**类视图**

之前我们接触的视图都是函数，所以一般简称视图函数。其实视图也可以基于类来实现，类视图的好处是支持继承，但是类视图不能跟函数视图一样，写完类视图还需要通过app.add\_url\_rule(url\_rule,view\_func)来进行注册。以下先对标准类视图进行讲解：

**1.标准类视图使用步骤**

**1.** 标准类视图，必须继承自`flask.views.View`.

**2.** 必须实现`dipatch\_request`方法，以后请求过来后，都会执行这个方法。这个方法的返回值就相当于是之前的函数视图一样。也必须返回`Response`或者子类的对象，或者是字符串，或者是元组。

**3.** 必须通过`app.add\_url\_rule(rule,endpoint,view\_func)`来做url与视图的映射。`view\_func`这个参数，需要使用类视图下的`as\_view`类方法类转换：`ListView.as\_view('list')`。

**4.** 如果指定了`endpoint`，那么在使用`url\_for`反转的时候就必须使用`endpoint`指定的那个值。如果没有指定`endpoint`，那么就可以使用`as\_view(视图名字)`中指定的视图名字来作为反转。

|  |
| --- |
| **from** flask **import** Flask,views,url\_for  app = Flask(\_\_name\_\_)  @app.route(**'/'**)  **def** hello\_world():  **return 'Hello World!'**  #定义一个类视图  **class** ListView(views.View):  **def** dispatch\_request(self):  **return "这是List列表"**  #注册类视图  app.add\_url\_rule(**'/list/'**,endpoint=**'mlist'**,view\_func=ListView.as\_view(**'my\_list'**))  **with** app.test\_request\_context():  #若注册url时，没有指定endpoint，使用as\_view（）方法中的名称 来构建url  # print(url\_for('my\_list'))  # 若注册url时，有指定endpoint，就不能再使用as\_view（）方法中的名称 来构建url，而要使用endpoint的值来构建url  print(url\_for(**'mlist'**)) |

**2.类视图的好处**

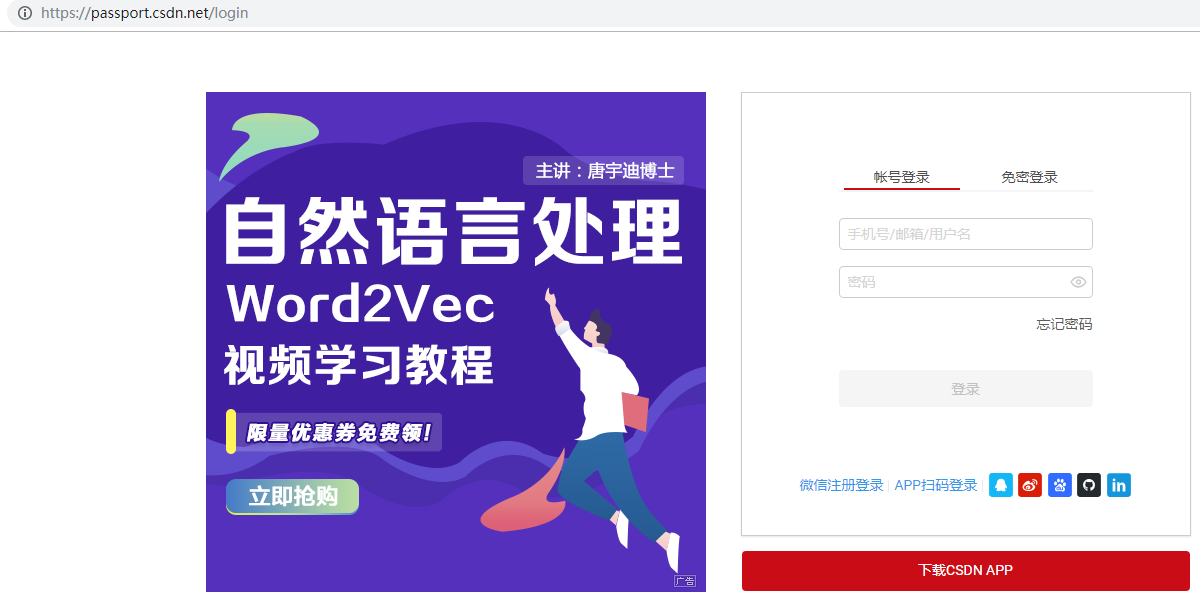
**1.**可以继承，把一些共性的东西抽取出来放到父视图中，子视图直接拿来用就可以了。

**2.**但是也不是说所有的视图都要使用类视图，这个要根据情况而定。视图函数用得最多

|  |
| --- |
| #类视图的好处：  #需求：以后有 好几个url，都需要返回json对象的格式  **class** ListView2(views.View):  **def** getData(self):  **raise** NotImplementedError  **def** dispatch\_request(self):  **return** jsonify(self.getData())  **class** JSONView(ListView2):  **def** getData(self):  **return** {**'uname'**:**'momo'**,**'age'**:**'22'**}  **class** JSONView2(ListView2):  **def** getData(self):  **return** {**'bname'**:**'水浒传'**,**'price'**:**'89'**}  app.add\_url\_rule(**'/json/'**,view\_func=JSONView.as\_view(**'my\_json'**))  app.add\_url\_rule(**'/json2/'**,view\_func=JSONView2.as\_view(**'my\_json2'**)) |

**3.标准类视图使用场景**







|  |
| --- |
| #需求2：有好几个url，跳转到到不同的页面时，会带一个相同的参数过去  # #登录功能  # class LoginView(views.View):  # def dispatch\_request(self):  # return render\_template('login.html',ads="茅台酒 998")  #  # #注册功能  # class RegisterView(views.View):  # def dispatch\_request(self):  # return render\_template('register.html',ads="茅台酒 998")  #改进：突出类视图的好处  **class** ADSView(views.View):  **def** \_\_init\_\_(self):  super(ADSView, self).\_\_init\_\_()  self.context={  **'ads'**:**'华为 5G'**  }  #登录功能  **class** LoginView(ADSView):  **def** dispatch\_request(self):  self.context.update({**'pid'**:**'好牛的一本书'**})  **return** render\_template(**'login.html'**,\*\*self.context)  #注册功能  **class** RegisterView(ADSView):  **def** dispatch\_request(self):  **return** render\_template(**'register.html'**,\*\*self.context)  app.add\_url\_rule(**'/login/'**,view\_func=LoginView.as\_view(**'login'**))  app.add\_url\_rule(**'/register/'**,view\_func=RegisterView.as\_view(**'register'**)) |