# 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：

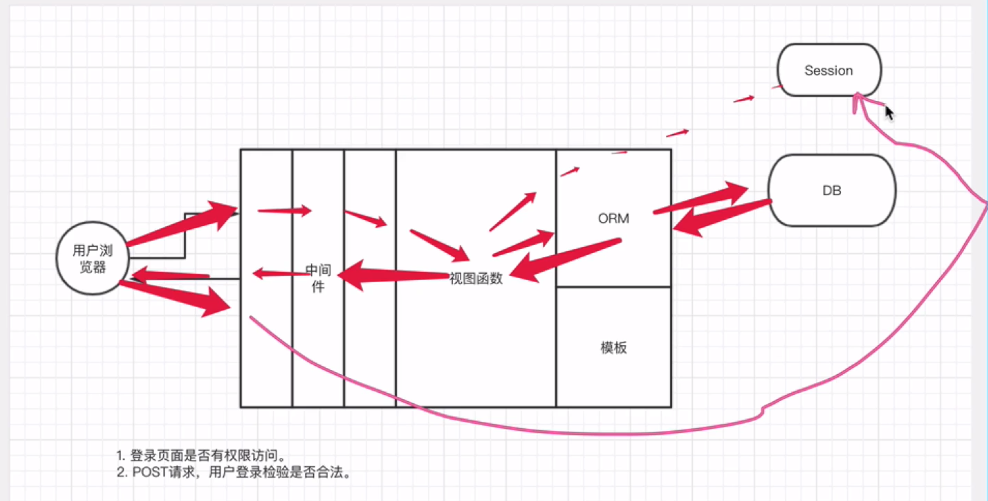
1.templates,static查找顺序

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

**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())