我们已经在security.yaml配置文件中配置了我们自己编写的认证器类，我们查看一下认证器的源码。我们自己的认证器类，它有一个父类，我们查看一下父类的代码。父类是个抽象类，它定义了getLoginUrl()抽象方法，子类必须实现这个方法。

我们再往下看，还有个supports()方法，当浏览器访问一个请求时，通过supports()方法来判断当前的认证器类是否要继续执行认证。如果返回为真 就继续往下执行认证，否则的话就中断认证。

继续往下看，还有`onAuthenticationFailure()`方法，这个方法见名知意，当认证失败时会执行的方法。我们再往下看，有个start()方法，我们看注释，当我们访问一个被保护的页面时。如果我们还没有登录，就会调用这个方法，在这个方法中他会进行个跳转，跳转到我们的登录页面。

回到我们自己的认证器类，我们在`authenticate()`方法中，添加一个断点，监听xdebug请求。回到浏览器，我们来访问一下管理端页面，我们当前没有登录，会自动跳转到登录界面。

我们输入管理员账号和密码，点击登录按钮，这时断点就断在`authenticate()`方法中。我们来查看一下方法调用的堆栈，首先它会调用index.php文件，我们继续往上看，在HttpKernel.php文件中，首先它会执行handle()方法，在handle()方法中，执行handleRaw()方法。handleRaw()方法我们已经很熟悉了。

在handleRaw()方法中，一开始就监听了`RequestEvent`事件，事件的名称是`kernel.request`事件，我们继续往下看，会调用一个监听器。在这里，Firewall.php文件中会调用`onKernelRequest()`方法，我们来查看这个方法，`onKernelRequest()`方法处理`kernel.request`事件。

我们看一下代码，在方法中会获取所有的关于认证的监听器，最后使用`callListeners()`方法来执行所有监听器。我们继续往下看，最后它会找到`AuthenticatorManagerListening`类执行`authenticate()`方法。我们查看一下这个方法，在方法中，它会获取`AuthenticateManager`对象，调用`authenticateRequest()`方法。

我们查看一下`AuthenticatorManager`类，在`authenticateRequest()`方法中调用`executeAuthenticators()`方法，我们继续往下看，在`executeAuthenticators()`方法中，它会调用`executeAuthenticator()`方法。

查看一下`executeAuthenticator()`方法，在`executeAuthenticator()`方法中，会获取我们自己编写的$authenticator对象，然后执行对象的`authenticate()`方法，返回一个$passport对象。然后发送一个事件，我们继续往下看，最后会根据$passport对象和防火墙的名称生成一个$token。

我们继续往下看，如果出现异常的话，就会执行认证失败的方法。我们查看一下认证失败的方法，在认证失败的方法中，最后它会调用我们自定义的$authenticator对象的`onAuthenticationFailure()`方法返回一个响应，最后将响应发送浏览器。

回到刚刚的代码，如果没有异常的话，执行success方法。我们查看认证成功的方法，在认证成功的方法中，将创建的用户令牌设置到storage中，最后返回一个Response对象。这就是用户登录认证的所有流程。

回到我们的Authecticator类，我们可以在登录成功的方法中获取登录的用户对象，根据用户的角色跳转到不同的页面。我们可以使用$token对象的getUser()方法来获取用户对象，然后再使用$user对象的getRoles()方法获取用户的角色。然后进行条件判断，最后根据不同的角色跳转到不同的页面。

我们打开用户登录的模板文件，在模板文件中有一段注释的代码，通常我们在用户登录时会有一个复选框提示我们要不要保存登录状态，我们取消代码的注释。回到浏览器刷新，现在就有一个Remember me复选框。

我们希望在勾选复选框之后，浏览器可以记录我们的登录状态，在下次进行访问时会自动的登录我们的账号。下节课，我们将讲解Remember me配置。