因为因特网HTTP协议的特性，每一次来自于用户浏览器的请求（request）都是无状态的、独立的。通俗地说，就是无法保存用户状态，后台服务器根本就不知道当前请求和以前及以后请求是否来自同一用户。对于静态网站，这可能不是个问题，而对于动态网站，尤其是京东、天猫、银行等购物或金融网站，无法识别用户并保持用户状态是致命的，根本就无法提供服务。你可以尝试将浏览器的cookie功能关闭，你会发现将无法在京东登录和购物。

为了实现连接状态的保持功能，网站会通过用户的浏览器在用户机器内被限定的硬盘位置中写入一些数据，也就是所谓的Cookie。通过Cookie可以保存一些诸如用户名、浏览记录、表单记录、登录和注销等各种数据。但是这种方式非常不安全，因为Cookie保存在用户的机器上，如果Cookie被伪造、篡改或删除，就会造成极大的安全威胁，因此，现代网站设计通常将Cookie用来保存一些不重要的内容，实际的用户数据和状态还是以Session会话的方式保存在服务器端。

Session依赖Cookie！但与Cookie不同的地方在于Session将所有的数据都放在服务器端，用户浏览器的Cookie中只会保存一个非明文的识别信息，比如哈希值。

Django提供了一个通用的Session框架，并且可以使用多种session数据的保存方式：

* 保存在数据库内
* 保存到缓存
* 保存到文件内
* 保存到cookie内

通常情况，没有特别需求的话，请使用保存在数据库内的方式，尽量不要保存到Cookie内。

Django的session框架默认启用，并已经注册在app设置内，如果真的没有启用，那么参考下面的内容添加有说明的那两行，再执行migrate命令创建数据表，就可以使用session了。

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions', # 这一行

'django.contrib.messages',

'django.contrib.staticfiles',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware', # 这一行

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

当session启用后，传递给视图request参数的HttpRequest对象将包含一个session属性，就像一个字典对象一样。你可以在Django的任何地方读写request.session属性，或者多次编辑使用它。

下面是session使用参考：

**class** **backends.**base**.**SessionBase

*# 这是所有会话对象的基类，包含标准的字典方法:*

\_\_getitem\_\_(key)

Example: fav\_color **=** request**.**session['fav\_color']

\_\_setitem\_\_(key, value)

Example: request**.**session['fav\_color'] **=** 'blue'

\_\_delitem\_\_(key)

Example: **del** request**.**session['fav\_color'] *# 如果不存在会抛出异常*

\_\_contains\_\_(key)

Example: 'fav\_color' **in** request**.**session

get(key, default**=**None)

Example: fav\_color **=** request**.**session**.**get('fav\_color', 'red')

pop(key, default**=**\_\_not\_given)

Example: fav\_color **=** request**.**session**.**pop('fav\_color', 'blue')

# 类似字典数据类型的内置方法

keys()

items()

setdefault()

clear()

# 它还有下面的方法：

flush()

# 删除当前的会话数据和会话cookie。经常用在用户退出后，删除会话。

set\_test\_cookie()

# 设置一个测试cookie，用于探测用户浏览器是否支持cookies。由于cookie的工作机制，你只有在下次用户请求的时候才可以测试。

test\_cookie\_worked()

# 返回True或者False，取决于用户的浏览器是否接受测试cookie。你必须在之前先调用set\_test\_cookie()方法。

delete\_test\_cookie()

# 删除测试cookie。

set\_expiry(value)

# 设置cookie的有效期。可以传递不同类型的参数值：

• 如果值是一个整数，session将在对应的秒数后失效。例如request.session.set\_expiry(300) 将在300秒后失效.

