小白学Flask第四天| 把路由转换器玩的更牛逼

原创JAP君 Python进击者
2019-08-21原文

本文内容：

路由转换器的进阶使用

自定义转换器

在上篇文章中我们也简单说了一下如何自定义转换器，我把代码重新弄过来：

```
# -*- coding: utf-8 -*-
from flask import Flask
from werkzeug.routing import BaseConverter
app = Flask(__name__)
```

```
# 1.定义自己的转换器
class RegexConverte(BaseConverter):
    def __init__(self, url_map, regex):
        # 调用父类的初始化方法
        super(RegexConverte, self).__init__(url_map)
        #
将正则表达式的参数保存在对象的属性中，flask会去使用这个属性来进行路由的
正则匹配
        self.regex = regex

# 2. 将自定义的转换器添加到flask的应用中
app.url_map.converters["re"] = RegexConverte

@app.route("/send/<re(r'1[345678]\d{9}'):moblie>")
def send_sms(moblie):

    return "send_sms: %s" % moblie

if __name__ == '__main__':

    # 启动flask程序
    app.run(debug=True)
```

大家可能看着一个例子会有点懵，这里我再写一个比较具体的例子给大家，这次的例子还是提取电话号码 ：

```python
# -*- coding: utf-8 -*-
from flask import Flask
from werkzeug.routing import BaseConverter
app = Flask(__name__)

class MobileConverte(BaseConverter):
    def __init__(self,url_map):
        # 调用父类的初始化方法
        super(MobileConverte, self).__init__(url_map)
        self.regex = r'1[345678]\d{9}'

# 2. 将自定义的转换器添加到flask的应用中
app.url_map.converters["mobile"] = MobileConverte

@app.route("<mobile:moblie_num>")
def send_sms(moblie_num):
    return "send_sms: %s" % moblie_num

if __name__ == '__main__':

    # 启动flask程序
    app.run(debug=True)
```

大家会注意到在第10行，self.regex = r'1[345678]\d{9}'，这句话其实就是我们整个功能的核心，在上一个例子中，我们是需要自己去定义正则表达式的，而这个例子只能实现提取电话号码这一个功能。

