Symfony2框架实战教程——第四天：用HWIOAuthBundle实现第三方登录

发表回复

昨天我们添加了新闻相关功能，今天我们来实现用户功能，毕竟我们不可以随便让用户来我们这里发布新闻。

为了方便用户能快速发布内容，我们只用要求QQ的第三方登录就行了。

如果你打算跟着我一起完成这个项目，可能会因为QQ需要验证你是否有个人域名而卡在这一章，我写了一篇不用OAuth的。但本篇文章也最好看看，大部分知识点是完全一样的。

使用**HWIOAuthBundle**实现第三方登录

可以想象得到，第三方登录这种常见需求，相关的库也是有的。这里我再推荐一个：HWIOAuthBundle

如同昨日，我们先将它通过Composer引入到我们项目当中：

```
$ composer require "hwi/oauth-bundle:0.4.*@dev"

```

并将其注册到AppKernel里：

```
// app/AppKernel.php

public function registerBundles()
{
    $bundles = array(
        // ...
        new HWI\Bundle\OAuthBundle\HWIOAuthBundle(),
    );
}

```

用过OAuth的同学应该知道，一个完整的OAuth客户端程序会包含许多链接。HWIOAuthBundle已经有一部分写好的路由，我们将它们引入到项目当中：

```
# app/config/routing.yml
hwi_oauth_redirect:
    resource: "@HWIOAuthBundle/Resources/config/routing/redirect.xm
    prefix:   /connect

hwi_oauth_login:
    resource: "@HWIOAuthBundle/Resources/config/routing/login.xml"
    prefix:   /login

```

Bundle里可以自带路由规则以及对应的控制器，这也是Bundle的特点之一。

然后依照HWIOAuthBundle的文档，我们需要做一些配置：

```
# app/config/config.yml

hwi_oauth:
    firewall_name: secured_area

```

此配置定义了需要使用HWIOAuthBundle登录后才能访问的地址。但是我们可以看到配置里并没有地址，只有一个防火墙名称，这是怎么一回事？先不用管我们接着添加配置：

```
# app/config/security.yml
security:
    firewalls:
        # ...
        secured_area:
            pattern: ^/
            oauth:
                login_path: hwi_oauth_connect
                failure_path: hwi_oauth_connect

```

`app/config/security.yml`是专门用来配置用户访问权限的文件。其中`firewalls`里定义了若干防火墙。可以看到，第一个防火墙就是刚才配置的防火墙名称。

在`secured_area`防火墙里，可以定义此防火墙作用的路径。由于开启了`HWIOAuthBundle`的缘故，防火墙可以

支持oauth配置，里面定义了登录路径login_path，登录失败去的路径failure_path，他们的值都是HWIOAuthBundle用来登录的路由名。这里教大家一个实用的命令，可以查看当前项目里都已经定义了哪些路由：

```
1 $ php app/console debug:router
2
```

最后还有一个oauth_user_provider选项，目前我们暂时不用管它。

如果您是按步骤看到这里来的，你会发现在firewalls里还定义了其他的防火墙dev、demo_login、demo_secure_area。因为之前被删掉的AcmeDemoBundle自带了权限控制和登录表单的演示，所以定义了两个demo_***的防火墙，目前已经可以放心删除。最后还剩一个dev防火墙，可以看到它定义了静态文件目录css|images|js可以不需要通过防火墙就能访问，另外_(profiler|wdt)这两个目录包含了开放工具的访问路径，也没有必要受防火墙的保护。

这里还需要注意防火墙定义的顺序和有限级：配置位置越靠前，优先级越高，比如说当用户访问一个路径/css/main.css的时候，第一个防火墙配置（虽然此配置其实是用来关闭防火墙的……）已经生效，就不会再往下匹配了。

HWIOAuthBundle已经帮我们做了很多代码上的工作，比如QQ的OAuth登录地址是什么，参数名是什么……这些HWIOAuthBundle都已经帮我们设置好了，我们需要做的仅仅是定义一个OAuth服务端，告诉HWIOAuthBundle帐号是谁提供的（这里有一个HWIOAuthBundle目前支持的OAuth服务端列表），以及设置api_key和api_secret：

```
1 # app/config/config.yml
2 hwi_oauth:
3     resource_owners:
4         # owner的名字可以随便取，只要是唯一的名字就行
5         qq:
6             type:                qq
7             client_id:           <client_id>
8             client_secret:       <client_secret>
9
```

定义好OAuth服务端后，我们再在secured_area防火墙里将OAuth服务端的回调地址设置上：

```
# app/config/security.yml
security:
    firewalls:
        # ...
        secured_area:
            oauth:
                resource_owners:
                    qq: check_qq

```

注意这里的resource_owners其实指的是resource_owner callback path，作者完全可以取一个更准确的名字。

目前我们还没有定义名为`check_qq`的路由，赶紧加上：

```
# app/config/routing.yml
check_qq:
    pattern: /login/check-qq

