Symfony2框架实战教程——第四天#Alt：用FOSUserBundle实现用户注册和登录

15条回复

欢迎来到第四天教程的里世界……

因为第四天的教程可能会导致一些同学无法继续跟着完成教程里的项目，所以在里世界里我将再写一篇教程。

使用**FOSUserBundle**实现用户注册功能

好吧，像用户管理这种常见的功能，我当然也有bundle推荐：FOSUserBundle

如同以往，安装Bundle：

```
1 $ composer require friendsofsymfony/user-bundle "~2.0@dev"
2
```

注册Bundle：

```
1  // app/AppKernel.php
2
3  public function registerBundles()
4  {
5      $bundles = array(
6          // ...
7          new FOS\UserBundle\FOSUserBundle(),
8      );
9  }
10
```

FOSUserBundle要求我们创建一个继承自`FOS\UserBundle\Model\User`的用户类：

```
1  <?php
2  // src/AppBundle/Entity/User.php
3
```

```
4  namespace AppBundle\Entity;
5
6  use FOS\UserBundle\Model\User as BaseUser;
7  use Doctrine\ORM\Mapping as ORM;
8
9  /**
10  * User
11  *
12  * @ORM\Entity
13  * @ORM\Table()
14  */
15 class User extends BaseUser
16 {
17     /**
18      * @ORM\Id
19      * @ORM\Column(type="integer")
20      * @ORM\GeneratedValue(strategy="AUTO")
21      */
22     protected $id;
23 }
24
```

为了使用FOSUserBundle来做登录，我们还得写一些配置：

```
1  # app/config/security.yml
2  security:
3      encoders:
4          FOS\UserBundle\Model\UserInterface:
5              algorithm: bcrypt
6              cost: 10
7
8      providers:
9          fos_user:
10             id: fos_user.user_provider.username
11
12     firewalls:
13         secured_area:
14             pattern: ^/
15             anonymous: ~
16             form_login:
17                 provider: fos_user
```

```
18                 csrf_provider: form.csrf_provider
19             logout:       true
20
21     access_control:
22         - { path: ^/news/(new|(\d+\/edit)), roles: IS_AUTHENTICATE
23
```

如果你也看了第四天的教程，你会发现一些类似的东西：定义了User Provider，指定了防火墙，防火墙可以匿名访问，但是新闻提交和编辑页需要登录权限……

不一样的地方：定义了一个encoder，这是用来给用户的密码加密的东西，配置里定义了什么User类（或者接口）使用什么样的密码加密算法。FOSUserBundle的例子使用的是sha256，其实常见的sha1，md5，都是可以的，但是因为彩虹表（可以搜搜看是什么东西）的存在，我推荐bcrypt。

这里强烈建议大家还是看看第四天的内容，里面提到的一些概念的东西还是值得去了解和体会的

另外为了让FOSUserBundle知道我们自己定义的User类，我们还得配置一些东西：

```
1 # app/config/config.yml
2 fos_user:
3     db_driver: orm
4     firewall_name: secured_area
5     user_class: AppBundle\Entity\User
6
```

到此为止我们就可以创建Mysql的User表了

```
1 $ php app/console doctrine:schema:update --force
2
```

友情提示：在使用--force更新数据库之前，最好改用--dump-sql参数确认一下会运行的SQL

如同HWIOAuthBundle，FOSUserBundle里也自带了现成的路由以及控制器代码，我们现在将他们引入进来：

```
1 # app/config/routing.yml
2 fos_user:
3     resource: "@FOSUserBundle/Resources/config/routing/all.xml"
```

```
4
```

这个时候再去访问新建新闻链接/news/，将会跳转到/login去登录。当然，你也可以直接访问/register去添加一个新的用户。不出意外，你已经实现了用户“注册/登录/权限控制”的功能。

在测试登录之前，需要创建用户账号，除了直接去注册页面注册新用户，FOSUserBundle已经提供创建用户的命令

```
1 $ php app/console fos:user:create username user@example.com passwor
2
```

与HWIOAuthBundle合体

目前很多网站，即可以选择注册新用户，又可以使用第三方登录。我们接下来也实现这个需求。如果你依然无法测试第三方登录，可以先跳过。

FOSUserBundle已经算是一个老牌第三方Bundle了，所以好多其他的Bundle也都多多少少会给使用FOSUserBundle的项目提供更多的便利。HWIOAuthBundle也是如此。首先，我们修改我们的配置文件，让HWIOAuthBundle知道我们打算将第三方登录功能和FOSUserBundle提供的用户类结合起来：

```
1  # app/config.yml
2  hwi_oauth:
3      # 与FOSUserBundle结合，并且将QQ返回的用户Id同我们创建的User类的qqId
4      fosub:
5          properties:
6              qq: qqId
7
8      # 实现User和QQ用户的绑定：如果使用QQ帐号登录，但是还没有绑定本地用户，
9      connect: ~
10
11 # app/config/security.yml
12 security:
13     providers:
14         # 第四天看过来的小伙伴注意了，这里不再需要hwi_oauth.user.provide
15         fos_user:
16             id: fos_user.user_provider.username
17
18     firewalls:
```

```
19         secured_area:
20             oauth:
21                 oauth_user_provider:
22                     # 这里需要改为支持FOSUserBundle的provider
23                     service: hwi_oauth.user.provider.fosub_bridge
24
```

之前我们设置了QQ帐号id需要同本地的User类的qqId属性绑定，我们来把它加上：

```
1  // src/AppBundle/Entity/User.php
2
3  /**
4   * @var string
5   *
6   * @ORM\Column(name="qq_id", type="string", length=50)
7   */
8  protected $qqId;
9
10 public function setQqId($qqId)
11 {
12     $this->qqId = $qqId;
13
14     return $this;
15 }
16
17 public function getQqId()
18 {
19     return $this->qqId;
20 }
21
```

至此，我们就可以通过QQ登录并且创建一个绑定QQ帐号的本地用户了。


header.register
form.email
form.username
谜星-夜
form.password
form.password_confirmation
connect.registration.submit
connect.registration.cancel