to_python方法

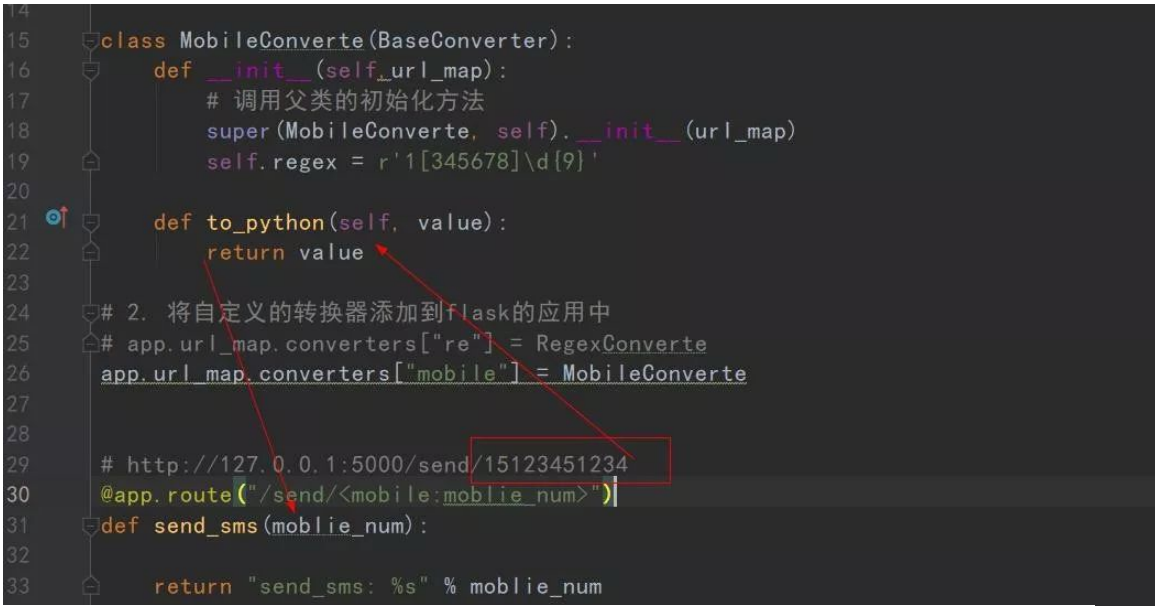
大家在自定义转换器时可能会发现我们需要创建一个类，然后我们初始化这个类。

按照这种写法一般不仅仅只是写了个__init__方法就可以实现转换器，而是这个类中还有其他的方法。

```
class BaseConverter(object):
    """Base class for all converters."""

    regex = "[^/]+"
    weight = 100

    def __init__(self, map):
        self.map = map

    def to_python(self, value):
        return value

    def to_url(self, value):
        if isinstance(value, (bytes, bytearray)):
            return _fast_url_quote(value)
        return _fast_url_quote(text_type(value).encode(self.map.charset))
```

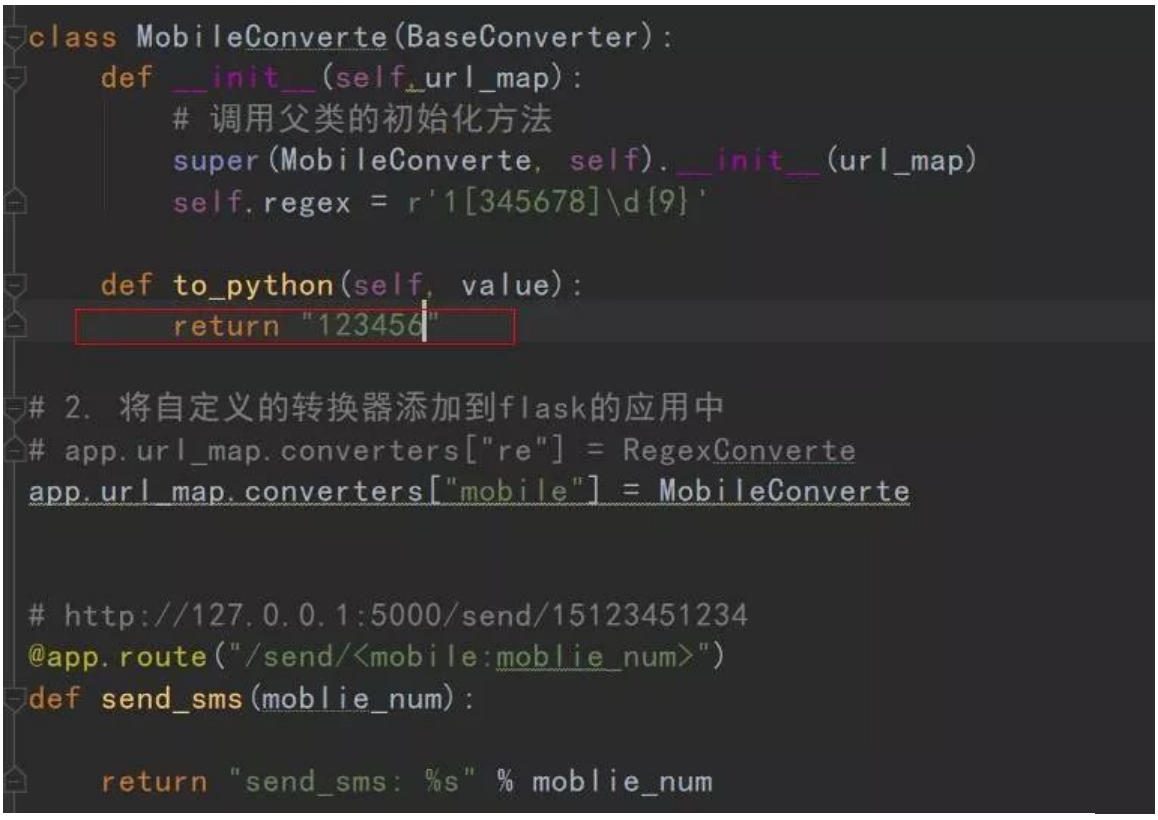
我们可以看到我们所继承的父类BaseConverter中，有to_python和to_url两个方法，我们首先来看看to_python方法。

其实**to_python这个方法才是转换器的核心**，当我们转换器提取到路径上面的参数后，不是直接返回给视图函数中的参数，而是要经过to_python方法才返回给视图函数，我给大家画了张图可能更容易理解：



```
class MobileConverte(BaseConverter):
    def __init__(self,url_map):
        # 调用父类的初始化方法
        super(MobileConverte, self).__init__(url_map)
        self.regex = r'1[345678]\d{9}'

    def to_python(self, value):
        return value

# 2. 将自定义的转换器添加到flask的应用中
# app.url_map.converters["re"] = RegexConverte
app.url_map.converters["mobile"] = MobileConverte


# http://127.0.0.1:5000/send/15123451234
@app.route("/send/<mobile:moblie_num>")
def send_sms(moblie_num):

    return "send_sms: %s" % moblie_num
```

怎么验证这个说法呢？我给大家举个例子：



```
class MobileConverte(BaseConverter):
    def __init__(self,url_map):
        # 调用父类的初始化方法
        super(MobileConverte, self).__init__(url_map)
        self.regex = r'1[345678]\d{9}'

    def to_python(self, value):
        return "123456"

# 2. 将自定义的转换器添加到flask的应用中
# app.url_map.converters["re"] = RegexConverte
app.url_map.converters["mobile"] = MobileConverte


# http://127.0.0.1:5000/send/15123451234
@app.route("/send/<mobile:moblie_num>")
def send_sms(moblie_num):

    return "send_sms: %s" % moblie_num
```

大家可以看到我把to_python方法的返回值给改成了123456，我们运行一下看看它是返回“123456”还是返回路径中所提取的参数

可以看到无论我在地址栏上输入什么，返回都是123456

那么有人会问这个方法有些什么用呢？

当然是有很大用处的，就拿我们提取手机号码参数举例，如果我们只希望用户提交的是133开头的手机号，我们就可以在to_python这个方法里面去进行操作。

to_url方法

除了to_python方法，这个方法有什么用呢？其实to_url方法和我们之前讲的url_for方法有着很大的联系，我们可以看下下面的代码：

```
# -*- coding: utf-8 -*-
from flask import Flask,redirect,url_for
from werkzeug.routing import BaseConverter
app = Flask(__name__)


class MobileConverte(BaseConverter):
    def __init__(self,url_map):
```

