# CRM

# 1.快速实现简单的权限控制

1.登录页面是否有权限访问

2.POST请求，用户登录检验是否合法。

3.获取当前用户相关的所有权限并放入session

4.再次向服务器发起请求，后端编写中间件对用户当前访问的url进行权限的校验(查看是否在session中)

知识点：

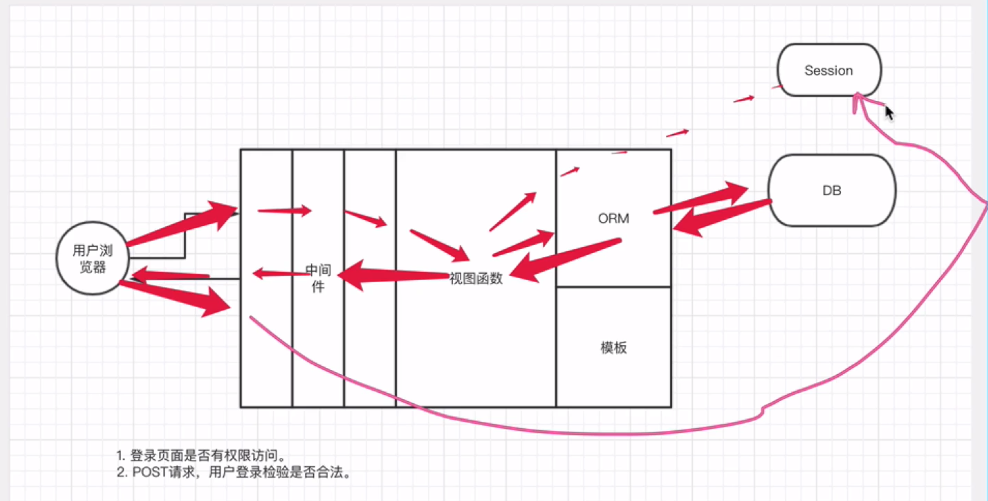
1.templates,static查找顺序

首先查找最外层的templates

然后根据app的注册顺序进行查找

2.{}.get(key)key不存在时，不会报错，返回值为None

3.中间件return None会继续运行views.py



# 2.功能完善

将权限相关的功能放到rbac应用下，以便于以后组件的应用。

# 3.动态菜单

一级菜单:如何实现动态显示一级菜单

a.表结构修改

b.获取菜单信息存到session中

c.模板中显示session数据

知识点：inclusion\_tag

二级菜单：

menu\_list数据结构

*# {1: {'title': '信息管理', 'icon': 'fa-fire', 'children': [{'title': '客户列表', 'url': '/customer/list/'}]},  
# 2: {'title': '用户管理', 'icon': 'fa-fire', 'children': [{'title': '账单列表', 'url': '/payment/list/'}]}}*

# 4.点击非菜单的按钮时，默认选中或默认展开

当点击某个不能成为菜单的选项时，设置菜单的默认选中和展开

1)登录：做权限和菜单的初始化：

菜单信息：

*# {1: {'title': '信息管理', 'icon': 'fa-fire', 'children': [{'title': '客户列表', 'url': '/customer/list/'}]},  
# 2: {'title': '用户管理', 'icon': 'fa-fire', 'children': [{'title': '账单列表', 'url': '/payment/list/'}]}}*

权限信息：

*# [{'id': 1, 'url': '/customer/list/', 'pid': None}, {'id': 2, 'url': '/customer/add/', 'pid': 1},  
# {'id': 7, 'url': '/payment/list/', 'pid': None}, {'id': 8, 'url': '/payment/add/', 'pid': 7}]*

2)再次访问

中间件进行权限的校验(根据权限信息)

获取ID或PID(应该被选中的可以做菜单的权限ID)

**for** permission\_item **in** session\_permission\_url\_list:  
 **if** re.match(permission\_item[**"url"**], current\_path):  
 flag = **True** request.current\_selected\_permission = permission\_item[**'pid'**] **or** permission\_item[**'id'**]

模板中使用inclusion\_tag生成动态菜单(根据菜单信息动态生成)

**if** per[**'id'**] == request.current\_selected\_permission:  
 per[**'class'**] = **'active'** *# 父菜单去掉hide* val[**'class'**] = **''**

# 5.路径导航

## 5.1中间件中

*# 用于路径导航*url\_record = [{**'title'**: **"首页"**, **"url"**: **"#"**}]  
*# [{'title': '首页', 'url': '#'},  
# {'title': '客户列表', 'url': '/customer/list/'},  
# {'title': '添加客户', 'url': '/customer/add/', 'class': 'active'}]***for** permission\_item **in** session\_permission\_url.values():  
 **if** re.match(permission\_item[**"url"**], current\_path):  
 flag = **True** request.current\_selected\_permission = permission\_item[**'pid'**] **or** permission\_item[**'id'**]  
 **if not** permission\_item[**"pid"**]:  
 url\_record.append({**'title'**: permission\_item[**'title'**],  
 **'url'**: permission\_item[**"url"**],  
 **'class'**: **'active'**})  
 **else**:  
 url\_record.extend([{**'title'**: permission\_item[**'p\_title'**],  
 **'url'**: permission\_item[**"p\_url"**]},  
 {**'title'**: permission\_item[**'title'**],  
 **'url'**: permission\_item[**"url"**],  
 **'class'**: **'active'**}])  
 request.breadcrumb = url\_record

