



flask通用rbac权限框架

原创 置顶 MrLe 最后发布于2018-05-18 13:48:13 阅读数 3732 ☆ 收藏

首先是数据库表设计models.py

```
1 from . import db
2 import datetime
3
4
5 # 用户表
6 class Users(db.Model):
7     __tablename__ = 'users'
8
9     id = db.Column(db.Integer, primary_key=True)
10    name = db.Column(db.String(32), index=True, nullable=False)
11    email = db.Column(db.String(32), unique=True, nullable=False)
12    password = db.Column(db.String(64), nullable=False)
13    ctime = db.Column(db.DateTime, default=datetime.datetime.now)
14    # 关联roles表, 一对多关系
15    role_id = db.Column(db.Integer, db.ForeignKey("roles.id"))
16    # relationship, 联查时候用到, 详细见《flask插件之flask_sqlalchemy使用》中有介绍
17    role = db.relationship("Roles", backref='user')
18    __table_args__ = (
19        # 联合唯一
20        db.UniqueConstraint('id', 'name', 'email', name='uix_id_name_email'),
21        # 联合索引
22        db.Index('ix_id_name_email', 'name', 'email'),
23    )
24
25    def __repr__(self):
26        return self.name
27
28
29 class Roles(db.Model):
30    __tablename__ = 'roles'
```



1



3



举报

```
31 id = db.Column(db.Integer, primary_key=True)
32 name = db.Column(db.String(32), index=True, nullable=True)
33 # 与生成表结构无关, 仅用于查询方便
34 permission = db.relationship('Permission', secondary='roles2permission', backref='role')
35
36 def __repr__(self):
37     return self.name
38
39
40 class Permission(db.Model):
41     __tablename__ = 'permission'
42
43     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
44     name = db.Column(db.String(32), unique=True, nullable=False)
45     url = db.Column(db.String(128))
46     menu_id = db.Column(db.Integer, db.ForeignKey("menus.id"))
47
48     menu = db.relationship("Menus", backref='permission')
49
50     def __repr__(self):
51         return self.name
52
53
54 # roles和permission多对多关系, flask中没有ManyToMany用法, 多对多需要这么创建
55 class Roles2Permisson(db.Model):
56     __tablename__ = 'roles2permission'
57     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
58     role_id = db.Column(db.Integer, db.ForeignKey('roles.id'))
59     permission_id = db.Column(db.Integer, db.ForeignKey('permission.id'))
60
61
62 class Menus(db.Model):
63     __tablename__ = "menus"
64
65     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
66     name = db.Column(db.String(32), unique=True)
67     # 关联menus, 关联自己, 用于生成子菜单
68     parent = db.Column(db.Integer, db.ForeignKey("menus.id"))
69
70     def __repr__(self):
71         return self.name
```



举报

login视图函数:

```
1 from flask import Blueprint, session, redirect, url_for, render_template, request
2 from flask.views import MethodView
3 import hashlib
4
5 login = Blueprint('login', __name__, url_prefix='/login')
6
7 from .middlewares import *
8 from ..models import *
9 from .. import db
10 from .rbac import init_permission as rbac_init_permission
11
12
13 """
14 这是一个登陆认证类, cbv模式, 直接继承这个类就可以完成登陆认证, 但是后期我们把这块功能做到了中间件中, 所以这边就注释掉了
15 """
16 # class Auth(MethodView):
17 #     def dispatch_request(self, *args, **kwargs):
18 #         if not session.get('username', None):
19 #             return redirect(url_for("login.login", next=request.url_rule))
20 #         return super(Auth, self).dispatch_request(*args, **kwargs)
21
22
23 """
24 这是一个密码加密插件, 目前只支持md5方式
25 """
26 class MakePassword(object):
27     @staticmethod
28     def md5(arg):
29         hs = hashlib.md5()
30         hs.update(arg.encode('utf-8'))
31         return hs.hexdigest()
32
33
34 """
35 登陆视图函数
36 """
37 class Login(MethodView):
38
39     def get(self):
```



