**类视图中使用装饰器**

简言之，python装饰器就是用于拓展【原来函数功能】的一种函数，这个【函数的特殊之处在于它的返回值也是一个函数】，使用python装饰器的 【好处】就是在不用更改原函数的代码前提下给函数增加新的功能。

**1.** 在视图函数中使用自定义装饰器，那么自己定义的装饰器必须放在`app.route`下面。否则这个装饰器就起不到任何作用。

例：

定义一个装饰器

*#需求：查看设置个人信息时，只有检测到用户已经登录了才能查看，若没有登录，则无法查看并给出提示信息*

**def** login\_requierd(func):

@wraps(func)

**def** wrapper(\*args,\*\*kwargs):

username = request.args.get(**"username"**)

**if** username **and** username ==**'momo'**:

**return** func(\*args,\*\*kwargs)

**else**:

**return '请先登录'**

**return** wrapper

使用自定义装饰器：

@app.route(**'/settings/'**)

@login\_requierd

**def** settings():

**return '这是设置界面'**

**2.** 在类视图中使用装饰器,需要重写类视图的一个类属性`decorators`，这个类属性是一个列表或者元组都可以，里面装的就是所有的装饰器。

定义一个装饰器

*#需求：查看设置个人信息时，只有检测到用户已经登录了才能查看，若没有登录，则无法查看并给出提示信息*

**def** login\_requierd(func):

@wraps(func)

**def** wrapper(\*args,\*\*kwargs):

username = request.args.get(**"username"**)

**if** username **and** username ==**'momo'**:

**return** func(\*args,\*\*kwargs)

**else**:

**return '请先登录'**

**return** wrapper

使用自定义装饰器：

**class** ProfileView(views.View):

decorators = [login\_requierd]

**def** dispatch\_request(self):

**return '这是个人中心界面'**

app.add\_url\_rule(**'/profile/'**,view\_func=ProfileView.as\_view(**'profile'**))