小白学Django第十天| 模板的知识全部给你总结好了！

原创JAP君 Python进击者
2020-03-11原文

前言

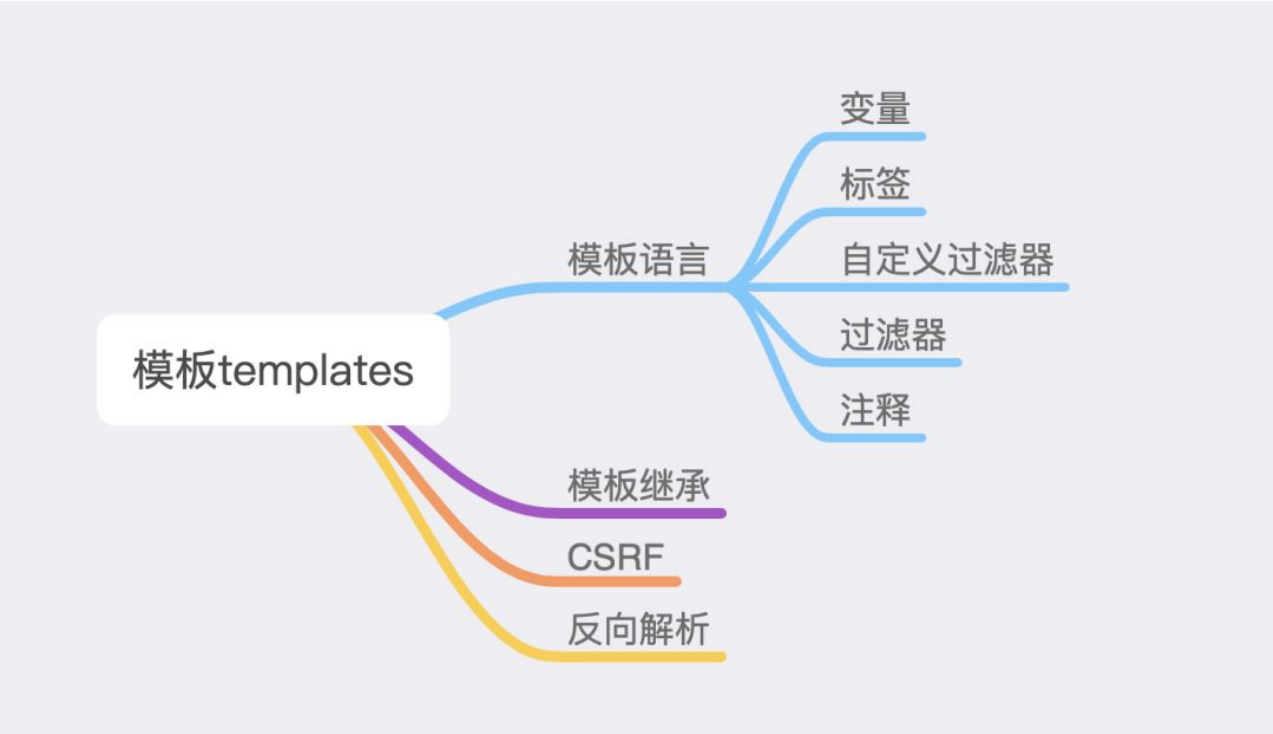
既然被灰度到了，那就来体验一把，也让公众号的读者体验一把微信的付费阅读。虽然这篇文章是付费文章，但是本文内容大家都是可见的，那么付费有啥用？就当支持一下坚持写技术原创系列文的kuls吧！

小白学Django系列：

- 小白学Django第一天| MVC、MVT以及Django的那些事
- 小白学Django第二天| Django原来是这么玩的！
- 小白学Django第三天| 一文带你快速理解模型Model

- 小白学Django第四天| Django后台管理及配置MySQL数据库
- 小白学Django第五天| 视图View的初步使用
- 小白学Django第六天| 一文快速搞懂模板的使用
- Django实战小型图书人物信息网页(MVT的综合运用)
- 小白学Django第七天| 模型类Model进阶学习
- 小白学Django第八天| 登录案例实战
- Django| 给你博客装个Markdown编辑器
- 小白学Django第九天| Cookie和session的那些骚操作
- 手把手教你在centos上配置Django项目（超详细步骤）
- 持续更新中...