1



3



举报

```
40         return render_template("login/login.html")
41
42     def post(self):
43         form_data = dict(request.form)
44         """
45         form_data格式
46         form_data = {
47             'csrf_token': ["xxx"],
48             'name': ['xxx'],
49             'password': ['xxx']
50         }
51         """
52         # 通过循环form_data的key, 排除掉csrf_token
53         for k in list(form_data.keys()):
54             if k in ("csrf_token",):
55                 del form_data[k]
56             else:
57                 form_data[k] = form_data[k][0]
58         # 密码加密
59         form_data['password'] = MakePassword.md5(form_data['password'])
60         try:
61             # 获取user对象
62             user_obj = db.session.query(Users).filter_by(**form_data).first()
63             if user_obj:
64                 # 初始化权限,
65                 rbac_init_permission.INIT_PERMISSION(user_obj, db)
66                 # 把这个user对象存到session中, 注意是对象, 如有特殊需求, 存入用户名字字符串也可以
67                 session['username'] = user_obj
68                 # 登陆成功后跳转页面, next是。。。自己理解吧
69                 return redirect(request.args.get("next", url_for("index.index")))
70             # 登陆失败
71             return render_template("login/login.html", error_msg="用户名或密码错误")
72         finally:
73             db.session.close()
74
75         """
76         注销, 删除session
77         """
78
79     class Logout(MethodView):
80         def get(self):
81             session.pop('username')
```



1



3



举报

```
82         return redirect(url_for("login.login"))
83
84
85 # url routing
86 login.add_url_rule('/', view_func=Login.as_view(name="login"))
87 login.add_url_rule('/logout/', view_func=Logout.as_view(name="logout"))
```

初始化权限插件 INIT_PERMISSION

```
1 from flask import session
2 from ...models import *
3
4
5 class INIT_PERMISSION(object):
6
7     def __init__(self, user_obj, db):
8         """
9         获取当前用户权限，并写入session
10        """
11        """当前用户有权限的url"""
12        permission_url_list = [row.url for row in list(set(user_obj.role.permission))]
13        """当前用户有权限的并且显示在菜单的url"""
14        permission_menu_list = [
15            {'name': row.name,
16             'url': row.url,
17             'menu_id': row.menu_id
18            } for row in list(set(user_obj.role.permission)) if row.menu_id
19        ]
20        """菜单列表"""
21        menu_list = [
22            {'id': row.id,
23             'name': row.name,
24             'parent_id': row.parent
25            } for row in
26            db.session.query(Menus.id.label("id"), Menus.name.label("name"), Menus.parent.label("parent")).all()
27        ]
28        """写入到session"""
29        session['SESSION_PERMISSION_URL'] = permission_url_list
30        session['SESSION_PERMISSION_MENU'] = permission_menu_list
31        session['SESSION_MENU'] = menu_list
```



举报

中间件 `RbacMiddleware` , 配套的有一个settings配置文件, 里面存有白名单等 `settings.py`

```
1  URL_REGEX = '^{}$'
2
3  PASS_URL_LIST = [
4      "^/login/",
5      "^/static/",
6      "^/api",
7      "^/register/$",
8  ]

1  from flask import redirect, render_template, session, request, current_app as app, url_for
2  from . import settings
3  import re
4
5
6  class RbacMiddleware(object):
7      @staticmethod
8      def rbac_middleware():
9          # 当前请求url
10         url_rule = str(request.url_rule)
11         # 过滤白名单, 支持正则
12         for url in settings.PASS_URL_LIST:
13             if re.match(url, url_rule):
14                 return None
15         # 获取权限
16         permission_url_list = session.get("SESSION_PERMISSION_URL")
17         # 获取用户对象
18         user_obj = session.get("username")
19         # 如果其中一个没有获取到, 需要重新登陆
20         if not permission_url_list or not user_obj:
21             return redirect(url_for("login.login", next=url_rule))
22         # 如果角色为管理员, 直接跳过权限限制
23         if user_obj.role.name == "admin":
24             return None
25
26         # 定义一个标识
27         flag = False
28         for db_url in permission_url_list:
29             # pattern = ^db_url$,开头和结束, 精确匹配, 当然, 权限数据表中可以使用正则, 这里支持正则
30             pattern = settings.URL_REGEX.format(db_url)
```