## 5.2inclusion\_tag中

@register.inclusion\_tag(**"rbac/breadcrumb.html"**)  
**def** breadcrumb(request):  
 **return** {**'record\_list'**: request.breadcrumb}

## 5.3html中

<**div**>  
 <**ol class="breadcrumb no-radius no-margin" style="border-bottom**: 1**px solid #ddd**;**"**>  
 {% **for item in record\_list** %}  
 {% **if item**.**class** %}  
 <**li class="active"**>{{ **item**.**title** }}</**li**>  
 {% **else** %}  
 <**li**><**a href="**{{ **item**.**url** }}**"**>{{ **item**.**title** }}</**a**></**li**>  
 {% **endif** %}  
 {% **endfor** %}  
 </**ol**>  
</**div**>

# 6.力度控制到按钮级别

permission\_dict = {}  
  
*# {'customer\_list': {'id': 1, 'title': '客户列表',   
# 'url': '/customer/list/', 'pid': None, 'p\_title': None, 'p\_url': None},   
# 'customer\_add': {'id': 2, 'title': '添加客户',   
# 'url': '/customer/add/', 'pid': 1, 'p\_title': '客户列表', 'p\_url': '/customer/list/'},   
# 'payment\_list': {'id': 7, 'title': '账单列表',   
# 'url': '/payment/list/', 'pid': None, 'p\_title': None, 'p\_url': None},   
# 'payment\_add': {'id': 8, 'title': '添加账单',   
# 'url': '/payment/add/', 'pid': 7, 'p\_title': '账单列表', 'p\_url': '/payment/list/'}}*

menu\_dict = {}  
*# {1: {'title': '信息管理', 'icon': 'fa-fire', 'children': [{'title': '客户列表', 'url': '/customer/list/'}]},  
# 2: {'title': '用户管理', 'icon': 'fa-fire', 'children': [{'title': '账单列表', 'url': '/payment/list/'}]}}*

## 6.1过滤器中

@register.filter  
**def** has\_permission(request, name):  
 *"""  
 判断是否有权限* **:param** *request:* **:param** *name:* **:return***:  
 """* **if** name **in** request.session[settings.SESSION\_PERMISSION\_URL]:  
 **return True**

## 6.2html中

{% **if request**|**has\_permission**:**"payment\_del" or request**|**has\_permission**:**"payment\_edit"** %}  
 <**td**>  
 {% **if request**|**has\_permission**:**"payment\_edit"** %}  
 <**a style="color**: **#333333**;**" href="/payment/edit/**{{ **row**.**id** }}**/"**>  
 <**i class="fa fa-edit" aria-hidden="true"**></**i**></**a**>  
 {% **endif** %}  
 {% **if request**|**has\_permission**:**"payment\_del"** %}  
 <**a style="color**: **#d9534f**;**" href="/payment/del/**{{ **row**.**id** }}**/"**><**i  
 class="fa fa-trash-o"**></**i**></**a**>  
 {% **endif** %}  
 </**td**>  
{% **endif** %}

# 7.权限分配

## 7.1角色管理

知识点：

ModelForm

根据namespace和name反向生成URL

模板的查找顺序

## 7.2用户管理

知识点：

ModelForm

字段的自定制

钩子方法

错误提示(中文)

重写init方法，统一给所有字段添加属性(form-control)

### 7.2.1modelForm

*# modelForm中的报错信息显示中文*LANGUAGE\_CODE = **'zh-hans'**

**class** UserModelForm(forms.ModelForm):  
 *# 自己新加的字段* confirm\_password = forms.CharField(label=**'确认密码'**)  
  
 **class** Meta:  
 model = models.UserInfo  
 *# 只想添加和编辑这些字段* fields = [**'name'**, **'email'**, **'password'**, **'confirm\_password'**]  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(UserModelForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 *# 统一给ModelForm生成字段添加样式* **for** name, field **in** self.fields.items():  
 field.widget.attrs[**'class'**] = **'form-control'  
  
 def** clean\_confirm\_password(self):  
 *"""  
 局部钩子  
 检测密码是否一致* **:return***:  
 """* password = self.cleaned\_data[**'password'**]  
 confirm\_password = self.cleaned\_data[**'confirm\_password'**]  
 **if** password != confirm\_password:  
 **raise** ValidationError(**'两次密码输入不一致'**)  
 **return** confirm\_password