前言



本文内容大纲

本文将详细讲解Django里的模板知识。讲解目录如上。

模板语言

1.变量

作为一个Web框架，Django需要一种**动态**生成HTML的便捷方法。最常用的方法依赖于模板。模板包含所需HTML输出的静态部分以及描述动态内容将被插入的一些特殊语法。简单的来说，就是在html文件中插入一些视图函数传输过来的数据。

调用变量的语法：

```
{{变量}}
```

★ 变量名必须由字母、数字、下划线（不能以下划线开头”）和点组成。

我之前在**小白学Django第六天| 一文快速搞懂模板的使用** 一文中讲解过如何简单的使用模板，所以这里不再阐述。

我们来了解一下模板是如何去读取这些数据，当模板引擎碰到了我们的模板变量到底是个怎样的过程：

这里我以{{book.title}}来举个简单的例子

1. 先把它当做字典book['title']
2. 先属性后方法。将book当作对象，查找属性title，如果没有再查找方法title()
3. 如果是格式为book.0则解析为列表book[0]

如果变量不存在则插入空字符串''。”

2.标签

语法：

```
{%代码段%}
```

关于模板标签其实有非常的多，大家可以查询官网：https://docs.djangoproject.com/zh-hans/3.0/ref/templates/builtins/

我这里简单给大家举最常用的for和if

for:

```
{%for item in 列表%}
循环逻辑
{{forloop.counter}}表示当前是第几次循环，从1开始
{%empty%}
列表为空或不存在时执行此逻辑
{%endfor%}
```

if:

```
{%if ...%}
逻辑1
```

```
{%elif ...%}
逻辑2
{%else%}
逻辑3
{%endif%}
```

这里一定要注意，运算符左右两侧不能紧挨变量或常量，必须有空格。”

3.过滤器

过滤器从字面意思就能看出是什么意思，也就是对我们目前拿到的数据进行进一步的过滤。

语法：

```
变量|过滤器:参数
```

对于过滤器这两点你得清楚：

1. 使用管道符号|来应用过滤器，用于进行计算、转换操作，可以使用在变量、标签中。
2. 如果过滤器需要参数，则使用冒号:传递参数。

其实过滤器也有非常的多，例如设置默认值：

```
data|default:'默认值'
```

就是当返回的变量为空时，默认显示的值。

还有日期过滤器：

```
value|date:"Y年m月j日  H时i分s秒"
```

过滤器并不需要你全部记住，当你需要使用的时候可以去https://docs.djangoproject.com/zh-hans/3.0/ref/templates/builtins/进行查询。