1



3



举报

```
31         if re.match(pattern, url_rule):
32             flag = True # 匹配到之后, 表示改为True, 跳出循环
33             break
34
35     # 如果没有权限, 则会返回定义的页面
36     if not flag:
37         # debug模式
38         if app.config["DEBUG"]:
39             url_html = "<br/>".join(permission_url_list)
40             return "没有权限<br/>%s" % url_html
41         else:
42             return render_template('page_404.html')
43     ..
```

生成菜单的函数 `rbac_menus`

```
1 import re
2 from . import settings
3 from flask import Markup, session, request
4
5
6 class Process_menu_data(object):
7     """
8     获取菜单数据
9     """
10
11     @classmethod
12     def Menu_data(self):
13         url_rule = str(request.url_rule)
14         permission_url_list = session.get("SESSION_PERMISSION_MENU")
15         menu_list = session.get("SESSION_MENU")
16         all_menu_dict = {}
17         for row in menu_list:
18             row['children'] = []
19             row['status'] = False
20             row['open'] = False
21             all_menu_dict[row['id']] = row
22
23         for row in permission_url_list:
24             row['status'] = True
25             row['open'] = False
26             all_menu_dict[row['menu_id']]['children'].append(row)
```



举报

```
27
28     pid = row['menu_id']
29     while pid:
30         all_menu_dict[pid]['status'] = True
31         pid = all_menu_dict[pid]['parent_id']
32
33     pattern = settings.URL_REGEX.format(row['url'])
34     if re.match(pattern, url_rule):
35         row["open"] = True
36         ppid = row['menu_id']
37         while ppid:
38             all_menu_dict[ppid]['open'] = True
39             ppid = all_menu_dict[ppid]['parent_id']
40
41     result = []
42     for k, v in all_menu_dict.items():
43         if not v.get("parent_id"):
44             result.append(v)
45         else:
46             all_menu_dict[v["parent_id"]]["children"].append(v)
47     return result
48
49
50 class Process_menu_html(object):
51     """
52     获取菜单html
53     """
54
55     @classmethod
56     def Menu_html(self, menu_list):
57
58         url = """
59         <a href="{0}" class="{1}">{2}</a>
60         """
61
62         menu = """
63         <div class='rbac-menu-item'>
64             <div class='rbac-menu-header'>{0}
65                 <img src='/static/img/{3}' width="9" height="9">
66             </div>
67             <div class="rbac-menu-body {1}">{2}</div>
68         </div>
```



1



3



举报


```
69     """
70     menu_html = ""
71
72     """
73     for item in menu_list:
74         if not item['status']:
75             continue
76         if item.get("url"):
77             # 权限
78             menu_html += url.format(item["url"],
79                                     "rbac-active" if item["open"] else "",
80                                     item["name"])
81         else:
82             # 菜单
83             menu_html += menu.format(item["name"],
84                                     "" if item["open"] else "rbac-hidden",
85                                     self.Menu_html(item["children"]),
86                                     "setup_minus_lev1.gif" if item["open"] else "setup_plus_lev1.gif",
87                                     )
88     return menu_html
89
90
91 class Process_menu(object):
92     @classmethod
93     def process_menu(self):
94         menu_list = Process_menu_data.Menu_data()
95         menu_html = Process_menu_html.Menu_html(menu_list)
96         return Markup("<div class='rbac-menu-box'>%s</div>" % menu_html)
97 --
```

create_app

```
1 from flask import Flask
2 from flask_sqlalchemy import SQLAlchemy
3 from flask_session import Session
4 from flask_wtf.csrf import CSRFProtect
5 from flask_admin import Admin, AdminIndexView
6 from flask_babel import Babel
7
8
9 db = SQLAlchemy()
10 csrf = CSRFProtect()
```



