统功能的增加，我们可以执行单元测试，然后来修复测试代码中的错误，来对版本中的所有bug进行修复，用这种方法来修复Post类中代码的更改。为了节省时间，我这里就不演示了。

回到Post类，我们来搜索getStatus()方法，在单元测试方法第39行，这里我们需要修改一下，修改为`assertArrayHasKey()`。

现在代码就已经修改完了 (本节课代码的修改仍有问题，后面会继续修改)，在下节课我们来修改管理端，为不同的角色添加不同的操作按钮。