1. url_for url反转

第一种用法, 由视图函数得到所对应的url:

router装饰器将url对应到视图函数, url_for是将视图函数对应到url

可以通过url_for('视图函数名', 变量名=数值) 这种写法构建含参的url, 当url_for()中传入的参数是url未定义的, 则会通过查询字符串参数拼接到返回的url后

为什么使用url_for而不是直接写url: url有可能经常修改, 而视图函数是定义好不会轻易修改的, 当页面较多时, 修改可能会很麻烦

第二种用法, 定位静态文件

```
<script src="{{ url_for('static', filename='javascript/bootstrap.min.js') }}"></script>
<img src="{{ url_for('static', filename='image/reid.png') }}" class="logo" alt="">
```

2. 自定义url转换器

url中参数类型由转换器实现, 可以自已定义所需的参数类型, 比如定义手机号码数据类型:

```
# 引入基础转换器 自定义手机号码匹配 加入到map中
from werkzeug.routing import BaseConverter
class PhoneNumberConverter(BaseConverter):
    reg = r'1[345789]\d{9}'
app.url_map.converters['tel'] = PhoneNumberConverter
```

转换器中的to_python方法, 可以在转换器中处理参数, 将处理好的结果返回给视图函数

转换器中的to_url方法, 可以在转换器中处理url_for传入的参数, 将处理好的结果以url形式返回给url_for

---

```
class ListConverter(BaseConverter):
    def to_python(self, value):
        return value.split('+')

    def to_url(self, value):
        return "+".join(value)
        # return "hello"

app.url_map.converters['list'] = ListConverter
```

```
@app.route('/')
def test1():
    print(url_for('posts',boards=['a','b']))
    return 'Hello World!'

@app.route('/posts/<list:boards>/')
def test2(boards):
    print(boards)
    return "您提交的板块是：%s" % boards
```

如上代码段 , 在test1视图函数中, url_for打印出的结果是 /post/a+b 经过to_url处理过的结果

在test2视图函数中, 如果输入在地址栏输入url为/post/a+b/ 视图函数中打印出的是['a', 'b'], 是经to_python处理过的结果