```

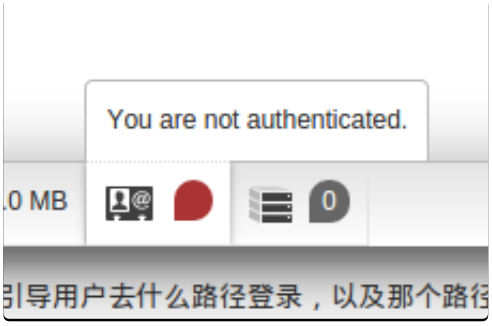
我们可以发现，这里并没有为某个路径对应一个控制器，这又是怎么一回事？我只先透露通过事件监听器是可以做到的，目前就不多说了。

让我们再回到`security.yml`文件。这里再简单说说firewall的设计思想。每一个firewall都会定义了以下一些事情：

1. 是开启还是关闭firewall，如果`security: false`，就是指此firewall是被关闭的，比如dev防火墙
2. 哪些路径适用于此firewall规则，这通过`pattern`来指定，可以参考dev防火墙的配置
3. 如果用户没有登录，将以什么方式登录，比如常见的使用登录表单的方式，如果使用此种方式，还要定义将引导用户去什么路径登录，以及那个路径是用来检查用户登录信息输入的。这里需要注意的是，登录页面的路径是不可以被防火墙保护起来的（不过有一种情况除外，后面再说），原因我就不说了吧，还是很容易想到的。

其他还有很多控制行为细节和技术细节的东西，这里就不多说了，遇到的时候自然会说的。

在用户没有登录的时候，开发工具栏也会有显示，方便大家了解当前的登录状态，未登录的时候将会如下显示



在使用防火墙的路径访问时，Symfony检查访问者一个叫做Auth Token的东西，就好象武侠片里的太监进皇宫，手里得有令牌证明自己的身份一样。要想取得令牌，就得去登录页面登录获取。防火墙的工作也及其简单：带着令牌的，欢迎！没有令牌？出去！只不过，有一种情况除外，那就是防火墙可以设置未登录的访问者，各个都自动颁发一个叫anonymous auth token的东西，匿名访问。此时虽然用户并没有真正登录，但是却跟登录的状态一样。如果是这样一种配置，等于防火墙是都放行的，所以还需要一个叫access_control的东西，来检查每一个用户拿的都是什么样的令牌，可以做什么样的事情。

知道了以上概念，可以说，做身份检查其实可以有两种风格，第一种风格，不登录没有令牌，配置好防火墙的访问路径，将不能公开访问的”皇宫“用墙直接隔开起来；第二种风格，大家都可以进，通过权限控制的方式来限制”某些身份不明不白的人“的行为。

我个人是比较喜欢第一种方式。我喜欢让防火墙检查用户”你是谁“，而让访问控制器检查”你能做什么“这种明确的分工方式，而且，本来就是公开的区域还需要给用户创建”令牌“对象我觉得是一个比较浪费的做法。不过，因为HWIOAuthBundle在登录界面的时候，必须要检查Auth Token（按理来说应该先检查有没有Auth Token，再检查Auth Token是什么身份），这里我们只能采用第二种方式。

我们将让secured_area可以匿名访问（还叫secured这样好吗……）

```
1 # app/config/security.yml
2 security:
3     firewalls:
4         secured_area:
5             anonymous: ~
6
```

然后在access_control里设置，在创建新闻以及修改新闻的时候，要求必须登录：

```
1 # app/config/security.yml
2 security:
3     access_control:
```

```
4         - { path: ^/news/(new|(\d+\/edit)), roles: IS_AUTHENTICATED
5
```

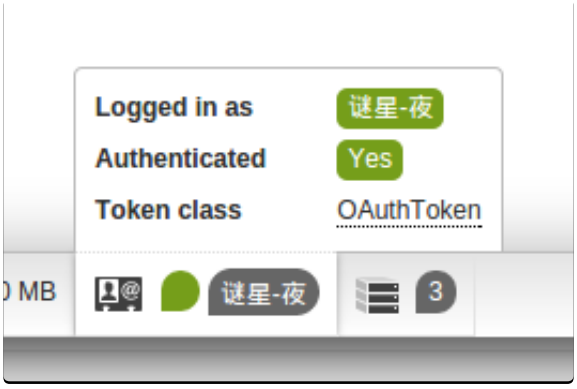
到此，我们权限相关的配置已经完成了。

不过，事情还是没结束……在Symfony2的世界里，每一个在网站访问的登录用户，都需要一个叫做User Provider的东西来生成，在Symfony2里UserProvider都需要实现UserProviderInterface，我们只需要实现此接口，返回一个Symfony2的用户对象就行。不过还好HWIOAuthBundle已经帮我们实现了这个接口，并且已经为其定义好了服务，我们只用使用它就可以了：

```
1  # app/config/security.yml
2  security:
3      providers:
4          oauth:
5              id: hwi_oauth.user.provider
6
7      firewalls:
8          secured_area:
9              oauth:
10                 oauth_user_provider:
11                     service: hwi_oauth.user.provider
12
```

好吧，其实我也不知道为什么要配置两次……

事情到这个地步，还没有结束……好吧，其实已经结束了。现在我们可以访问一下/news路径，看看是不是会自动跳转到/login页面，是否会从QQ网站登录并成功返回登录的用户名：



可以看到，虽然HWIOAuthBundle有这样那样的问题，但是不得不承认，它还是极大的提高了开发的效率。

最后记得将测试用的`in_memory`类型的user provider删掉。