## 7.3菜单和权限管理

### 7.3.1一级菜单的展示和选择

row.id是数字类型，munu\_id是字符串

模板中整型转换为字符串类型

**row**.**id**|**safe转换为字符串**

<**tr class="**{% **if row**.**id**|**safe** == **menu\_id** %}**active**{% **endif** %}**"**>

### 7.3.2保留原搜索条件(QueryDict()使用)

添加，编辑之后，原页面样式不变，保留原来的url中的内容

进入添加页面(django-admin源码的方式):

[http://127.0.0.1:8000/rbac/menu/edit/2/?\_filter=mid%3D2](http://127.0.0.1:8000/rbac/menu/edit/2/?_filter=mid=2)

url为：/menu/add/?\_filter="mid=2&age=27"

提交：

跳转回:/rbac/menu/list/?mid=2&age=27

**def** memory\_url(request, name, \*args, \*\*kwargs):  
 *"""  
 生成带有原搜索条件的URL（替代了模板中的url）* **:param** *request:* **:param** *name:* **:return***:  
 """* basic\_url = reverse(name, args=args, kwargs=kwargs)  
  
 *# 当前URL中无参数,即没有?mid=2* **if not** request.GET:  
 **return** basic\_url  
  
 query\_dict = QueryDict(mutable=**True**)  
 query\_dict[**'\_filter'**] = request.GET.urlencode() *# 获取路径后的参数mid=2&age=99  
 # 不能直接使用字符串的拼接  
 # 因为/menu/add/\_filter=mid=2&age=27这样就是\_filter=mid=2，而不是\_filter=""mid=2&age=27,含义就变了  
 # 所以这里使用query\_dict进行打包，然后给key赋值  
 # query\_dict.urlencode()以url的编码格式返回数据字符串  
 # 在HttpRequest对象中，GET和POST属性都是一个django.http.QueryDict的实例  
 # QueryDict 实现了Python字典数据类型的所有标准方法，因为它是字典的子类。* **return "%s?%s"** % (basic\_url, query\_dict.urlencode())

### 7.3.3二级菜单

内容过长，导致样式过丑，解决方法

<**td colspan="2" style="border-top**: 0**"**>

<**tr class="**{% **if row**.**id**|**safe** == **second\_menu\_id** %}**active** {% **endif** %}**"**>  
 <**td rowspan="2"**>  
 <**a href="?mid=**{{ **menu\_id** }}**&sid=**{{ **row**.**id** }}**"**>{{ **row**.**title** }}</**a**>  
 </**td**>  
 <**td**>{{ **row**.**name** }}</**td**>  
 <**td**>  
 <**a style="color**: **#333333**;**" href="**{% **memory\_url request 'rbac:second\_menu\_edit' pk**=**row**.**id** %}**"**>  
 <**i class="fa fa-edit" aria-hidden="true"**></**i**></**a**>  
  
  
 <**a style="color**: **#d9534f**;**" href="**{% **memory\_url request 'rbac:second\_menu\_del' pk**=**row**.**id** %}**"**><**i  
 class="fa fa-trash-o"**></**i**></**a**>  
  
 </**td**>  
</**tr**>  
<**tr class="**{% **if row**.**id**|**safe** == **second\_menu\_id** %}**active** {% **endif** %}**"**>  
 <**td colspan="2" style="border-top**: 0**"**>{{ **row**.**url** }}</**td**>  
</**tr**>

### 7.3.4二级菜单增删改查

#### ModelForm设置某个字段的默认值

initial={**'menu'**: menu\_object}

**class** BootStrapModelForm(forms.ModelForm):  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(BootStrapModelForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 *# 统一给ModelForm生成字段添加样式* **for** name, field **in** self.fields.items():  
 field.widget.attrs[**'class'**] = **'form-control'**

**class** SecondMenuModelForm(BootStrapModelForm):  
 **class** Meta:  
 model = models.Permission  
 exclude = [**'pid'**]

**if** request.method == **'GET'**:  
 form = SecondMenuModelForm(initial={**'menu'**: menu\_object})  
 **return** render(request, **'rbac/change.html'**, {**'form'**: form})

#### ModelForm在save之前，设置关联字段的值

form.instance就是Permission对象的一个实例

form.instance.pid = second\_menu\_object

**if** form.is\_valid():  
 second\_menu\_object = models.Permission.objects.filter(id=second\_menu\_id).first()  
 **if not** second\_menu\_object:  
 **return** HttpResponse(**'二级菜单不存在，请重新选择！'**)  
 form.instance.pid = second\_menu\_object  
 form.save()  
 **return** redirect(memory\_reverse(request, **'rbac:menu\_list'**))

## 7.4formset

<https://github.com/aawuliliaa/python-test/tree/master/formset>

1.什么是formset?

