# xadmin

## xadmin的全局配置

将全局配置修改:

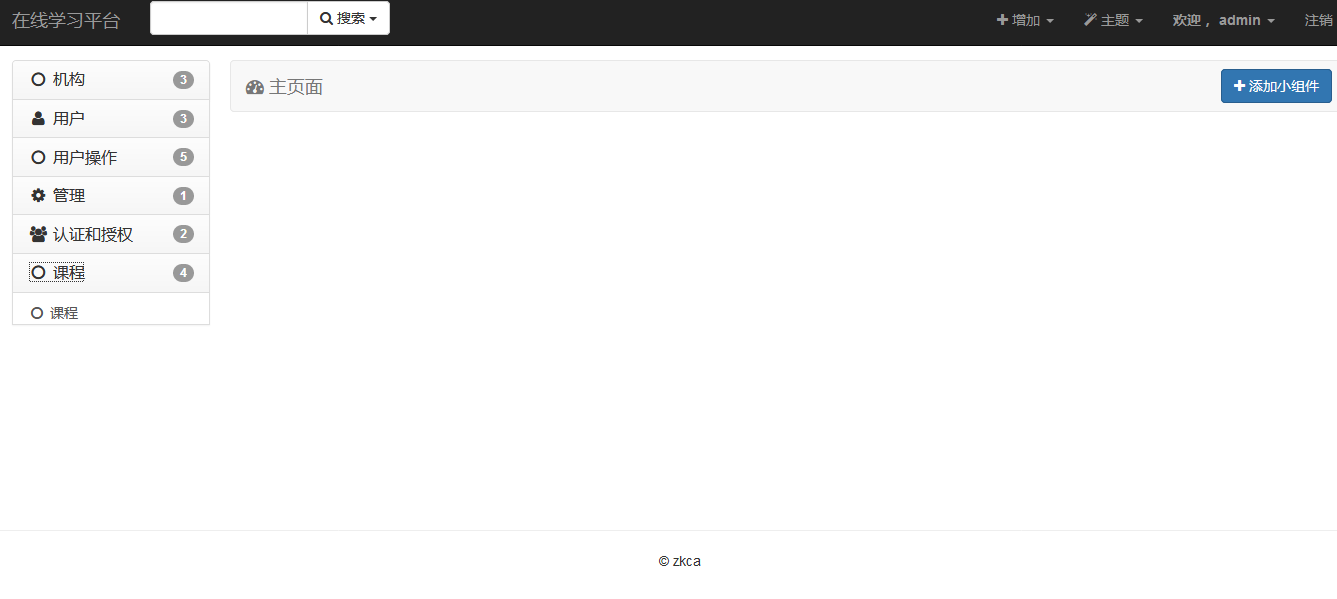
* 如左上角：django Xadmin。下面的我的公司
* 主题修改，app名称汉化，菜单收叠。

**使用Xadmin的主题功能。**

把全站的配置放在users\adminx.py中:

### 1.1添加主题功能

1. *# 创建xadmin的最基本管理器配置，并与view绑定*class BaseSetting(object):  
    *# 开启主题功能* enable\_themes = True  
    use\_bootswatch = True  
     
     
   *# 全局修改，固定写法*class GlobalSetting(object):  
    *# 修改title* site\_title = "在线学习平台"  
    *# 修改footer* site\_footer = 'zkca'  
    *# 收起菜单* menu\_style = 'accordion'  
     
     
   xadmin.site.register(EmailVerifyRecord, EmailVerifyRecordAdmin)  
   xadmin.site.register(Banner, BannerAdmin)  
     
   *# 将基本配置管理与view绑定*xadmin.site.register(views.BaseAdminView, BaseSetting)  
   *# 将title和footer信息进行注册*xadmin.site.register(views.CommAdminView, GlobalSetting)



## 修改app的名字

在apps.py里面配置app的显示名称:

from django.apps import AppConfig  
  
  
class CourseConfig(AppConfig):  
 name = 'course'  
 verbose\_name = '课程'

还要在users/\_\_init\_\_.py中引用apps.py的配置

添加代码如下：

default\_app\_config = 'course.apps.CourseConfig'

# Django部分

## 增加邮箱登录

让用户可以通过邮箱或者用户名都可以登录，用自定义authenticate方法这里是继承ModelBackend类来做的验证

#settings.py

AUTH\_USER\_MODEL = 'users.UserProfile'

#views.py

*# 邮箱和用户名都可以登录*class CustomBackend(ModelBackend):  
 def authenticate(self, request, username=None, password=None, \*\*kwargs):  
 try:  
 *# 不希望用户存在两个，get只能有一个。两个是get失败的一种原因 Q为使用并集查询* user = UserProfile.objects.get(Q(username=username) | Q(email=username))  
 *# django的后台中密码加密：所以不能password==password  
 # UserProfile继承的AbstractUser中有def check\_password(self, raw\_password):* if user.check\_password(password):  
 return user  
 except Exception as e:  
 return None

## 用form实现登录

1. 把前面views中的user\_login()函数改成基于类的形式, 基于类的urls配置
2. users下新建form.py文件

from django import forms  
  
  
*# 登录表单验证*class LoginForm(forms.Form):  
 *# 用户名密码不能为空* username = forms.CharField(required=True)  
 password = forms.CharField(required=True, min\_length=5)

1. 定义好forms后利用它来做验证，并完善错误提示信息

#views.py

*# 利用表单进行验证*def post(self, request):  
 *# 实例化* login\_form = LoginForm(request.POST)  
 if login\_form.is\_valid():  
 *# 获取提交的用户名密码* user\_name = request.POST.get('username', None)  
 pass\_word = request.POST.get('password', None)  
 *# 成功返回user对象,失败None* user = authenticate(username=user\_name, password=pass\_word)  
 *# 如果不是Null，说明验证成功* if user is not None:  
 *# login* login(request, user)  
 return render(request, 'index.html')  
 *# 只有当验证不通过才返回数据到前端* else:  
 return render(request, 'login.html', {'msg': '用户名或密码错误！', 'login\_form': login\_form})  
 *# form.is\_valid()已经判断不合法了，所以这里不需要返回错误信息到前端* else:  
 return render(request, 'login.html', {'login\_form': login\_form})

## 3 . URL配置和松耦合

1. 传统模式

提交URL地址固定耦合在action里

<form action=**"/**register**/"** method=**"post"** autocomplete=**"off"**>

1. 优化，松耦合