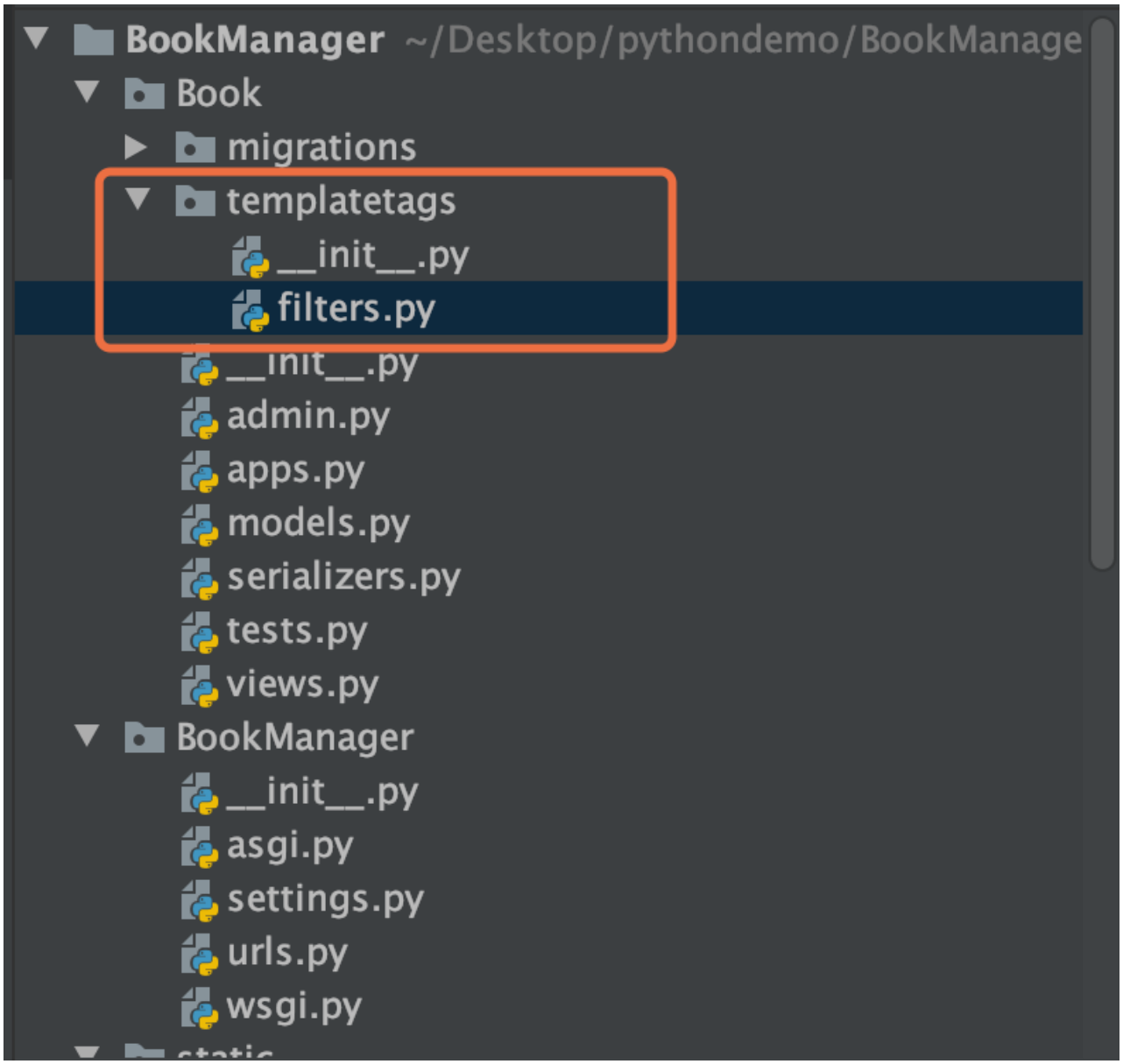
4.自定义过滤器

很多时候，官方提供的过滤器往往不能满足我们的需求，这个时候就需要我们自己来造一个过滤器。如何自己创造一个过滤器，看下面：