form组件或ModelForm用于做一个表单验证。

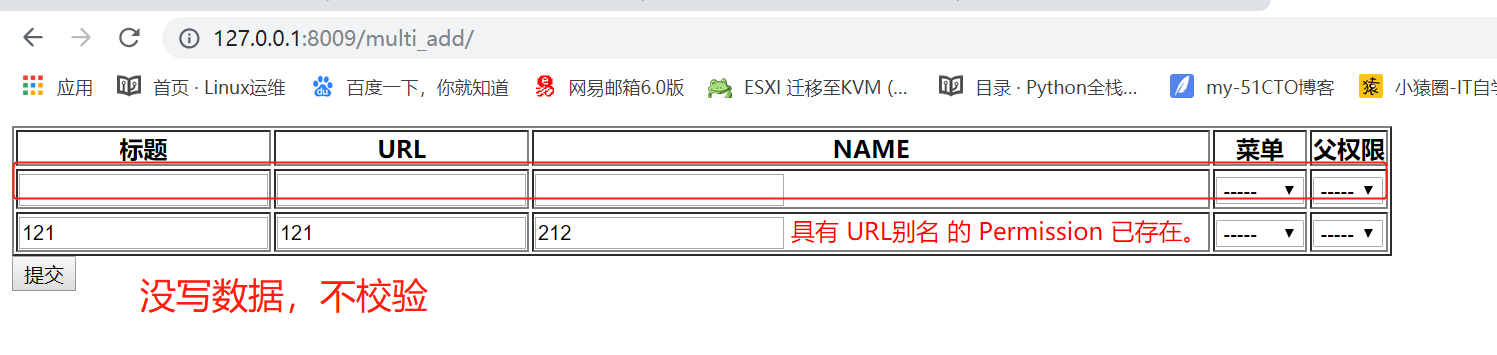
formset用于做多个表单验证的组件。

2.应用场景？

批量操作

3.如何使用formset?

*只有写了数据，才会进行校验  
没写数据的行不会校验*



## 7.5自动发现项目中的URL(django 2.2.3版本)

递归的使用

当前项目中有哪些URL及name

*# <URLResolver <URLPattern list> (admin:admin) 'admin/'>  
# <URLResolver <module 'web.urls' from '\*\*permission\\web\\urls.py'> (web:web) '^'>  
# <URLResolver <module 'rbac.urls' from '\*\*permission\\rbac\\urls.py'> (rbac:rbac) 'rbac/'>  
# <URLPattern '^user/list/$' [name='user\_list']>  
# urlpatterns = [  
# path('admin/', admin.site.urls),  
# re\_path(r'^', include('web.urls', namespace="web")),  
# re\_path(r'rbac/', include('rbac.urls', namespace="rbac")),  
# re\_path(r'^user/list/$', user.user\_list, name='user\_list'),  
# ]  
# for item in md.urlpatterns:  
# print(item)*

获取URL的方式

**or** item **in** urlpatterns:  
 **if** isinstance(item, URLPattern): *# 非路由分发，讲路由添加到url\_ordered\_dict  
 # 没有设置name别名，就跳过* **if not** item.name:  
 **continue  
  
 if** pre\_namespace:  
 name = **"%s:%s"** % (pre\_namespace, item.name)  
 **else**:  
 name = item.name

*# 下面是获取配置的url,下面是调试之后知道的*

pattern = **""  
 if** isinstance(item.pattern, URLPattern) **or** isinstance(item.pattern, RoutePattern):  
 pattern = item.pattern.\_route  
 **elif** isinstance(item.pattern, RegexPattern):  
 pattern = item.pattern.\_regex  
 url = pre\_url + pattern *# /rbac/user/edit/(?P<pk>\d+)/  
 # 起始符和终止符去掉* url = url.replace(**'^'**, **''**).replace(**'$'**, **''**)

## 7.6批量操作权限

获取项目中所有url set1

到数据库中获取已经录入的所有权限 set2

情况一：自动发现有 数据库无 实现批量添加

set1 - set2 添加

情况二：数据库多了 自动发现的少 实现批量删除

set2 -set1 删除

情况三：自动发现和数据库的数量一致，但url相关信息改了，实现批量更新

set3 = set1 & set2 取交集，更新

添加数据的时候，可以使用bulk\_create()一次性批量添加，避免多次save()造成的消耗数据库性能。

## 7.7权限分配

展示用户，角色，权限信息

选择用户、角色时，页面上设置权限默认值

角色和权限分配[保存]

prop()，removeProp()只用于单选框中获取**checked**值，值为true或false

## 7.8使用说明

*#!/usr/bin/env python  
# -\*- coding:utf-8 -\*-  
"""  
RBAC组件的使用文档  
  
1. 将rbac组件拷贝项目。  
  
  
2. 将rbac/migrations目录中的数据库迁移记录删除  
  
  
3. 业务系统中用户表结构的设计  
  
 业务表结构中的用户表需要和rbac中的用户有继承关系，如：  
  
 rbac/models.py  
 class UserInfo(models.Model):  
 # 用户表  
 name = models.CharField(verbose\_name='用户名', max\_length=32)  
 password = models.CharField(verbose\_name='密码', max\_length=64)  
 email = models.CharField(verbose\_name='邮箱', max\_length=32)  
 roles = models.ManyToManyField(verbose\_name='拥有的所有角色', to=Role, blank=True) # 严重提醒 Role 不要加引号，*

