打开src目录下DataFIxtures目录，我们打开AppFixtures类，这个类是我们在安装fixtures-bundle时自动添加的，我们在这个类中来添加一些用户的数据。

我们来创建一个超级管理员，设置管理员的用户名，这里我们设置为admin，设置管理员的角色，角色名称是可以自由定义的，但是我们一般都是用大写的ROLE_加下划线来开头，然后后面加上对应的单词都使用大写。然后我们设置超级管理员的密码，密码是个字符串，这里我们不能设置为明文，我们需要对密码进行加密。

我们打开上节课的配置文件，在security.yaml配置文件中，为User类添加了一个加密器的配置。我们打开控制台，输入`symfony console debug:container`，参数我们输入encoder，看第15个服务类，当前的服务类是用户密码的加密器类。我们可以通过依赖注入的方式，将`UserPasswordEncoder`对象注入到AppFixtures类中。

我们先来查看一下这个类的代码，复制类名，双击shift键粘贴。如果你没有找到这个类的话，这里需要勾选一下，我们打开这个类，`UserPasswordEncoder`类提供了`encodePassport()`方法。它需要传递一个User对象，然后传递密码的明文，在第42行`encoderFactory`会根据用户对象来获取一个$encoder对象，$encoder对象的算法就是我们在配置文件中配置的算法。

回到AppFixtures类添加构造方法，我们注入`UserPasswordEncoderInterface`，这里我们看到它已经被弃用了。我们查看一下源码，在第17行当前接口被另外一个接口替代了。我们使用这个新的接口，回到AppFixtures，输入`UserPasswordHasherInterface`。按着alt键点击回车来初始化这个属性，我们再次打开控制台来查看一下这个接口。

这次我们输入新接口的名称，新的接口它会使用`UserPasswordHasher`类来对用户密码进行加密，我们查看一下这个类，选择第二个，在这个类中使用`hashPassword()`方法来对用户的明文密码进行加密。

回到AppFixtures，在setPassword()方法中，我们输入`$this->usserPasswordHasher`使用`hashPassword()`。第一个参数我们传入admin，第二个参数我们也传入admin。最后使用`$manager->persist()`方法来保存一下当前的管理员数据。

```
#src/DataFixtures/AppFixtures.php

class AppFixtures extends Fixture
{
    /**
     * @var UserPasswordHasherInterface
     */
    private UserPasswordHasherInterface $userPasswordHasher;

    public function __construct(UserPasswordHasherInterface
$userPasswordHasher)
    {
        $this->userPasswordHasher = $userPasswordHasher;
    }

    public function load(ObjectManager $manager)
    {
        // $product = new Product();
        // $manager->persist($product);
        $admin = new User();
```

```
        $admin->setUsername('admin');
        $admin->setRoles(['ROLE_SUPER_ADMIN']);
        $admin->setPassword($this->userPasswordHasher-
>hashPassword($admin, 'admin'));

        $manager->persist($admin);

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

打开控制台，我们来重置一下数据库中的数据`doctrine:fixtures:load`，yes，我们查看数据库。刷新，这里就有了个用户表，查询。我们就创建了一个超级管理员用户，并且它的密码已经被加密了。

在下节课，我们来创建用户登录的界面。