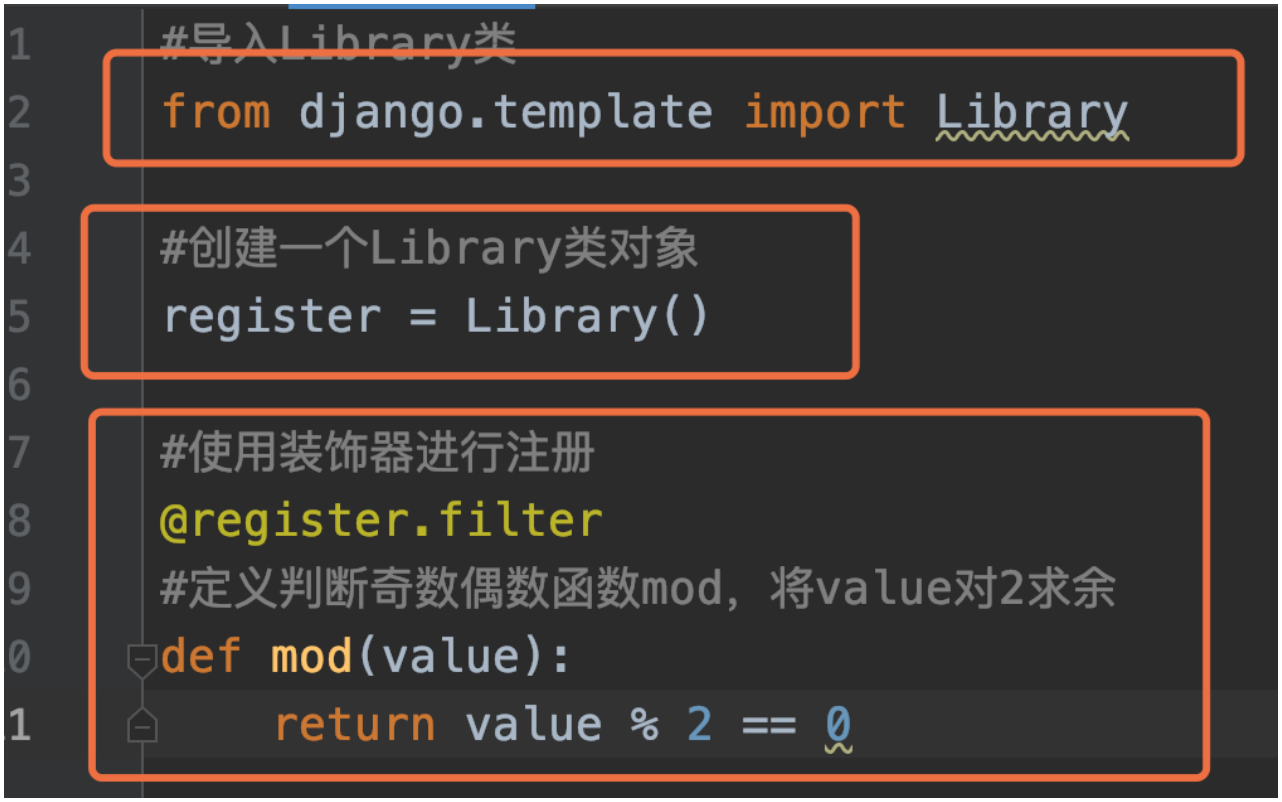
1）在应用中创建templatetags目录，当前示例为"你的应用/templatetags"，创建_init_文件，内容为空。