*#因为加了引号，会从当前py文件中找Role表*

*# 如果这样，我们再继承的时候，会报错  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 class Meta:  
 # django以后再做数据库迁移时，不再为UserInfo类创建相关的表以及表结构了。  
 # 此类可以当做"父类"，被其他Model类继承。  
 abstract = True  
  
 业务/models.py  
 class UserInfo(RbacUserInfo):  
 phone = models.CharField(verbose\_name='联系方式', max\_length=32)  
 level\_choices = (  
 (1, 'T1'),  
 (2, 'T2'),  
 (3, 'T3'),  
 )  
 level = models.IntegerField(verbose\_name='级别', choices=level\_choices)  
  
 depart = models.ForeignKey(verbose\_name='部门', to='Department')  
对rbac中代码的修改*

*在URL中将用户表的增删改查和修改密码功能删除*

re\_path(**r'^user/list/$'**, user.user\_list, name=**'user\_list'**),  
re\_path(**r'^user/add/$'**, user.user\_add, name=**'user\_add'**),  
re\_path(**r'^user/edit/(?P<pk>\d+)/$'**, user.user\_edit, name=**'user\_edit'**),  
re\_path(**r'^user/del/(?P<pk>\d+)/$'**, user.user\_del, name=**'user\_del'**),  
re\_path(**r'^user/reset/password/(?P<pk>\d+)/$'**, user.user\_reset\_pwd, name=**'user\_reset\_pwd'**),

*4. 讲业务系统中的用户表的路径写到配置文件。  
  
 # 业务中的用户表  
 RBAC\_USER\_MODLE\_CLASS = "app01.models.UserInfo"  
  
 用于在rbac分配权限时，读取业务表中的用户信息。*

*rbac/views/menu.py中使用userInfo表的方式  
 # 业务中的用户表 "app01.models.UserInfo""* user\_model\_class = import\_string(settings.RBAC\_USER\_MODLE\_CLASS)  
  
 user\_object = user\_model\_class.objects.filter(id=user\_id).first()

*5. 业务逻辑开发  
 将所有的路由都设置一个name，如：  
 url(r'^login/$', account.login, name='login'),  
 url(r'^logout/$', account.logout, name='logout'),  
  
 url(r'^index/$', account.index, name='index'),  
  
 url(r'^user/list/$', user.user\_list, name='user\_list'),  
 url(r'^user/add/$', user.user\_add, name='user\_add'),  
 url(r'^user/edit/(?P<pk>\d+)/$', user.user\_edit, name='user\_edit'),  
 url(r'^user/del/(?P<pk>\d+)/$', user.user\_del, name='user\_del'),  
 url(r'^user/reset/password/(?P<pk>\d+)/$', user.user\_reset\_pwd, name='user\_reset\_pwd'),  
  
 url(r'^host/list/$', host.host\_list, name='host\_list'),  
 url(r'^host/add/$', host.host\_add, name='host\_add'),  
 url(r'^host/edit/(?P<pk>\d+)/$', host.host\_edit, name='host\_edit'),  
 url(r'^host/del/(?P<pk>\d+)/$', host.host\_del, name='host\_del'),  
 用于反向生成URL以及粒度控制到按钮级别的权限控制。  
  
6. 权限信息录入  
 - 在url中添加rbac的路由分发,注意：必须设置namespace  
 urlpatterns = [  
 ...  
 url(r'^rbac/', include('rbac.urls', namespace='rbac')),  
  
 ]  
  
 - rbac提供的地址进行操作  
 - http://127.0.0.1:8000/rbac/menu/list/  
 - http://127.0.0.1:8000/rbac/role/list/  
 - http://127.0.0.1:8000/rbac/distribute/permissions/  
  
 相关配置：自动发现URL时，排除的URL：  
  
 # 自动化发现路由中URL时，排除的URL  
 AUTO\_DISCOVER\_EXCLUDE = [  
 '/admin/.\*',  
 '/login/',  
 '/logout/',  
 '/index/',  
 ]  
  
  
7. 编写用户登录的逻辑【进行权限初始化】  
  
 from django.shortcuts import render, redirect  
 from app01 import models  
 from rbac.service.init\_permission import init\_permission  
  
  
 def login(request):  
 if request.method == 'GET':  
 return render(request, 'login.html')  
  
 user = request.POST.get('username')  
 pwd = request.POST.get('password')  
  
 user\_object = models.UserInfo.objects.filter(name=user, password=pwd).first()  
 if not user\_object:  
 return render(request, 'login.html', {'error': '用户名或密码错误'})  
  
 # 用户权限信息的初始化  
 init\_permission(user\_object, request)  
  
 return redirect('/index/')  
  
