双击shift键，我们搜索AbstractController。如果没有找到这个类的话，我们勾选一下这里。打开AbstractController类，我们查看一下类中的方法，找到isGranted()方法。我们通过这个方法来检查用户的权限，在这个方法通过容器来获取一个服务类，通过服务类调用isGranted()方法，我们查看一下isGranted()方法。

在isGranted()方法中，最终会调用AccessDecisionManager对象的decide()方法，我们查看一下decide()方法。decide()方法是AccessDecisionManagerInterface定义的一个方法。我们查看接口所在文件夹，在文件夹中有AccessDecisionManager类。

我们查看一下这个类，类中有个decide()方法。在decide()方法中，最终它会根据一个策略来调用不同的方法，我们往下随便看一个方法，在这个方法中遍历所有的Voter对象，使用Voter对象的vote()方法来返回一个结果，最终根据结果来判断当前用户是否有对应的权限。

我们可以通过自定义Voter来实现一个自定义的权限。回到浏览器，我们搜索symfony voter。查看Symfony文档，我们查看自定义Voter那段文档，自定义的Voter类需要继承于Voter类。在Voter类中定义了两个常量，分别对应了两个权限。

然后是supports()方法，如果supports()方法返回为真，就开始对$attribute属性进行投票验证。如果vote()方法投票结果为真，那么当前的用户就有这个权限。

回到项目，我们打开控制台。输入symfony console list make，make:voter命令行可以帮助我们创建一个Voter类。复制命令行，输入symfony console make:voter，Voter名称我们叫做PostVoter。我们查看一下命令行生成的类，Voter类中有两个方法。分别是supports()方法和voteOnAttribute()方法。

在supports()方法中，我们判断当前验证的属性是不是POST_EDIT属性或者POST_VIEW属性。我们来修改一下定义两个常量。const POST_OWNER_EDIT，const POST_OWNER_DELETE。

修改supports()方法，如果当前验证的属性是这两个属性的话。并且当前的$subject对象是文章对象的一个实例，那我们就开始执行投票vote()。这里修改为self::POST_OWNER_DELETE，self::POST_OWNER_EDITOR。

我们继续修改voteOnAttribute()方法，我们修改switch()方法的case条件。这里修改为self::POST_OWNER_EDIT，self::POST_OWNER_DELETE。注释35行的break，我们让这两个条件执行同一个操作。我们来添加一个条件判断，if $subject，当前的$subject对象是个文章对象，我们获取作者getAuthor()。如果当前文章的作者等于当前的$user，那么就返回true，否则的话就返回false。

```
#src/Security/Voter/PostVoter.php

class PostVoter extends Voter
{
    const POST_OWNER_EDIT = 'post_owner_edit';
    const POST_OWNER_DELETE = 'post_owner_delete';

    protected function supports(string $attribute, $subject): bool
    {
        // replace with your own logic
        // https://symfony.com/doc/current/security/voters.html
        return in_array($attribute, [self::POST_OWNER_DELETE,
self::POST_OWNER_EDIT])
```

```
            && $subject instanceof \App\Entity\Post;
    }

    protected function voteOnAttribute(string $attribute, $subject,
TokenInterface $token): bool
    {
        $user = $token->getUser();
        // if the user is anonymous, do not grant access
        if (!$user instanceof UserInterface) {
            return false;
        }

        // ... (check conditions and return true to grant permission) ...
        switch ($attribute) {
            case self::POST_OWNER_EDIT:
                // logic to determine if the user can EDIT
                // return true or false
//                break;
            case self::POST_OWNER_DELETE:
                // logic to determine if the user can VIEW
                // return true or false
                if ($subject->getAuthor() == $user){
                    return true;
                }
                break;
        }

        return false;
    }
}
```

回到浏览器，我们需要为文章操作的controller方法添加权限验证。回到项目，我们打开PostCrudController类查看父类。查看父类中的方法，在父类中通过edit()方法和delete()方法来编辑或者删除文章，查看一下edit()方法。在edit()方法中，首先发送了事件。在delete()方法中也同样发送了事件。我们可以使用监听器或者订阅器来处理这个事件，在事件处理中来对文章对象进行权限的检查。

打开控制台，输入`symfony console make:subscriber`，订阅器的名称我们叫做`PostPermissionSubscriber`，订阅`BeforeCrudActionEvent`事件，拷贝类名，粘贴。

打开订阅器源码，我们查看一下事件类的源码，在事件类中有个$adminContext对象。我们可以使用$adminContext对象来获取当前的文章对象，并且获取当前的操作名称。

回到订阅器类，添加一个变量`$adminContext=$event->getAdminContext();`。首先我们来获取当前的文章对象，`$instance = $adminContext->getEntity()->getInstance();`。然后我们来获取Crud对象，等于`$adminContext->getCrud()`。

我们添加一个条件判断，如果$instance变量是Post类的一个实例。再次增加一个判断，如果当前的Crud操作getCurrentAction()等于edit，那么我们就添加一个条件判断。elseif，如果当前的Crud操作名称等于delete，我们添加另外一个判断。

在两个判断中，我们需要验证权限。验证权限我们需要使用到Security对象。在构造方法中，我们通过依赖注入的方式注入Security对象。按着alt键加回车我们初始化属性，在条件判断中我们来进行权限的认证。$this->security->isGranted();，第一个参数是属性的名称，这里使用PostVoter类的POST_OWNER_EDIT常量。第二个参数是$subject，这里我们输入$instance，我们添加一个结果$result，如果$result的值为false。那么我们抛出一个异常throw new AccessDeniedException();。

只有文章作者才可以编辑当前文章，同样的我们复制代码，粘贴。这里修改为POST_OWNER_DELETE，消息这里修改为可以删除当前文章。现在我们的订阅器代码就已经编写好了。

```
#src/EventSubscriber/PostPermissionSubscriber.php

class PostPermissionSubscriber implements EventSubscriberInterface
{
    /**
     * @var Security
     */
    private Security $security;

    public function __construct(Security $security)
    {
        $this->security = $security;
    }

    public function onBeforeCrudActionEvent(BeforeCrudActionEvent $event)
    {
        $adminContext = $event->getAdminContext();
        $instance = $adminContext->getEntity()->getInstance();
        $crud = $adminContext->getCrud();
        if ($instance instanceof Post) {
            if ($crud->getCurrentAction() == 'edit') {
                $result = $this->security-
>isGranted(PostVoter::POST_OWNER_EDIT, $instance);
                if (!$result){
                    throw new AccessDeniedException('只有文章作者才可以编辑当前
文章');
                }
            } elseif ($crud->getCurrentAction() == 'delete') {
                $result = $this->security-
>isGranted(PostVoter::POST_OWNER_DELETE, $instance);
                if (!$result){
                    throw new AccessDeniedException('只有文章作者才可以删除当前
文章');
                }
            }
        }
    }

    public static function getSubscribedEvents()
    {
        return [
            BeforeCrudActionEvent::class => 'onBeforeCrudActionEvent',
        ];
```

```
        }
    }
```

回到浏览器，我们当前是editor用户，我们来编辑一下自己的文章。没有问题。我们再来编辑下其他用户的文章，它提示我们只有文章作者才可以编辑当前文章。回退，我们点击删除，也只有文章的作者才可以删除文章，我们的Voter起作用了。

我们可以隐藏其他作者文章的编辑和删除按钮来提升用户的体验，在下节课我们继续修改代码。