**Flask\_RESTful结合蓝图使用和渲染模版：**

**1.Flask\_RESTful结合蓝图使用**

在蓝图中，如果使用Flask\_RESTful，

那么在创建Api对象的时候，使用蓝图对象，不再是使用`app`对象了

蓝图news.py文件

|  |
| --- |
| **from** flask **import** url\_for,render\_template,Blueprint,make\_response,Response  **from** flask\_restful **import** Api,Resource,reqparse,inputs,fields,marshal\_with  **import** json  news\_bp = Blueprint(**'news'**,\_\_name\_\_,url\_prefix=**'/news'**)  api = Api(news\_bp)  #1.flask-restful结合蓝图使用  #复杂的参数结构  #实体间关系有 1：1 1:n n:n(转为2个1:n)  #新闻系统后台 用户 新闻 新闻标签  **class** User():  **def** \_\_init\_\_(self,id,uname,age):  self.id = id  self.uname = uname  self.age = age  **def** \_\_repr\_\_(self):  **return "<User:{id},{uname},{age}>"**.format(id=self.id, uname=self.uname,age=self.age)  **class** News():  **def** \_\_init\_\_(self,id,title,content):  self.id=id  self.title=title  self.content=content  #关系映射  self.author=**None**  self.tags=[]  **def** \_\_repr\_\_(self):  **return "<News:{id},{title},{content},{author},{tags}>"**\  .format(id=self.id, title=self.title,content=self.content,author=self.author,tags=self.tags)  **class** NewsTag():  **def** \_\_init\_\_(self,id,name):  self.id=id  self.name=name  **def** \_\_repr\_\_(self):  **return '<NewsTage:{id},{name}>'**.format(id =self.id , name=self.name)  **def** createData():  user = User(110,**'莫莫'**,30)  tag1 = NewsTag(200,**"要闻"**)  tag2 = NewsTag(210,**"娱乐"**)  news =News(300,**'吴京征服了世界上海拔最高的派出所'**,**'4月23日中午11点，吴京发了一条微博，配文“世界上海拔最高的派出所”，并@了另外一名演员张译。微博中有两张图片，第一张是吴京和张译两人坐在地上的合照，背后几个大字“中国边防”。第二张则是两人与派出所民警们的合照。 '**)  news.author = user #绑定新闻作者  news.tags.append(tag1) #绑定新闻标签1  news.tags.append(tag2) #绑定新闻标签2  print(news)  **return** news  **class** NewsView2(Resource):  resource\_fields={  **'id'**:fields.Integer,  **'title'**: fields.String,  **'content'**: fields.String,  #如在一个字段下面又是一个字典，那么可以使用fields.Nested({...})  **'author'**: fields.Nested({  **'id'**: fields.Integer,  **'uname'**: fields.String,  **'age'**:fields.Integer  }),  #如要在一个字段中放置一个列表，那么可以使用fields.List(fields.Nested({...}))  **'tags'**: fields.List(fields.Nested({  **'id'**: fields.Integer,  **'name'**: fields.String  }))  }  @marshal\_with(resource\_fields)  **def** get(self):  news = createData()  **return** news  api.add\_resource(NewsView2,**'/news2/'**) |

app.py文件

|  |
| --- |
| **from** flask **import** Flask,url\_for,render\_template  **from** blueprints.news **import** news\_bp  app = Flask(\_\_name\_\_)  app.register\_blueprint(news\_bp)  @app.route(**'/'**)  **def** hello\_world():  **return** render\_template(**'index.html'**)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  app.run(debug=**True**) |

**2.Flask\_RESTful渲染模版**

如果在Flask\_RESTful的类视图中想要返回html片段代码，或者是整个html文件代码，即渲染模版的意思。

那么就应该使用`api.representation`这个装饰器来定义一个函数，

在这个函数中，应该对`html`代码进行一个封装，再返回。

|  |
| --- |
| **from** flask **import** url\_for,render\_template,Blueprint,make\_response,Response  **from** flask\_restful **import** Api,Resource,reqparse,inputs,fields,marshal\_with  **import** json  news\_bp = Blueprint(**'news'**,\_\_name\_\_,url\_prefix=**'/news'**)  api = Api(news\_bp)  #2. 使用flask-restful渲染模版  **class** ListView(Resource):  **def** get(self):  **return** render\_template(**'index.html'**)  api.add\_resource(ListView,**'/list/'**)  # 渲染模版经过修改后，能支持html和json  @api.representation(**'text/html'**)  **def** output\_html(data,code,headers):  **if** isinstance(data,str):  # 在representation装饰的函数中，必须返回一个Response对象  # resp = make\_response(data)  resp =Response(data)  **return** resp  **else**:  **return** Response(json.dumps(data),mimetype=**'application/json'**) |