举报

```
11 # 修改主页, index_view=(template="welcome.html", name='名称')
12 flask_admin = Admin(name="首页", url="/admin", index_view=AdminIndexView(name='Admin后台管理', template="admin/welcome.html"))
13 babel = Babel()
14
15 from .models import *
16 from .views.login import login
17 from .views.register import register
18 from .views.index import index
19 from .views.rbac.Middleware import *
20 from .views.rbac.RBAC_menu import Process_menu
21 from .views import admin
22
23
24 def create_app(settings_cls):
25     app = Flask(__name__)
26     app.config.from_object(settings_cls) # 加载配置文件
27     Session(app) # session写到redis
28     db.init_app(app) # 初始化数据库
29     csrf.init_app(app) # 加载csrf
30     flask_admin.init_app(app)
31     babel.init_app(app) # 国际化
32     app.register_blueprint(login)
33     app.register_blueprint(register)
34     app.register_blueprint(index)
35     # 加载中间件
36     app.before_request(RbacMiddleware.rbac_middleware)
37     # template全局函数在template中直接使用{{ make_menu() }}可以执行这个函数, 得到菜单标签, 然后自己去定义样式, 支持子菜单
38     app.add_template_global(Process_menu.process_menu, name='make_menu')
39     # admin
40     models = [(Users, '用户管理'), (Roles, '角色管理'), (Permission, "权限管理"), (Menus, "菜单管理")]
41     for model in models:
42         flask_admin.add_view(admin.MyVl(model[0], db.session, name=model[1], category='系统管理'))
43     flask_admin.add_view(admin.MyAdminView(name="test", category='test'))
44     return app
```



1



3



举报

简单样式的效果图



附送一个js

```
1 $('.rbac-menu-header').click(function () {  
2     // $(this).toggleClass('rbac-hidden');  
3     $(this).next().toggleClass('rbac-hidden').parent().siblings().find('.rbac-menu-body').addClass('rbac-hidden');  
4     $(".rbac-menu-box").find("[class='rbac-menu-header']").each(function () {  
5         if($(this).next().hasClass("rbac-hidden")){  
6             $(this).children("img:first-child").attr("src", "/static/img/setup_plus_lev1.gif");  
7         }else {  
8             $(this).children("img:first-child").attr("src", "/static/img/setup_minus_lev1.gif");  
9         }  
10     })  
11 });
```

附送一个简单的css，笔者审美很low的，css只能写到这里了

```
1 .rbac-menu-box {  
2     padding-left: 20px;  
3     font-size: 15px;  
4     border-top: solid 1px #0f0f0f;  
5 }  
6  
7 .rbac-hidden {  
8     display: none;  
9 }  
10  
11 .rbac-menu-item .rbac-menu-body {
```



举报

```
12     margin-left: 20px;
13 }
14
15 .rbac-menu-header{
16     margin-top: 10px;
17 }
18
19 .rbac-menu-body a {
20     display: block;
21 }
22
23 .rbac-active {
24     color: red;
25 }
26
27 .rbac-menu-header{
28     cursor: pointer;
29 }
--
```

1



3



点赞 1 收藏 分享 ...



MrLe

发布了25 篇原创文章 · 获赞 4 · 访问量 1万+

私信

关注

开发一个小程序要多少钱呢



想对作者说点什么



食肉大灰兔 8个月前

查看回复



现在新推出了一个权限框架，叫PyCasbin (<https://github.com/casbin/pycasbin>)。PyCasbin采用了元模型的设计思想，支持多种经典的访问控制方案，如ACL、RBAC、ABAC等。还支持对RESTful API的控制。现在已经支持Django、Flask等Web框架了。需要中文文档的话，可以在百度搜索：PyCasbin: <https://github.com/casbin/pycasbin>

举报



jackadam1981 1年前 貌似不完整吧？

