我们来搜索symfony userchecker，打开Symfony文档。这里有一段关于UserChecker的一些描述。我们来翻译一下，在用户身份验证期间，可能需要额外的检查，来验证是否允许识别的用户登录。可以通过自定义UserChecker，然后为每个防火墙定义使用哪个UserChecker。

我们继续往下看，下面有段UserChecker的示例代码。UserChecker需要实现UserCheckerInterface接口，然后实现两个方法，分别是checkPreAuth()和checkPostAuth()。见名知意，checkPreAuth()发生在用户认证之前，checkPostAuth()发生在用户认证之后。

在checkPreAuth()方法中，如果用户已经被删除了，那么就抛出一个异常，您的账号已经不存在了。在checkPostAuth()中，如果用户已经过期了，我们就可以抛出账号过期异常。

回到项目打开src目录，我们在Security目录中新增一个php类，类名叫做UserChecker。UserChecker需要实现UserCheckerInterface，鼠标移动到红线上。我们点击添加方法，点击OK。

回到浏览器，我们来复制代码，粘贴。这里修改一下，我们有自己的User类，引入一下异常的类型。这里消息我们修改一下，Your account is Expired。

```
#src/Security/UserChecker.php

class UserChecker implements UserCheckerInterface
{
    public function checkPreAuth(UserInterface $user): void
    {
        if (!$user instanceof User) {
            return;
        }

        if ($user->isDeleted()) {
            // the message passed to this exception is meant to be
displayed to the user
            throw new CustomUserMessageAccountStatusException('Your user
account no longer exists.');
        }
    }

    public function checkPostAuth(UserInterface $user): void
    {
        if (!$user instanceof User) {
            return;
        }

        // user account is expired, the user may be notified
        if ($user->isExpired()) {
            throw new CustomUserMessageAccountStatusException('Your
account is Expired.');
        }
    }
}
```

现在我们需要修改用户类，为用户类添加字段。打开控制台，输入`symfony console make:entity`，类名输入User，新的属性名称我们使用日期格式`deletedAt`。如果删除日期属性不为空，那么当前的用户就是已删除的状态，类型我们选择datetime。当前字段在数据库中可以为空吗？可以为空。添加另外一个属性，`expiredAt`，也是日期类型，可以为空。

回车，我们来查看一下User类，在User类中添加了两个属性，我们来添加两个方法。`public function isDeleted()`，它的返回值是个布尔值。我们根据删除日期来判断当前用户是否已被删除。`return $this->deletedAt不为空`，那么当前用户就被删除了。

添加另外一个方法，`public function isExpired()`，返回值是bool类型。如果$this->expiredAt不为空，并且过期时间$this->expiredAt小于当前时间new \DateTime()，那么当前的用户就已经过期了。

我们打开控制台创建一下数据库的更改文件，我们查看一下数据库的更改文件。在user表中添加了两列，我们来执行数据库的更改，输入yes。现在数据库的更改就可以了，最后我们需要将`UserChecker`配置到防火墙中。

打开security.yaml配置文件，在main防火墙下我们添加配置`user_checker`，复制全类名，粘贴。我们来修改一下Fixtures文件，为系统添加两个用户。打开DataFixtures目录，打开AppFixtures，为了节省时间，这里我就粘贴代码了。

```
#src/DataFixtures/AppFixtures.php

class AppFixtures extends Fixture
{
    // ...

    public function load(ObjectManager $manager)
    {
        $admin = new User();
        $admin->setUsername('admin');
        $admin->setRoles(['ROLE_SUPER_ADMIN']);
        $admin->setPassword($this->userPasswordHasher-
>hashPassword($admin, 'admin'));

        $deletedUser = new User();
        $deletedUser->setUsername('deleted');
        $deletedUser->setRoles(['ROLE_SUPER_ADMIN']);
        $deletedUser->setDeletedAt(new \DateTime('-1 day'));
        $deletedUser->setPassword($this->userPasswordHasher-
>hashPassword($deletedUser, '123'));

        $expiredUser = new User();
        $expiredUser->setUsername('expired');
        $expiredUser->setRoles(['ROLE_SUPER_ADMIN']);
        $expiredUser->setExpiredAt(new \DateTime('-1 day'));
        $expiredUser->setPassword($this->userPasswordHasher-
>hashPassword($expiredUser, '123'));

        $manager->persist($admin);
        $manager->persist($deletedUser);
        $manager->persist($expiredUser);
```

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

我们分别使用new关键字创建了两个用户，然后我们设置第一个用户的删除时间。`$deletedUser->setDeletedAt(new \DateTime());`删除时间-1 day，减一天，密码是123。然后我们为第二个用户设置过期时间，`setExpiredAt(new \DateTime())`，也是减一天，密码我也修改为123。最后使用$manager保存一下这两个用户。`persist($deletedUser)`，`$manager->persist($expiredUser);`。

打开控制台，重置一下数据库。`symfony console doctrine:fixtures:load`，yes。我们查看一下数据库中的数据，现在用户表中就新增了两个用户。第一个用户他的删除时间不为空，第二个用户他的过期时间不为空。

我们打开浏览器尝试登录这两个用户，第一个用户名deleted，密码123点击登录，现在就显示了一个错误信息，`您的账号已经不存在了`。我们来登录另外一个账号密码123，点击登录，这里的错误信息显示的是无效的凭证。

回到项目，我们在checkPostAuth()方法中下一个断点。开启xdebug监听，再次登录账号，断点断在了`checkPostAuth()`方法中。我们向下执行，当用户账号已过期的时候，会抛出`AccountExpiredException`这个异常。

我们继续往下执行，然后`AuthenticatorManager`对象来处理这个异常，最后返回一个响应对象。继续往下，我们查看一下$response对象，$response对象当前进行了一个跳转，又跳转到了登录页面。

我们来修改一下UserChecker类，我们将异常类的名称进行修改。修改为上面的异常。回到浏览器再次进行登录，现在就提示了`您的账户已过期`。我们已经完成了用户登录的功能，所有用户登录的功能我们可以叫做用户的认证，只有通过认证的用户才可以登录系统。

下面我们需要为已登录的用户添加授权，限制某些操作的权限。在下节课我们开始学习用户的授权。