• 如果值是一个datetime或者timedelta对象, 会话将在指定的日期失效

• 如果为0，在用户关闭浏览器后失效

• 如果为None，则将使用全局会话失效策略

失效时间从上一次会话被修改的时刻开始计时。

get\_expiry\_age()

# 返回多少秒后失效的秒数。对于没有自定义失效时间的会话，这等同于SESSION\_COOKIE\_AGE.

# 这个方法接受2个可选的关键字参数

• modification:会话的最后修改时间（datetime对象）。默认是当前时间。

•expiry: 会话失效信息，可以是datetime对象，也可以是int或None

get\_expiry\_date()

# 和上面的方法类似，只是返回的是日期

get\_expire\_at\_browser\_close()

# 返回True或False，根据用户会话是否是浏览器关闭后就结束。

clear\_expired()

# 删除已经失效的会话数据。

cycle\_key()

# 创建一个新的会话秘钥用于保持当前的会话数据。django.contrib.auth.login() 会调用这个方法。

基本上背下来上面的内容，Django的session你就可以信手拈来了。

**一、使用session**

下面结合我们的项目实战，使用session。

首先，修改login/views.py中的login()视图函数：

**def** **login**(request):

**if** request**.**session**.**get('is\_login',None):

**return** redirect("/index/")

**if** request**.**method **==** "POST":

login\_form **=** forms**.**UserForm(request**.**POST)

message **=** "请检查填写的内容！"

**if** login\_form**.**is\_valid():

username **=** login\_form**.**cleaned\_data['username']

password **=** login\_form**.**cleaned\_data['password']

**try**:

user **=** models**.**User**.**objects**.**get(name**=**username)

**if** user**.**password **==** password:

request**.**session['is\_login'] **=** True

request**.**session['user\_id'] **=** user**.**id

request**.**session['user\_name'] **=** user**.**name

**return** redirect('/index/')

**else**:

message **=** "密码不正确！"

**except**:

message **=** "用户不存在！"

**return** render(request, 'login/login.html', locals())

通过下面的if语句，我们不允许重复登录：

if request.session.get('is\_login',None):

return redirect("/index/")

通过下面的语句，我们往session字典内写入用户状态和数据：

request.session['is\_login'] = True

request.session['user\_id'] = user.id

request.session['user\_name'] = user.name

你完全可以往里面写任何数据，不仅仅限于用户相关！

既然有了session记录用户登录状态，那么就可以完善我们的登出视图函数了：

**def** **logout**(request):

**if** **not** request**.**session**.**get('is\_login', None):

*# 如果本来就未登录，也就没有登出一说*

**return** redirect("/index/")

request**.**session**.**flush()

*# 或者使用下面的方法*

*# del request.session['is\_login']*

*# del request.session['user\_id']*

*# del request.session['user\_name']*

**return** redirect("/index/")

flush()方法是比较安全的一种做法，而且一次性将session中的所有内容全部清空，确保不留后患。但也有不好的地方，那就是如果你在session中夹带了一点‘私货’，会被一并删除，这一点一定要注意。

**二、完善页面**

有了用户状态，就可以根据用户登录与否，展示不同的页面，比如导航条内容：

首先，修改base.html文件：

{% load staticfiles %}

***<!DOCTYPE html>***

<html lang**=**"zh-CN">

<head>

<meta charset**=**"utf-8">

<meta http-equiv**=**"X-UA-Compatible" content**=**"IE=edge">

<meta name**=**"viewport" content**=**"width=device-width, initial-scale=1">

*<!-- 上述3个meta标签\*必须\*放在最前面，任何其他内容都\*必须\*跟随其后！ -->*

<title>{% block title %}base{% endblock %}</title>

*<!-- Bootstrap -->*

<link href**=**"{% static 'bootstrap-3.3.7-dist/css/bootstrap.min.css' %}" rel**=**"stylesheet">

*<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->*

*<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->*

*<!--[if lt IE 9]>*

*<script src="https://cdn.bootcss.com/html5shiv/3.7.3/html5shiv.min.js"></script>*