  
 相关配置: 权限和菜单的session key：  
  
 setting.py  
 PERMISSION\_SESSION\_KEY = "permission\_url\_list\_key"  
 MENU\_SESSION\_KEY = "permission\_menu\_key"  
  
8. 编写一个首页的逻辑  
  
 def index(request):  
 return render(request, 'index.html')  
  
  
 相关配置：需要登录但无需权限的URL  
  
 # 需要登录但无需权限的URL  
 NO\_PERMISSION\_LIST = [  
 '/index/',  
 '/logout/',  
 ]  
  
9. 通过中间件进行权限校验  
  
 # 权限校验  
 MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
 'rbac.middlewares.rbac.RbacMiddleware',  
 ]  
  
 # 白名单，无需登录就可以访问* WHITE\_LIST *= [  
 '/login/',  
 '/admin/.\*'  
 ]  
  
  
10. 粒度到按钮级别的控制  
  
 {% extends 'layout.html' %}  
 {% load rbac %}  
  
 {% block content %}  
 <div class="luffy-container">  
 <div class="btn-group" style="margin: 5px 0">  
  
 {% if request|has\_permission:'host\_add' %}  
 <a class="btn btn-default" href="{% memory\_url request 'host\_add' %}">  
 <i class="fa fa-plus-square" aria-hidden="true"></i> 添加主机  
 </a>  
 {% endif %}  
  
 </div>  
 <table class="table table-bordered table-hover">  
 <thead>  
 <tr>  
 <th>主机名</th>  
 <th>IP</th>  
 <th>部门</th>  
 {% if request|has\_permission:'host\_edit' or request|has\_permission:'host\_del' %}  
 <th>操作</th>  
 {% endif %}  
  
 </tr>  
 </thead>  
 <tbody>  
 {% for row in host\_queryset %}  
 <tr>  
 <td>{{ row.hostname }}</td>  
 <td>{{ row.ip }}</td>  
 <td>{{ row.depart.title }}</td>  
 {% if request|has\_permission:'host\_edit' or request|has\_permission:'host\_del' %}  
 <td>  
 {% if request|has\_permission:'host\_edit' %}  
 <a style="color: #333333;" href="{% memory\_url request 'host\_edit' pk=row.id %}">  
 <i class="fa fa-edit" aria-hidden="true"></i></a>  
 {% endif %}  
 {% if request|has\_permission:'host\_del' %}  
 <a style="color: #d9534f;" href="{% memory\_url request 'host\_del' pk=row.id %}"><i  
 class="fa fa-trash-o"></i></a>  
 {% endif %}  
 </td>  
 {% endif %}  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
 </div>  
  
 {% endblock %}  
  
  
  
  
总结，目的是希望在任意系统中应用权限系统。  
 - 用户登录 + 用户首页 + 用户注销 业务逻辑  
 - 项目业务逻辑开发  
 注意：开发时候灵活的去设置layout.html中的两个inclusion\_tag  
 <div class="pg-body">  
 <div class="left-menu">  
 <div class="menu-body">  
 {% multi\_menu request %} # 开发时，去掉；上线时，取回。  
 </div>  
 </div>  
 <div class="right-body">  
 <div>  
 {% breadcrumb request %} # 开发时，去掉；上线时，取回。  
 </div>  
 {% block content %} {% endblock %}  
 </div>  
 </div>  
 - 权限信息的录入  
 - 配置文件  
 # 注册APP  
 INSTALLED\_APPS = [  
 ...  
 'rbac.apps.RbacConfig'  
 ]  
 # 应用中间件  
 MIDDLEWARE = [  
 ...  
 'rbac.middlewares.rbac.RbacMiddleware',  
 ]  
  
 # 业务中的用户表  
 RBAC\_USER\_MODLE\_CLASS = "app01.models.UserInfo"  
 # 权限在Session中存储的key  
 PERMISSION\_SESSION\_KEY = "luffy\_permission\_url\_list\_key"  
 # 菜单在Session中存储的key  
 MENU\_SESSION\_KEY = "luffy\_permission\_menu\_key"  
  
 # 白名单  
 VALID\_URL\_LIST = [  
 '/login/',  
 '/admin/.\*'  
 ]  
  
 # 需要登录但无需权限的URL  
 NO\_PERMISSION\_LIST = [  
 '/index/',  
 '/logout/',  
 ]  
  
 # 自动化发现路由中URL时，排除的URL  
 AUTO\_DISCOVER\_EXCLUDE = [  
 '/admin/.\*',  
 '/login/',  
 '/logout/',  
 '/index/',  
 ]  
  
 - 粒度到按钮级别的控制  
  
  
  
  
  
  
"""*

# 8.stark组件

介绍

stark组件，是一个帮助开发者快速实现数据库表的增删改查

目标:

10s完成一张表的增删改查。

## 8.1django项目启动时，自定义执行某个py文件

django启动时即python3 manage.py runserver，路由加载之前，自定义执行某个py文件

settings.py

INSTALLED\_APPS = [  
 **'django.contrib.admin'**,  
 **'django.contrib.auth'**,  
 **'django.contrib.contenttypes'**,  
 **'django.contrib.sessions'**,  
 **'django.contrib.messages'**,  
 **'django.contrib.staticfiles'**,  
 **'rbac'**,  
 **'web.apps.WebConfig'**, #这里一定要这样注册app，否则不会自动执行**auto\_discover.**py文件  
]

**apps.py**

**from** django.apps **import** AppConfig  
  
**from** django.utils.module\_loading **import** autodiscover\_modules  
**class** WebConfig(AppConfig):  
 name = **'web'  
 def** ready(self):  
 autodiscover\_modules(**"auto\_discover"**)

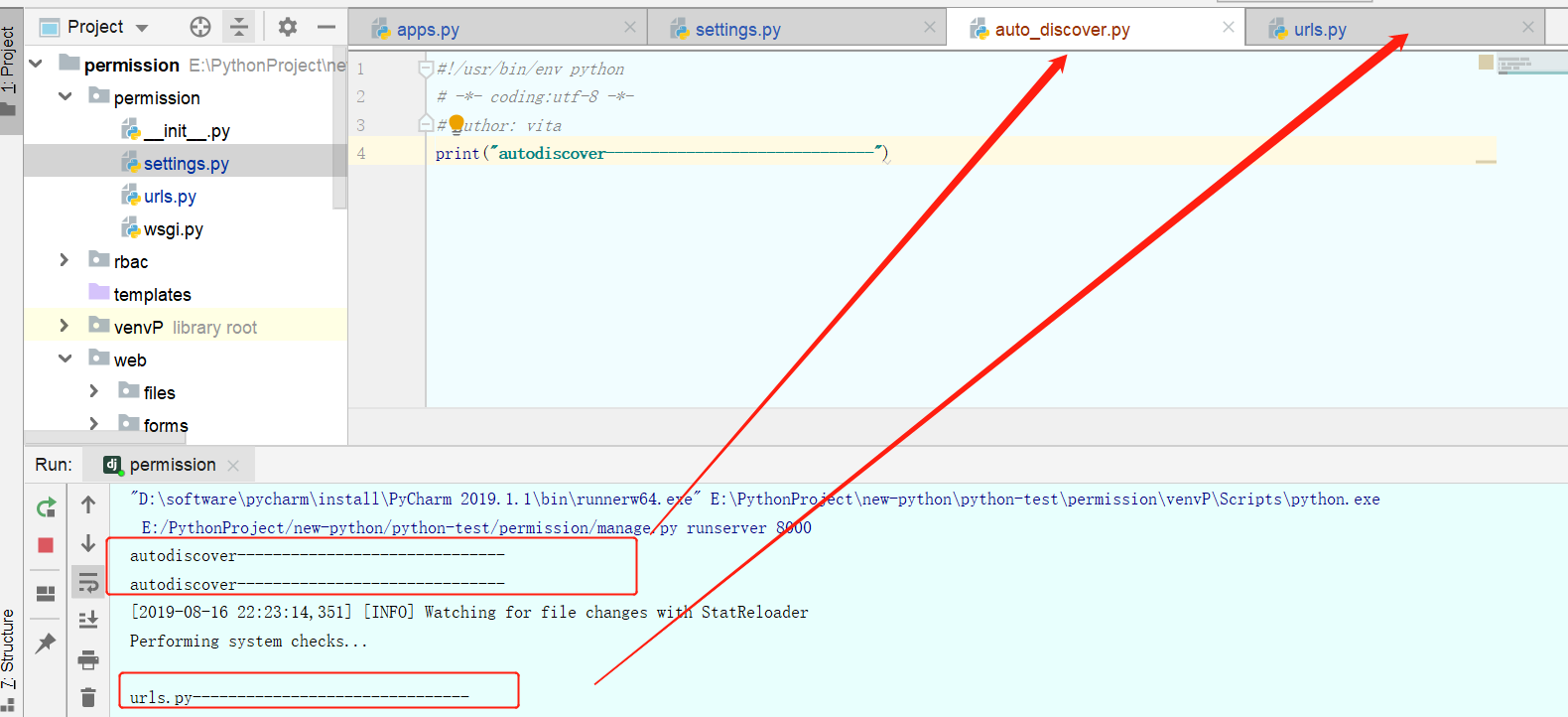
*# 在任意app的apps.py中的Config类中定义ready方法，并调用autodiscover\_modules*

*# django在启动时，就会去已注册的(---所有---)app的目录下找  
# auto\_discover.py  
# 并自动导入。  
#   
#   
# 如果执行两次，是因为django内部自动重启导致：  
# python manage.py runserver 120.0.0.1: 8001 --noreload*

