我们已经有管理端的页面了，但是管理端的页面并没有被保护起来，随时都可以访问。我们只允许登录过的管理员用户才可以访问管理端，我们就需要有个用户系统了。首先我们要创建用户类，回到项目，我们查看一下所有的make命令行。输入`symfony console list make`，这里有个`make:user`命令行，可以创建一个用于安全系统的用户类。

我们复制命令行，输入`symfony console`，粘贴命令行，命令行会帮助我们创建一个用户类。用户类的类名，我们叫做`User`类。现在提示我们需要把用户的数据保存到数据库中吗？yes，我们直接回车，现在让我们输入一个属性名称，它必须是唯一的用于用户的登录。我们这里选择`username`，这一步提示我们需要验证用户的密码吗？我们需要验证。命令行帮助我们创建了一个`User`类和`UserRepository`类。然后更新了`security.yaml`配置文件。

我们先来看一下User类，User类实现了两个接口，User中有个$username属性，角色属性和$password属性，分别是用户名，用户的角色和密码。我们再来查看一下配置文件，找到`security.yaml`文件。

```
#config/packages/security.yaml

security:
    encoders:
        App\Entity\User:
            algorithm: auto

    #
https://symfony.com/doc/current/security/experimental_authenticators.html
    enable_authenticator_manager: true
    # https://symfony.com/doc/current/security.html#where-do-users-come-
from-user-providers
    providers:
        users_in_memory: { memory: null }
        # used to reload user from session & other features (e.g.
switch_user)
        app_user_provider:
            entity:
                class: App\Entity\User
                property: username
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            provider: app_user_provider

            # activate different ways to authenticate
            # https://symfony.com/doc/current/security.html#firewalls-
authentication
            #
https://symfony.com/doc/current/security/impersonating_user.html
            # switch_user: true
    # Easy way to control access for large sections of your site
    # Note: Only the *first* access control that matches will be used
    access_control:
```

```
        # - { path: ^/admin, roles: ROLE_ADMIN }
        # - { path: ^/profile, roles: ROLE_USER }
```

配置文件中增加了一些配置，首先是加密器配置，这里增加了User类的加密配置，它会对$password属性进行加密。这里算法设置设置为auto。Symfony会选择当前环境最合适的算法。我们看第7行，Symfony从5.1版本开始提供了一个新的基于认证器的认证系统，在本套课程我们将来学习这套新的认证系统。

然后是providers配置，providers配置用来提供用户类的数据，这里是provider的配置名称。这个名称是可以自由定义的，定义好这个provider名称之后，Symfony会自动的在容器中注册一个对应的服务类。

我们打开控制台，输入`symfony console debug:container provider`，这里就显示了所有跟`provider`关键字有关的服务类，我们看第15条服务类。Symfony会根据provider配置自动的在容器中注册一个服务类，这个服务类可以通过property属性来获取对应的User对象。

我们继续往下看，下面是`firewalls`防火墙配置，dev和main分别是防火墙的标识在dev配置下有个正则表达式。当我们在开发环境时，我们页面的底部有个工具栏，它会发送Ajax请求。Ajax请求中包含这些路径。当访问这些路径时，就不会经过安全系统的验证，另外所有的资源文件也都不会经过安全系统的验证。在main配置中没有设置正则表达式，那么所有路径的访问都会经过当前的防火墙，所以防火墙的配置是有先后的，先配置dev后配置main。这样在访问一些路径时，就可以不经过安全系统。

`lazy`这里设置为true，这里就按照默认的设置就可以了。我们再看provider配置，provider设置它指向`app_user_provider`，这样在main防火墙配置中，会通过`app_user_provider`服务类对username属性来进行查询，来获取User对象。

最后是`access_control`配置，还可以在某些页面路径下设置一些访问的权限。比如说我们想把管理端保护起来，就可以设置一个用户的角色，只有拥有这个角色的用户才可以进行访问。

打开控制台，我们使用命令行创建了用户类，我们来创建一个migration文件，查看一下migration文件。migration文件创建了一个用户表，里面包含用户名，角色和密码，我们来执行这个数据库的更改，现在数据库就更改好了。

在下一节，我们使用Fixtures来增加一些用户数据。