这里需要注意创建的目录名称一定要为templatetags，不可以是其他名称。

2）我们在templatetags目录下新建一个py文件，例如我这里就是fliter.py。这个名称大家随意。

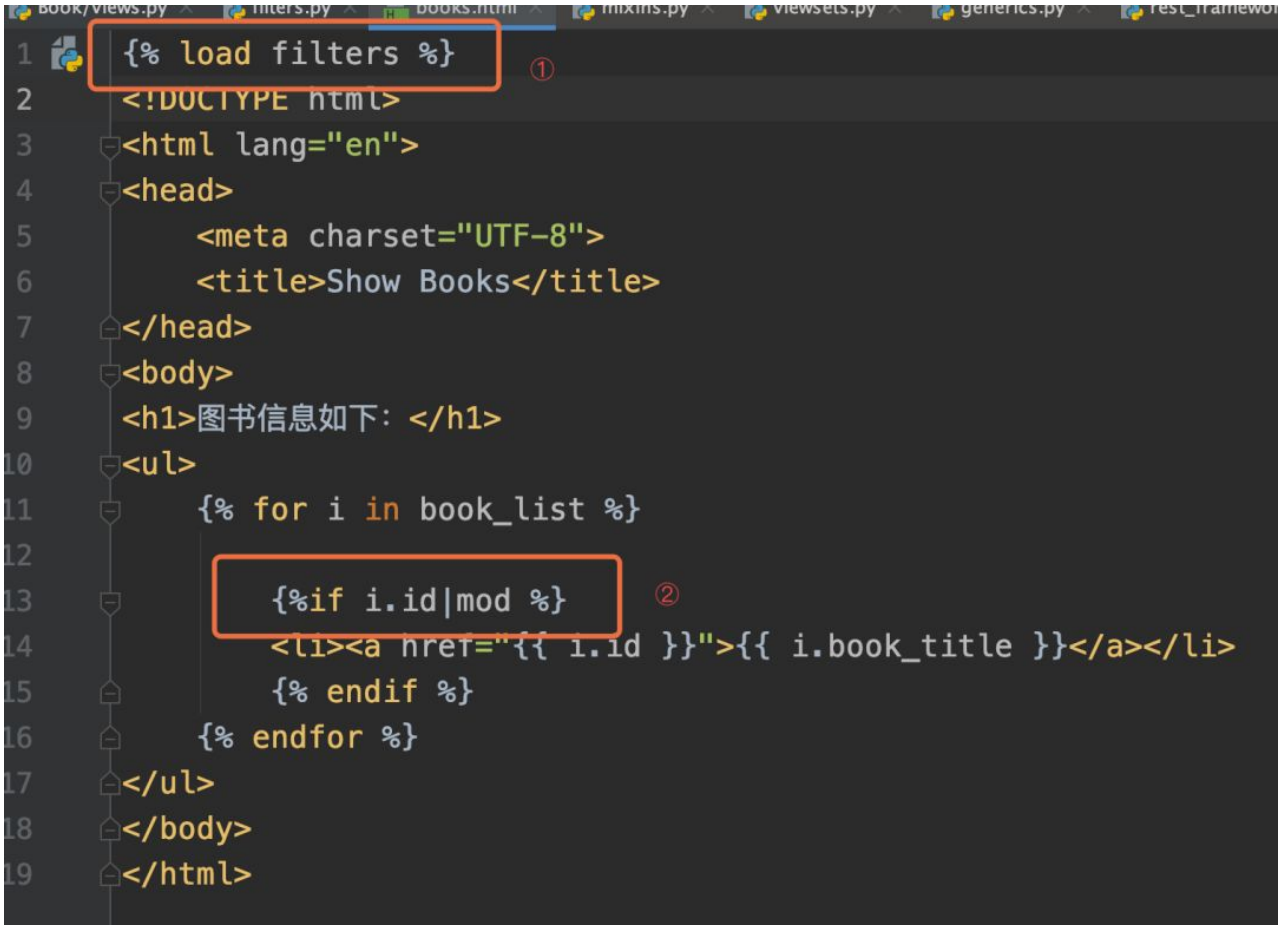

BookManager ~/Desktop/pythondemo/BookManage
Book
migrations
templatetags
__init__.py
filters.py
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
views.py
BookManager
__init__.py
asgi.py
settings.py
urls.py
wsgi.py

3）编写过滤器



4）在html中调用

首先需要导入相关文件，其次在跟普通过滤器一样调用，详细请看图。



当然，我们自定义的过滤器也是可以接受函数的。

我们再fliter.py文件中增加一个函数：

```
#使用装饰器进行注册
@register.filter
#定义求余函数mod_num，将value对num求余
def mod_num(value,num):
    return value%num
```

然后通过：

```
{%if i.id|mod:3 %}
```

进行传参调用

5.注释

在模板中的注释，大家应该需要了解一下，对于我们进行代码解释有着很大的帮助：

1）单行注释语法如下：

```
{#...#}
```

注释可以包含任何模版代码，有效的或者无效的都可以。

```
{# { % if foo % }bar{ % else % } #}
```

2）多行注释使用comment标签，语法如下：

```
{%comment%}
...
{%endcomment%}
```

模板继承

关于模板继承，其实和类的继承是差不多的，都是为了减轻我们的工作量。

大家想想，我们模板哪些地方需要继承呢？最多的就是导航栏，底部信息栏，侧边信息栏。

既然类似于类的继承，在模板继承中，也分为父模板和子模板。

1. 父模板

父模板主要是写模板中重复使用的地方。

标签block：用于在父模板中预留区域，留给子模板填充差异性的内容，名字不能相同。为了更好的可读性，建议给endblock标签写上名字，这个名字与对应的block名字相同。父模板中也可以使用上下文中传递过来的数据。

```
{%block 名称%}
预留区域，可以编写默认内容，也可以没有默认内容
{%endblock  名称%}
```

2.子模板

需要使用标签extend进行继承，并写在子模板第一行：

```
{% extends "父模板路径"%}
```

子模版不用填充父模版中的所有预留区域，如果子模版没有填充，则使用父模版定义的默认值。

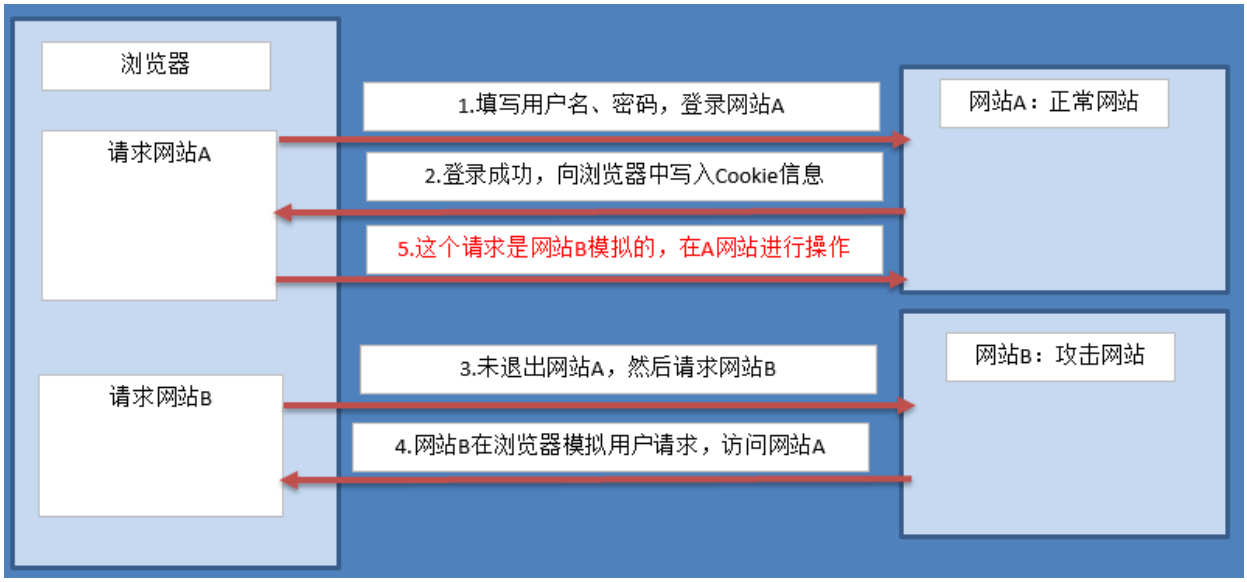
填充父模板中指定名称的预留区域。

```
{%block 名称%}
实际填充内容
{{block.super}}用于获取父模板中block的内容
{%endblock 名称%}
```

CSRF

CSRF（Cross Site Request Forgery），译为**跨站请求伪造**。CSRF指攻击者盗用了你的身份，以你的名义发送恶意请求。CSRF能够做的事情包括：以你名义发送邮件，发消息，盗取你的账号，甚至于购买商品，虚拟货币转账.....造成的问题包括：个人隐私泄露以及财产安全。

csrf攻击原理图：



通过上面的图，大家可能就会知道**csrf攻击的罪魁祸首就是cookie**，另一个网站利用了你当前网站的cookie来进行一些恶意操作。因为另一个网站拿到你的cookie之后，就可以对你当前网站为所欲为。

在前面的文章中说到过post适用于安全性需求高的数据，所以我们主要讲讲csrf在Django的post方式时的防范姿势。

1. Django提供了csrf中间件用于防止CSRF攻击，只需要在settings.py中启用csrf中间件即可。

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

但是你打开这个中间件后，你会发现访问不了自己的网站了，会出现403的警告。

2. 所以接下来修改html内容，在form表单中使用标签csrf_token。

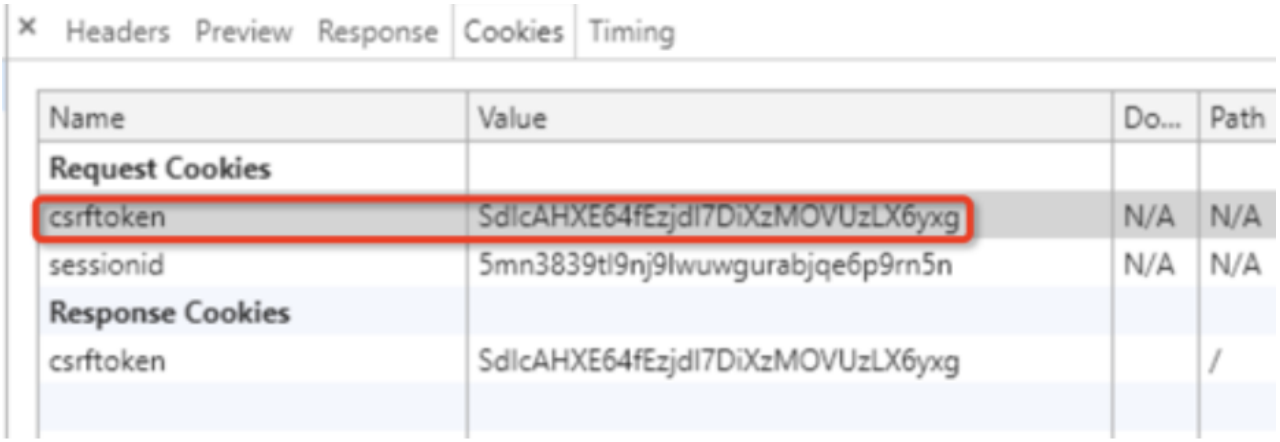
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>发帖页</title>
</head>
<body>
<form method="post" action="/post_action/">
    {% csrf_token %}
    标题:<input type="text" name="title"/><br/>
    内容:<textarea name="content"></textarea>
    <input type="submit" value="发帖"/>
</form>
</body>
</html>
```

这个时候就可以防住csrf的攻击。

以上内容只是简单讲解攻击原理以及防范方法，接下来给大家讲解防范CSRF的原理！

我们加入csrf_token的标签后，会发现form表单中出现了一个name为csrfmiddlewaretoken的值，下图：

然后此时，我们再去看下cookie



我们会发现这两个值一模一样，所以**它的原理就是来比对你提交时候表单里csrfmiddlewaretoken的值是不是一致的，如果是一致的，那么就放行；如果不一致，就返回403警告。**

反向解析

首先来举个例子说下为什么需要反向解析。

创建两个视图函数：

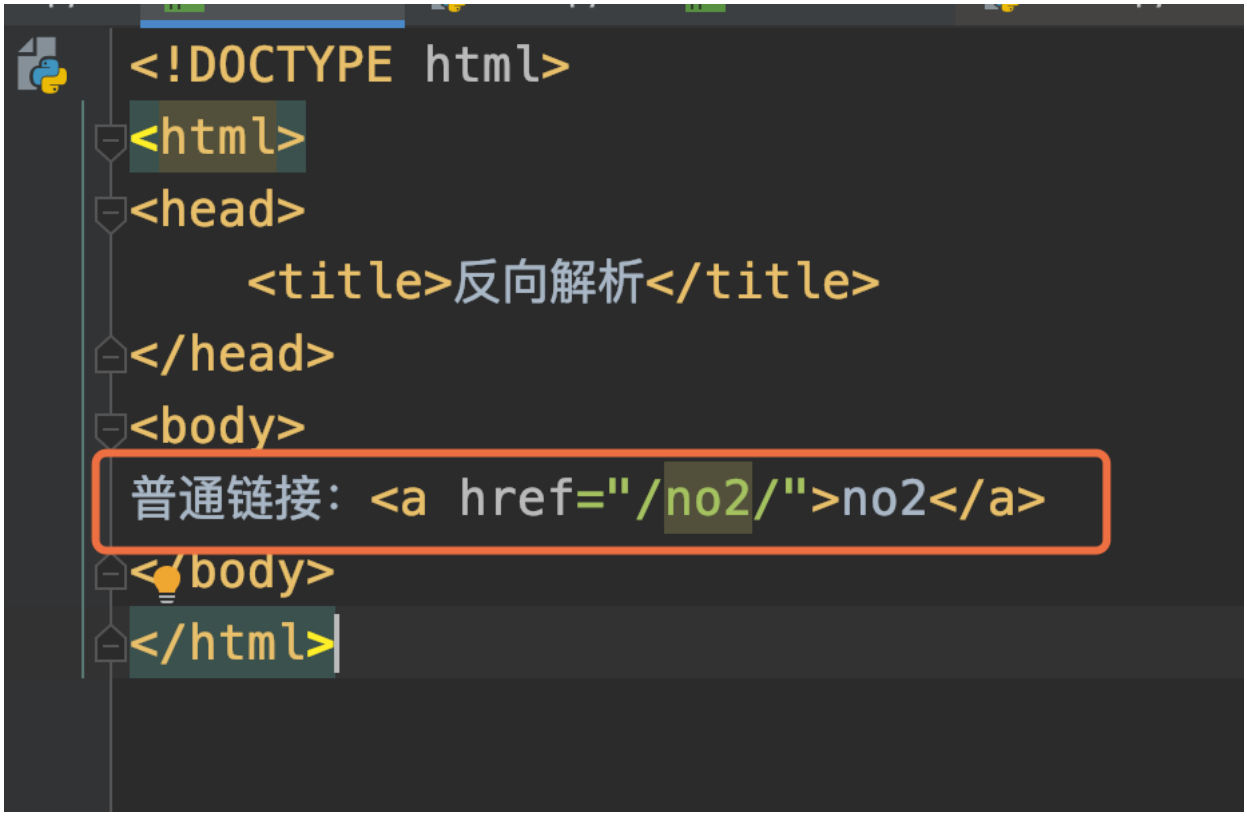
```
def no1(request):
    return render(request, 'Book/no1.html')

def no2(request):
    return HttpResponse('这里是no2页面')
```

配置URL

```
path('no1/', views.no1),
path('no2/', views.no2),
```

并且创建好no1.html



我们运行服务器

点击no2超链接，跳转到no2的页面

整个过程没有任何问题，但是如果我们此时修改了no2的url配置，如下图：

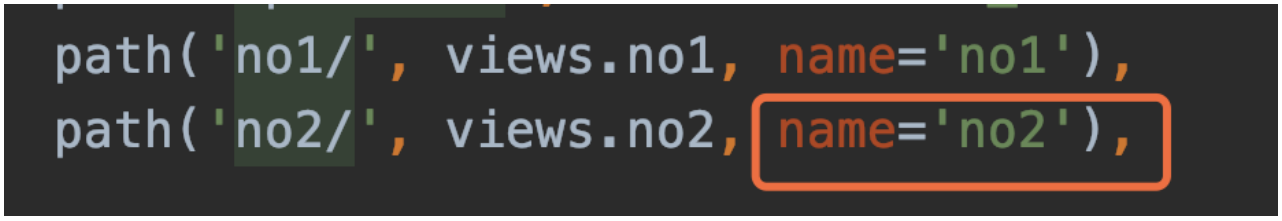
我们把原来的no2改为了no_url2。

那我们此时去点击no2的超链接肯定就不行了，因为no2的页面的url已经改了，如果需要实现点击跳转，那么我们需要去修改no1.html中超链接的路径。

但是，如果我们一个url配置在许多页面都有调用，那修改起来就有点头疼。所以这里就引来了反向解析的概念。

★ 反向解析应用在两个地方：模板中的超链接，视图中的重定向。”

如何实现反向解析，很简单，先将url配置增加name参数，如下图：



然后重新编写no1.html：

```
<html>
<head>
    <title>反向解析</title>
</head>
<body>
普通链接: <a href="/no2/">no2</a>
<hr>
反向解析: <a href="{%url 'no2'%}">no2</a>
</body>
</html>
```

注意红框内的写法。

就这样简单的两步就能够实现反向解析。

上面也说过反向解析还可以用于视图函数的重定向。

```
from django.shortcuts import redirect
from django.core.urlresolvers import reverse

return redirect(reverse('no2'))
```

总结：在定义url时，需要为url定义name属性，使用时，在模板中使用url标签，在视图中使用reverse函数，根据正则表达式动态生成地址，减轻后期维护成本。

带参数的反向解析

也许有些url是会带有参数的，那么我们如何解决呢？例如有下列的视图函数：

```
def jiafa(request, a, b):
    return HttpResponse(a+b)
```

我们可以通过下面形式来反向解析

```
反向解析：<a href="{%url 'jiafa' 2 3%}">jiafa</a>
```

例如上面是需要传递两个int整型参数的url，那我们就将参数写在后面即可，记得参数之间有空格。

```
return redirect(reverse('jiafa', args=(2,3)))
```

在视图函数，增添一个args的参数，将需要传递的值通过元组的形式传送。

还有一种需要传递关键词参数的url：

```
反向解析：<a href="{%url 'jiafa' a=100 b=200}">jiafa</a>
```

```
return redirect(reverse('jiafa', kwargs={'a':100,'b':18}))
```

其实下面还有文字

精选留言

暂无...