*<script src="https://cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>*

*<![endif]-->*

{% block css %}{% endblock %}

</head>

<body>

<nav class**=**"navbar navbar-default">

<div class**=**"container-fluid">

*<!-- Brand and toggle get grouped for better mobile display -->*

<div class**=**"navbar-header">

<button type**=**"button" class**=**"navbar-toggle collapsed" data-toggle**=**"collapse" data-target**=**"#my-nav" aria-expanded**=**"false">

<span class**=**"sr-only">切换导航条</span>

<span class**=**"icon-bar"></span>

<span class**=**"icon-bar"></span>

<span class**=**"icon-bar"></span>

</button>

<a class**=**"navbar-brand" href**=**"#">Mysite</a>

</div>

*<!-- Collect the nav links, forms, and other content for toggling -->*

<div class**=**"collapse navbar-collapse" id**=**"my-nav">

<ul class**=**"nav navbar-nav">

<li class**=**"active"><a href**=**"/index/">主页</a></li>

</ul>

<ul class**=**"nav navbar-nav navbar-right">

{% if request.session.is\_login %}

<li><a href**=**"#">当前在线：{{ request.session.user\_name }}</a></li>

<li><a href**=**"/logout/">登出</a></li>

{% else %}

<li><a href**=**"/login/">登录</a></li>

<li><a href**=**"/register/">注册</a></li>

{% endif %}

</ul>

</div>*<!-- /.navbar-collapse -->*

</div>*<!-- /.container-fluid -->*

</nav>

{% block content %}{% endblock %}

*<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->*

<script src**=**"{% static 'js/jquery-3.2.1.js' %}"></script>

*<!-- Include all compiled plugins (below), or include individual files as needed -->*

<script src**=**"{% static 'bootstrap-3.3.7-dist/js/bootstrap.min.js' %}"></script>

</body>

</html>

通过if判断，当登录时，显示当前用户名和登出按钮。未登录时，显示登录和注册按钮。

注意其中的模板语言，{{ request }}这个变量会被默认传入模板中，可以通过圆点的调用方式，获取它内部的{{ request.session }}，再进一步的获取session中的内容。其实{{ request }}中的数据远不止此，例如{{ request.path }}就可以获取先前的url地址。

再修改一下index.html页面，根据登录与否的不同，显示不同的内容：

{% extends 'base.html' %}

{% block title %}主页{% endblock %}

{% block content %}

{% if request.session.is\_login %}

<h1>你好,{{ request.session.user\_name }}！欢迎回来！</h1>

{% else %}

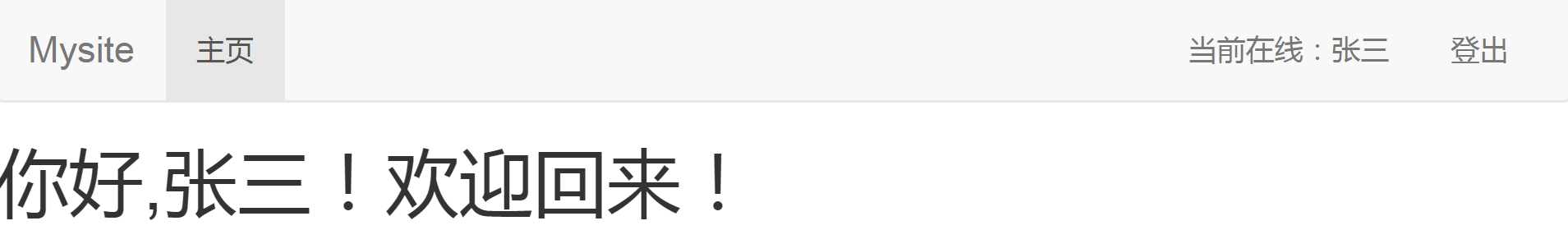
<h1>你尚未登录，只能访问公开内容！</h1>

{% endif %}

{% endblock %}

**三、效果展示**

这是用户‘张三’登录后的界面：



这是未登录用户的界面：

