标准类视图：

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。

写法:

```
#########################################################
calss ListView(views.View):
    def dispatch_request(self):
        return "list view"
app.add_url_rule('/list/', endpoint='list', view_func=ListView.as_view('list'))
#########################################################
```

5. 类视图有以下好处：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。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。

```
# 类视图例一 只要在子类中实现get_data方法, 就可以返回json数据, 减少代码量
#########################################################
class JSONView(views.View):
    def get_data(self):
        raise NotImplementedError

    def dispatch_request(self):
        return jsonify(self.get_data())
#########################################################
# 类视图例二 父类中定义的self变量, 子类可以直接调用 **self.param, 并且可以通过self.param.update()更新自己的参数
#########################################################
class ParamView(views.View):
    def __init__(self):
        super(ParamView, self).__init__()
        self.param = {}
```

##