```
#本节课代码请看这个链接
https://github.com/teebbstudios/teebblog/commit/fdd324e9a1136062cd9ba83457e5be4ed3da37d2
```

在上节课我们修改了Post类的$status属性，并且修改了status列在数据库中的类型，我们需要重置一下数据库中的数据。输入symfony console doctrine:fixtures:load，输入yes。我们打开PostController，在博客的首页，我们使用PostRepository对象的getPostPaginator()方法来获取首页的文章列表。

查看这个方法，在方法中对文章的状态进行了过滤，现在status列在数据库中是LONGTEXT类型，这里就不能使用等号了，我们要改用LIKE进行模糊查询。

我们修改$status参数，这里输入百分号点百分号 '%'.$status.'%'。回到浏览器，我们访问博客首页，现在博客的首页和之前显示的是一样的。回到项目，我们来对管理端进行修改。打开PostCrudController类，status属性我们已经修改了它的类型，在这里就不能再使用选择类型了。

注释42行43行代码，我们使用Array类型。ArrayField::new('status')，status属性我们只让它在首页列表显示，onlyOnIndex()。回到浏览器，我们打开博客管理端，登录超级管理员账户。打开文章列表，我们点击新增文章，随便输入一个文章，点击添加。

现在文章的状态是未赋值状态，在工作流中就是我们定义的草稿状态。回到项目，我们打开前面定义的工作流文件，我们希望在文章列表后方，对于不同状态的文章显示不同的状态转换按钮。

回到浏览器，我们希望草稿状态的文章后面可以显示review_request按钮，当点击按钮之后会有对应的controller方法，将对应的文章进行状态的转换。当转换好对应的状态之后，再根据当前的状态显示对应转换按钮。我们需要在PostCrudController类中添加按钮，在configureActions()方法中，首先我们定一个$reviewRequestAction按钮。

使用Action类的new()方法，我们定义一个按钮，按钮的名称我们叫做review_request，对按钮添加一些方法，我们使用linkToCrudAction()方法，指定一个action方法来处理这个按钮的事件，controller方法名称我们叫做reviewRequestAction()。

在PostCrudController类中，我们定义一个新的controller方法，public function 方法的名称叫做reviewRequestAction()。对于CrudController类的controller方法，我们可以传入一个参数$adminContext，我们可以使用$adminContext对象来获取当前按钮所操作的文章，也可以获取当前按钮的名称。

回到项目，我们再次修改configureActions()方法，将我们新增的按钮添加到页面上，我们使用$actions对象的add()方法添加新增的按钮，第一个参数是页面的名称。这里我们使用Crud类的常量PAGE_INDEX，在首页我们添加$reviewRequestAction。回到浏览器刷新，现在文章列表后面显示了这个按钮，我们修改按钮的名称。

回到项目，为Action的new()方法添加第二个参数$label，复制review_request，粘贴，刷新，这样就可以了。回到项目，我们来完成对应的controller方法，首先我们来获取Post对象，使用$adminContext，getEntity()对象，再通过Entity对象getInstance()方法来获取当前的文章。

我们添加一个判断，if，$post对象是Post类的一个实例，那么我们可以进行工作流的状态转换。我们打开控制台，我们输入symfony console debug:container，我们查看所有关于工作流的服务类，我们查看第

11个。当我们在workflow.yaml配置文件中配置了工作流之后，Symfony会自动的在容器中注册一个服务类对象。我们可以直接在普通的controller方法中注入这个对象，然后直接使用工作流对象，对当前的工作流进行状态的转换，但是我们现在使用的是CrudController。

CrudController类中的controller方法只能注入$adminContext这一个服务类，注入其他的服务类会出错。我们只能通过使用`$this->container->get()`方法来获取对应的工作流对象。我们查看第六个工作流对象，第6个工作流对象和第11个工作流对象，都是Symfony根据我们的工作流配置，自动生成的工作流服务类。

我们可以在容器中获取第6个工作流对象，这里服务类的ID我们输入`blog_publishing`，定义一个变量$workflow。使用dd()查看一下$workflow。回到浏览器，我们点击`review_request`按钮，出错了，这个异常我们之前已经见过了。

回到项目，我们需要覆盖另外一个方法，`getSubscribedServices()`，修改方法array_merge()，我们将服务类添加到数组中这里使用`'?'.`工作流的服务类型。我们打开控制台是`WorkflowInterface::class`。