*使用--noreload就执行一次了*

*# (venvP) E:\PythonProject\new-python\python-test\permission>python manage.py runserver 8000 --noreload  
# autodiscover------------------------------  
# Performing system checks...  
#   
# urls.py-------------------------------  
# System check identified no issues (0 silenced).  
# [2019-08-16 22:29:38,664] [DEBUG] (0.001)*

*#   
# 提示：  
# 如果auto\_discover.py执行的代码向 “某个神奇的地方” 放入了一些值。之后的路由加载时，可以去“某个神奇的地方”读取到原来设置的值。*



## 8.2单例模式

### 8.2.1多例模式

*#!/usr/bin/env python  
# -\*- coding:utf-8 -\*-***class** Foo(object):  
 **pass***# 实例化了一个Foo类的对象（实例）*obj1 = Foo()  
print(obj1) *# <<\_\_main\_\_.Foo object at 0x0000020C756798D0>>*obj1.name = **'vita'***# 实例化了一个Foo类的对象（实例）*obj2 = Foo()  
print(obj2) *# <\_\_main\_\_.Foo object at 0x0000020C75679AC8>  
# print(obj2.name)*

### 8.2.2单例模式

利用python模块导入的特性，如果已经导入过的模块，再次导入时，不会创建新的对象，而是直接使用内存中的对象

这只是单例模式的一种创建方式

*utils.py*

*#!/usr/bin/env python  
# -\*- coding:utf-8 -\*-  
# Author: vita***class** AdminSite(object):  
 **pass**site = AdminSite() *# 为AdminSite类创建了一个对象（实例）*

*commons.py*

*#!/usr/bin/env python  
# -\*- coding:utf-8 -\*-  
# Author: vita***import** utils  
print(utils.site)

*apps.py*

*#!/usr/bin/env python  
# -\*- coding:utf-8 -\*-  
# Author: vita***import** utils  
print(utils.site)  
  
**import** utils  
print(utils.site)  
  
**import** commons  
  
*# 三次导入，并没有创建两个对象  
# 在python中如果已经导入过的文件再次导入时，不会重新解释一次。而是选择从内存中将原来导入的值拿来使用。  
E:\PythonProject\new-python\python-test\permission\venvP\Scripts\python.exe E:/PythonProject/new-python/python-test/permission/apps.py*

*<utils.AdminSite object at 0x0000018263F39C50>*

*<utils.AdminSite object at 0x0000018263F39C50>*

*<utils.AdminSite object at 0x0000018263F39C50>*

*Process finished with exit code 0*

*# 提示：  
#*  ***如果以后存在一个单例模式的对象，可以先在此对象中放入一个值，然后再在其他的文件中导入该对象，通过对象再次讲值获取到。***

*utils.py*

*#!/usr/bin/env python  
# -\*- coding:utf-8 -\*-  
# Author: vita***class** AdminSite(object):  
 **pass**site = AdminSite() *# 为AdminSite类创建了一个对象（实例）*

*commons.py*

*#!/usr/bin/env python  
# -\*- coding:utf-8 -\*-  
# Author: vita***import** utils  
print(utils.site.name)

*apps.py*

*#!/usr/bin/env python  
# -\*- coding:utf-8 -\*-  
# Author: vita***import** utils  
print(utils.site)  
  
**import** utils  
utils.site.name=**"vita"**print(utils.site.name)  
  
**import** commons  
  
*# 两次导入，并没有创建两个对象  
# 在python中如果已经导入过的文件再次导入时，不会重新解释一次。而是选择从内存中将原来导入的值拿来使用。  
# E:\PythonProject\new-python\python-test\permission\venvP\Scripts\python.exe E:/PythonProject/new-python/python-test/permission/apps.py  
# <utils.AdminSite object at 0x00000212F925F320>  
# vita  
# vita  
#   
# Process finished with exit code 0*

## 8.3include路由分发的本质

方式一：  
 **from** django.conf.urls **import** url,include  
  
 urlpatterns = [  
 url(**r'^web/'**, include(**"app01.urls"**)),  
 ]  
  
方式二：  
 include函数主要返回有三个元素的元组。  
 **from** django.conf.urls **import** url,include  
 **from** app01 **import** urls  
 urlpatterns = [  
 url(**r'^web/'**, (urls, app\_name, namespace)), *# 第一个参数是urls文件对象，通过此对象可以获取urls.patterns获取分发的路由。* ]  
  
  
 在源码内部，读取路由时：  
 如有第一个参数有：urls.patterns 属性，那么子路由就从该属性中后去。  
 如果第一个参数无：urls.patterns 属性，那么子路由就是第一个参数。  
  
方式三：  
 urlpatterns = [  
 url(**r'^web/'**, ([  
 url(**r'^index/'**, views.index),  
 url(**r'^home/'**, views.home),  
 ], app\_name, namespace)), *# 第一个参数是urls文件对象，通过此对象可以获取urls.patterns获取分发的路由。* ]

## 8.4上面三个知识点的整合案例

