**add\_url\_rule和app.route原理剖析**

**`add\_url\_rule(rule,endpoint=None,view\_func=None)`**

这个方法用来添加url与视图函数的映射。如果没有填写`endpoint`，那么默认会使用`view\_func`的名字作为`endpoint`。以后在使用`url\_for`的时候，就要看在映射的时候有没有传递`endpoint`参数，如果传递了，那么就应该使用`endpoint`指定的字符串，如果没有传递，那么就应该使用`view\_func`的名字。

例：

**def** my\_list():

**return "我是列表页"**

app.add\_url\_rule(**'/list/'**,endpoint=**'sxt'**,view\_func=my\_list)

**`app.route(rule,\*\*options)`装饰器：**

这个装饰器底层，其实也是使用`add\_url\_rule`来实现url与视图函数映射的。

|  |
| --- |
| from flask import Flask,url\_for  app = Flask(\_\_name\_\_)  @app.route('/',endpoint='hello')  def hello\_world():  #构建url ：/list/  #研究app.add\_url\_rule()方法，若方法中【没有加】上endpoint时，可通过原来的函数名构建url，即url\_for('原函数名')  # print(url\_for('my\_list'))  #研究app.add\_url\_rule()方法，若方法中【加上】endpoint时，不能再通过原来的函数名构建url，而需要endpoint的值才行  # 即url\_for('endpoint值')  print(url\_for('li'))  return 'Hello World!'  def my\_list():  return "这是列表页"  #通过app对象的add\_url\_rule方法 来完成url与视图函数的映射  app.add\_url\_rule('/list/',endpoint='li',view\_func = my\_list)  #讨论：add\_url\_rule()方法 与@app.route()装饰器的关系  #结论：@app.route()装饰器 底层就是借助于add\_url\_rule()方法来实现的  #请求上下文对象 项目一启动就会执行  with app.test\_request\_context():  # print(url\_for('hello\_world'))  print(url\_for('hello'))  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |