登录

2019年04月19日 阅读 623

关注

Flask框架从入门到精通之转换器(四)

> 知识点：1、转换器 2、自定义转换器 3、转化器高级用法

一、概况

有很多请求我们URL地址后面会跟着参数，并且后端要提取到参数，比如: http://127.0.0.1:5000/center/1 这个URL地址后面有个1，可以代表我们要访问uid为1的个人信息。那么我们如何提取到参数呢?这个时候就会用到 Flask 的转换器。

二、转换器

在 Flask 中支持下面这些转换器：

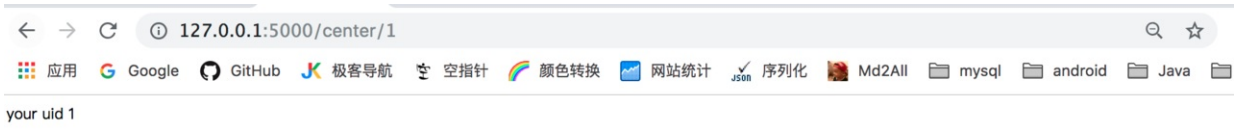
转换器	含义
default	接受字符串，默认转换器
stirng	接受字符串，跟默认一样
int	接受整数
float	同 int ，但是接受浮点数
uuid	唯一识别码
path	和默认的相似，但也接受斜线

复制代码

```
from flask import Flask
from flask import redirect, url_for

app = Flask(__name__)


@app.route('/center/<int:uid>')  # 代表个人中心页
def center(uid):  # 视图函数

    return 'your uid %s' % uid  # 返回内容


if __name__ == '__main__':
```

```
# 0.0.0.0代表任何能代表这台机器的地址都可以访问
app.run(host='0.0.0.0', port=5000)  # 运行程序
```

在浏览器访问：

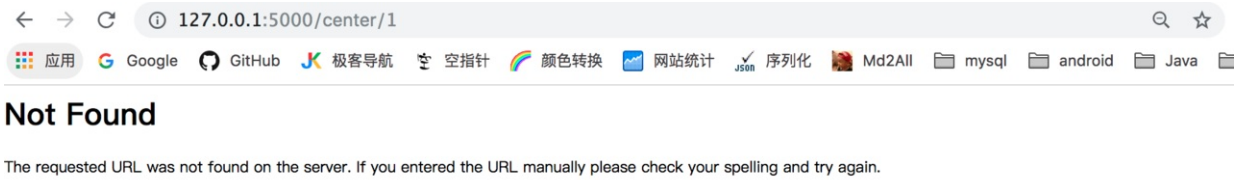


如果改成:

```
@app.route('/center/<float:uid>')  # 代表个人中心页
```

复制代码

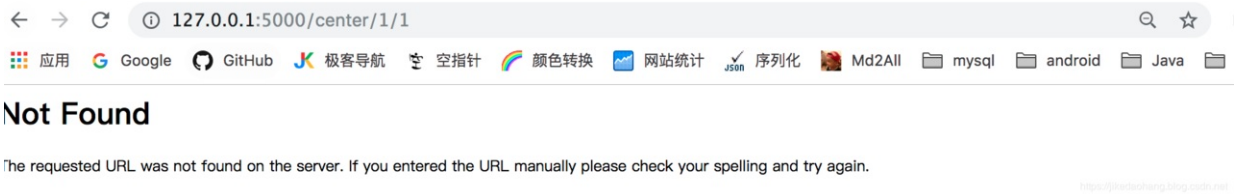
再去访问就会报错，因为 `float` 只支持浮点数。



如果不写转换器，那么它的默认转换器就是 `str` 类型。但是不包含 `/` ,因为它在这里面有特殊的意义。

```
@app.route('/center/<uid>')  # 代表个人中心页
```

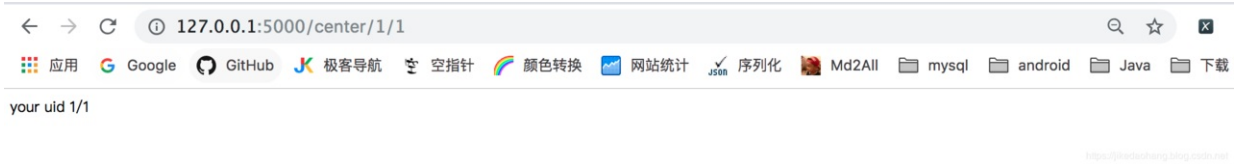
复制代码



如果要匹配 `/` ，请用 `path` 转换器：

```
@app.route('/center/<path:uid>')  # 代表个人中心页
```

复制代码



三、自定义转换器

有时候系统提供的转换器并不能满足我们，比如我们需要的UID是至少5最多10位。如果不是这个范围呢，我们就可以认为不满足我们的匹配规则。而现在我们不管输入多少位，后台都会匹配上，然后就去执行我们的视图。如果要排除这种现象，我们就需要自定义转换器。

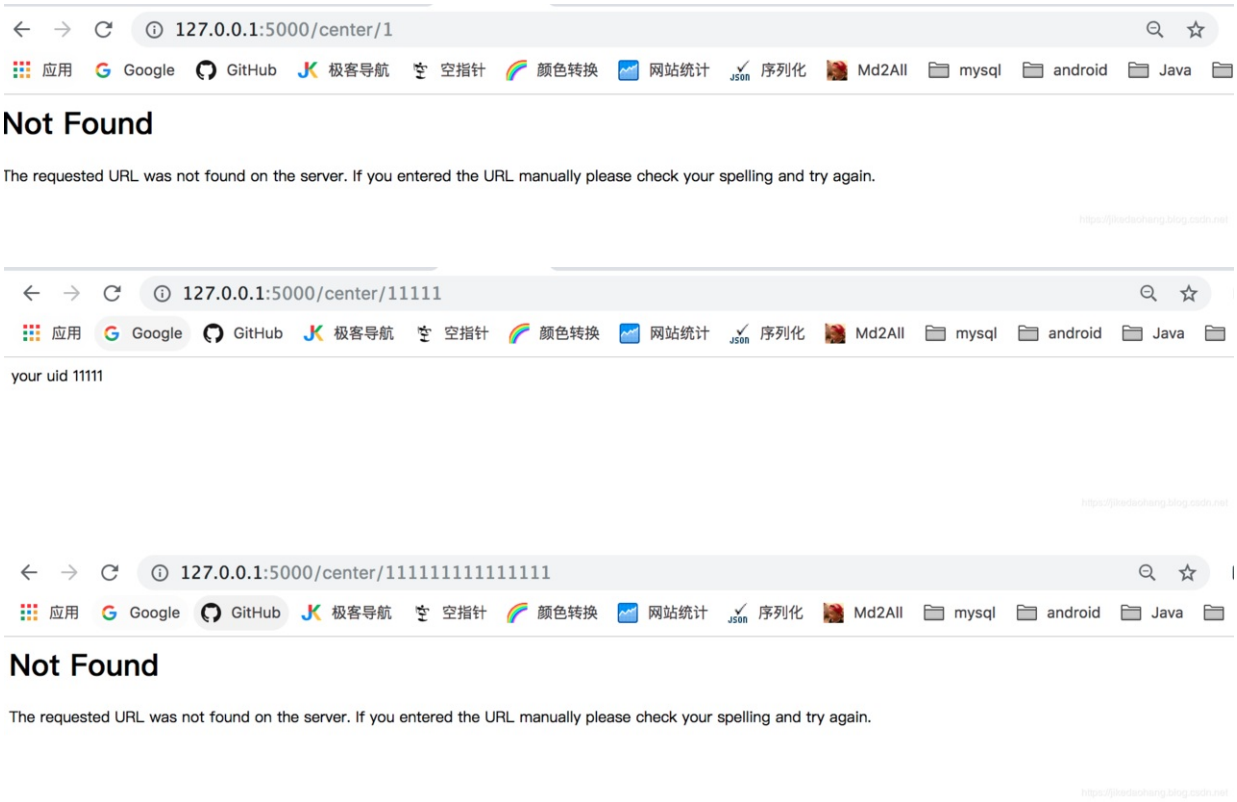
需要用到 `Flask` 这个转换器基类 `BaseConverter`

需要用到 Flask 这个转换器基类 BaseConverter

复制代码

```
from flask import Flask
from flask import redirect, url_for
from werkzeug.routing import BaseConverter

app = Flask(__name__)


# 自定义转换器
class MyConverter(BaseConverter):

    def __init__(self, map):
        super().__init__(map)  # 调用父类
        self.regex = r'\d{5,10}'  # 转换器的正则规则


app.url_map.converters['re'] = MyConverter  # 注册到converters

#会根据注册的键re找到MyConverter这个类创建对象
@app.route('/center/<re:uid>')  # 代表个人中心页
def center(uid):  # 视图函数

    return 'your uid %s' % uid  # 返回内容


if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
    app.run(host='0.0.0.0', port=5000)  # 运行程序
```

这个在浏览器试下：



我们发现已经完全解决了我们刚才的问题。

四、优化转换器

我们发现这个转化器的规则是写死的，只能匹配UID，如果其他地方用到要匹配手机号怎么办，所以最好匹配规则是动态传递的。那么改成如下：

复制代码

```
from flask import Flask
from flask import redirect, url_for
from werkzeug.routing import BaseConverter

app = Flask(__name__)


# 自定义转换器
class MyConverter(BaseConverter):

    def __init__(self, map, re):
        super().__init__(map)  # 调用父类
        self.regex = re  # 转换器的正则规则


app.url_map.converters['re'] = MyConverter  # 注册到converters


#会根据注册的键re找到MyConverter这个类创建对象,并把规则当做初始化参数传递进去
@app.route('/center/<re(r"\d{5,10}"):uid>')  # 代表个人中心页
def center(uid):  # 视图函数

    return 'your uid %s' % uid  # 返回内容


@app.route('/login/<re(r"1[3456789]\d{9}"):phone>')  # 代表登录页
def login(phone):  # 视图函数

    return 'your phone %s' % phone  # 返回内容


if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
    app.run(host='0.0.0.0', port=5000)  # 运行程序
```

用到的时候，在把具体的正则条件传递进去，就好了。

五、转换器高级用法

通过上面的一些场景我们用到了转换器，可是既然转换器，我们并没有看出转换这个两个字在哪有体现，感觉不如叫适配器。但是进入 BaseConverter 源码里面看一下，发现有两个方法，这两个方法是到底有什么作用呢，不妨我们来重写一下。

复制代码

```
class BaseConverter(object):

    """Base class for all converters."""
    regex = '[^/]+'
    weight = 100

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

    def to_python(self, value):
```

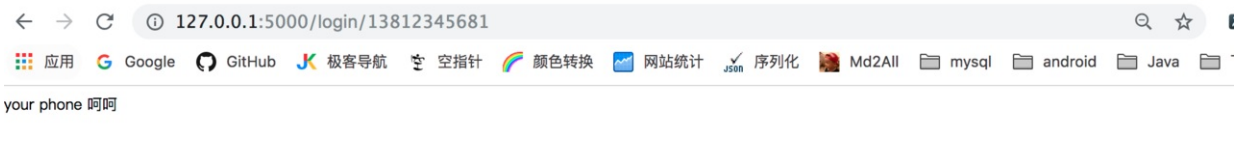
```
        return value

    def to_url(self, value):
        return url_quote(value, charset=self.map.charset)
```

- **重写to_python 我们在这个方法里面随便返回点内容**

复制代码

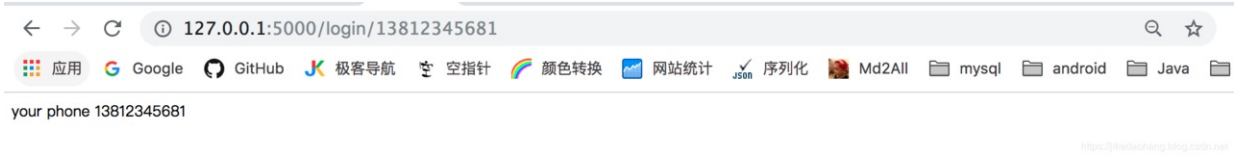
```
from flask import Flask
from flask import redirect, url_for
from werkzeug.routing import BaseConverter

app = Flask(__name__)


# 自定义转换器
class MyConverter(BaseConverter):

    def __init__(self, map, re):
        super().__init__(map)  # 调用父类
        self.regex = re  # 转换器的正则规则

    def to_python(self, value):
        return '呵呵'  # 随便给一个返回值


app.url_map.converters['re'] = MyConverter  # 注册到converters


# 会根据注册的键re找到MyConverter这个类创建对象,并把规则当做初始化参数传递进去
@app.route('/center/<re(r"\d{5,10}"):uid>')  # 代表个人中心页
def center(uid):  # 视图函数

    return 'your uid %s' % uid  # 返回内容


# 会根据注册的键re找到MyConverter这个类创建对象,并把规则当做初始化参数传递进去
@app.route('/login/<re(r"1[3456789]\d{9}"):phone>')  # 代表登录页
def login(phone):  # 视图函数

    return 'your phone %s' % phone  # 返回内容


if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
    app.run(host='0.0.0.0', port=5000)  # 运行程序
```

当我们请求的时候发现：



如果还继续返回value呢?

```
    def to_python(self, value):
        return value  # 返回value
```



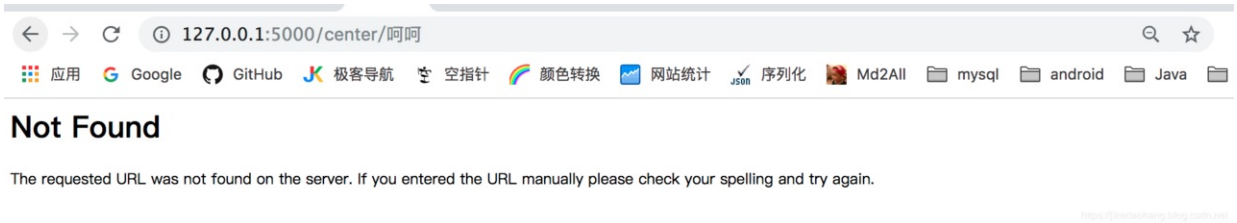
看到这我们应该明白了，父类里面默认返回的值就是正则匹配的值，如果重写了，传入进来的value还是正则匹配的值，但是return的值我们可以自己改变了。比如你传的是13888888888，那我就返回13999999999在这才有转换的体现。到底返回什么样的值，还是看具体场景。

- 重写to_url 不知道还记得我们的重定向吗?我们可以通过重定向到任何视图中。如果我们现在希望登录函数重定向到个人中心视图怎么办?直接写函数名字，好像不对了，因为现在路由需要一个UID参数。我们需要像下面这么写，并且我们在to_url方法呢随便返回点内容：

```
from flask import Flask
from flask import redirect, url_for
from werkzeug.routing import BaseConverter

app = Flask(__name__)


# 自定义转换器
class MyConverter(BaseConverter):

    def __init__(self, map, re):
        super().__init__(map)  # 调用父类
        self.regex = re  # 转换器的正则规则

    def to_python(self, value):
        return value  # 返回value

    def to_url(self, value):
        return '呵呵'  # 随便返回点内容


app.url_map.converters['re'] = MyConverter  # 注册到converters


# 会根据注册的键re找到MyConverter这个类创建对象,并把规则当做初始化参数传递进去
@app.route('/center/<re(r"\d{5,10}"):uid>')  # 代表个人中心页
def center(uid):  # 视图函数
    return 'your uid %s' % uid  # 返回内容


# 会根据注册的键re找到MyConverter这个类创建对象,并把规则当做初始化参数传递进去
@app.route('/login/<re(r"1[3456789]\d{9}"):phone>')  # 代表登录页
def login(phone):  # 视图函数
    return redirect(url_for('center', uid='11111'))  # 重定向到个人中心页, uid是路由上的参数


if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
    app.run(host='0.0.0.0', port=5000)  # 运行程序
```

我们在浏览器试一下：

我们在浏览器试一下：http://127.0.0.1:5000/login/13812345678



注意浏览器center后面的UID变成了呵呵两个字。如果还继续返回value呢？

```
def to_url(self, value):
    return value  # 返回value
```

我们在浏览器试一下：http://127.0.0.1:5000/login/13812345678

如果我们我们这改下代码：

```
def to_python(self, value):
    return '12312'

def to_url(self, value):
    return value  # 返回value
```

写到这，我们可以简单总结一下这两个函数什么时候执行

- **to_python** 只要使了转换器，一直会被调用
- **to_url** 只有在重定向了，并且路由上有参数的时候才会被调用

当我们重定向了，并传递参数的时候，参数会先到**to_url**，返回的值会去匹配路由上的正则。如果匹配上了，就把匹配到参数传到**to_python** ，这个方法返回的值才会到视图函数中。如果匹配不上，就会报404错误。流程如下：

```python
@app.route('/center/<re(r"\d{5,10}"):uid>')  # 代表个人中心页
def center(uid):  # 视图函数
    return 'your uid %s' % uid  # 返回内容


# 会根据注册的键re找到MyConverter这个类创建对象,并把规则当做初始化参数传递进去
@app.route('/login/<re(r"1[3456789]\d{9}"):phone>')  # 代表登录页
def login(phone):  # 视图函数
    return redirect(url_for('center', uid='111111'))  # 重定向到个人中心页，uid是路由上的参数
```

login()

欢迎关注我的公众号：

文章分类

后端

文章标签

Flask

相关推荐