回到浏览器刷新，现在我们就获取到了对应的工作流对象。工作流对象的名称就是`blog_publishing`。回到项目，继续修改代码，我们可以继续添加一个判断。`if ($workflow->can())`，can()方法可以判断当前的对象是否可以执行对应的转换，第一个参数我们传入$post，第二个参数我们传入转换的名称，这里是`review_request`。

如果当前的对象可以执行`review_request`转换，那我们使用`$workflow->apply()`方法应用转换。第一个参数我们传入$post，第二个参数我们传入转换的名称。在应用转换之后，我们查看一下当前的$post对象。回到浏览器，刷新，当前的属性已经修改为了`wait_for_review`，`wait_for_check`。

我们的工作流已经进行了转换。在下一步，我们需要将修改过的$post对象更新到数据库中，我们来获取EntityManager对象，输入$this->container来获取doctrine服务类，再获取对应的EntityManager。使用EntityManager的flush()方法将$post对象更新到数据库中，并不需要再次调用persist()方法保存$post，因为当前的$post对象在EntityManager中是受管理的对象。我们可以直接使用flush()方法保存数据的更改。

在数据插入到数据库之后，我们需要跳转，跳转到文章的列表页。我们可以使用容器来获取另外一个服务类，`AdminUrlGenerator::class`，最后进行返回return，$this->redirect()跳转到文章列表页。使用$adminUrlGenerator对象的setController()方法，设置类名PostCrudController，再设置当前要跳转的Action方法名称，setAction()，这里我们输入`Action::INDEX`。最后使用generateUrl()方法来生成首页的路径。

回到浏览器再次刷新页面，现在文章的状态就进行了修改，并且返回到了文章的列表页。我们查看数据库，刷新，查看最新的文章，我们查看status列。当前文章的状态就进行了修改，修改为`wait_for_review`，`wait_for_check`。在数据库中，它将状态以键的方式进行了存储。

再回到浏览器，我们修改一下状态列在列表中来显示。回到项目，我们打开templates目录，在fields目录中，我们新增一个模板文件，文件名称叫做`post_status.html.twig`。在Twig文件中，我们查看一下当前可用的所有参数，回到PostCrudController，我们来修改一下字段的模板。在status列这里我们使用另外一个方法，setTemplatePath()来指定模板文件，这里使用`fields/post_status.html.twig`。

回到浏览器刷新，我们可以使用field参数，然后获取field参数的value属性，通过对value属性进行便利来获取文章当前的状态。回到项目，我们添加一个for 循环，`for status in field.value`，`endfor`。`field.value`属性中保存的是数组，我们关心的是数组的键。我们可以为for循环添加两个变量，第一个是status，对应的是数组的键，第二个是value对应的数组的value。我们只关心数组的键值，这里输入status，然后添加一个逗号进行分割。

回到浏览器刷新，现在就显示了当前文章的状态，我们看到所有的文章后面都添加了review_request按钮。我们需要进行调整，只有可以进行对应转换的文章才显示对应的状态转换按钮。回到项目，打开PostCrudController，修改configureActions()方法。为$reviewRequestAction添加条件判断，displayIf()，fn()这里输入$entity，使用箭头方法，我们使用$workflow对象，对$entity对象添加一个判断，只有当前的文章对象可以进行review_request转换时，才会显示review_request按钮。

我们来获取一下$workflow对象，`$workflow = $this->container->get('workflow.blog_publishing');`，回到浏览器刷新，出错了。我们查看一下数据库。刷新文章列表，我们查看新增的文章。文章的状态是用键值对的方式在数据库中进行保存和显示，但是我们通过Fixtures生成的文章，文章的状态并不是以键值对的方式进行保存的。

我们需要修改Fixtures代码，回到项目，修改PostFixtures。在这里我们修改为published等于1，我们再来修改PostFactory类，这里$status属性我们要设置为指向1。重置数据库，输入yes。

回到浏览器刷新，登录超级管理员，我们看到现在已发布状态的文章就不会显示review_request按钮。草稿状态的文章就会显示review_request按钮。我们点击review_request，我们看到第二篇文章状态就进行了修改，修改为wait_for_review，wait_for_check。

回到项目，我们可以通过复制的方式添加另外几个按钮，并且添加对应的controller方法，为了节省时间我这里就不再添加了。

在下节课，我们将学习如何在工作流中对权限进行验证。