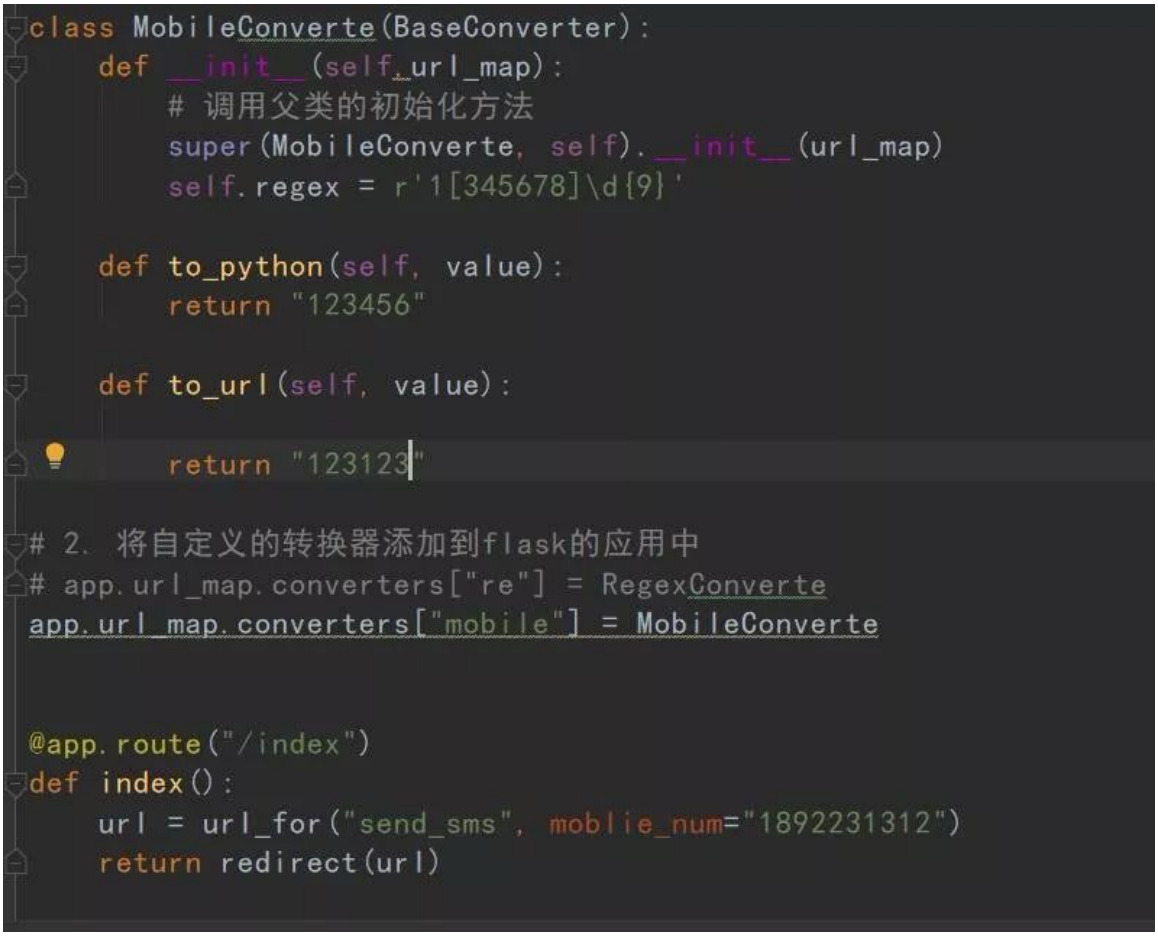
```
        # 调用父类的初始化方法
        super(MobileConverte, self).__init__(url_map)
        self.regex = r'1[345678]\d{9}'

    def to_python(self, value):
        return "123456"

# 2. 将自定义的转换器添加到flask的应用中
# app.url_map.converters["re"] = RegexConverte
app.url_map.converters["mobile"] = MobileConverte
# http://127.0.0.1:5000/send/15123451234
@app.route("/send/<mobile:moblie_num>")
def send_sms(moblie_num):

    return "send_sms: %s" % moblie_num

@app.route("/index")
def index():
    url = url_for("send_sms", moblie_num="1892231312")
    return redirect(url)

if __name__ == '__main__':

    # 启动flask程序
```

```
    app.run(debug=True)
```

我们主要看：

```
@app.route("/index")
def index():
    url = url_for("send_sms", moblie_num="1892231312")
    return redirect(url)
```

可以看到我们url_for里面有两个参数，第一则是指向我们send_sms视图函数的，后面那个则是send_sms函数中所提取电话号码的值。通过这样传值我们就可以通过url_for来调用一些有变化的参数的视图函数。



其实和to_python方法一样，每次进行url_for提交的参数都会先经过to_url，经过处理后才会返回回去。这里我就不过多演示。

连续打卡送书活动：

Flask系列文章大概会有15-20篇，如果读者在每次文章发布后进行打卡，**该系列结束后会赠送一本或者多本书籍。**

打卡方式：参与“1元混脸熟”的赞赏小活动，简单点说就是每次文章发布在文末赞赏1元，记住只能是1元。

“1元混脸熟”活动我会把经常赞赏我的朋友拉进铁粉群，群内会有一系列送书活动，当然也可以聊任何东西(赚钱、推广、经验分享）。

该系列文章结束，我会送一直坚持连续打卡的读者朋友一本或者多本书，当然书的价值绝对比你打卡的金额多。

精选留言

暂无...