学了一阵子的Python，了解到Django(家狗)的强大就像去接触。今天就写了这次的笔记了。

操作环境：

Win10 64位

Python 3.6.2

Django1.11.3

PyCham

其中，这次笔记中会记录本人在学习建站的时候遇到的问题，有大神如果觉得本人解决的问题有误或者不方便的话，希望能够给本人指出

学习前记：

接触Python不久，就想从项目开始熟悉Python。这次的笔记也是为自己以后学习的路上带上更多的方便。

安装Python就不说了，网上的教程也是很多的，而且安装不难。接下来就是安装Django了，安装《Python web实战》的书中使用的是Django1.9版本的，其实区别也不大。

可以访问 Django 官方网站进行下载：<https://www.djangoproject.com/>

学习Python之后，我安装第三方包使用最多的也就是pip了，所以今天也是使用pip来安装Django了，通过CMD命令打开Windows提示符，输入pip install Django 就可以安装了。因为不同宽带可以带来的网速也会不同，如果出现在”Read time out”问题，可以使用“pip –default-timeout=100 install –U django”来安装，这条命令的意思是设置更长的等待时间。

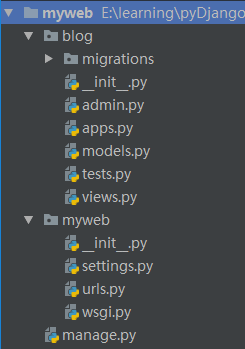
安装完成django之后，我们就可以来搭建一个属于自己的blog了。

我们现在一个自己熟悉的盘下新建一个文件夹，我是在E盘下新建的一个pyDjango。

1.创建项目和应用

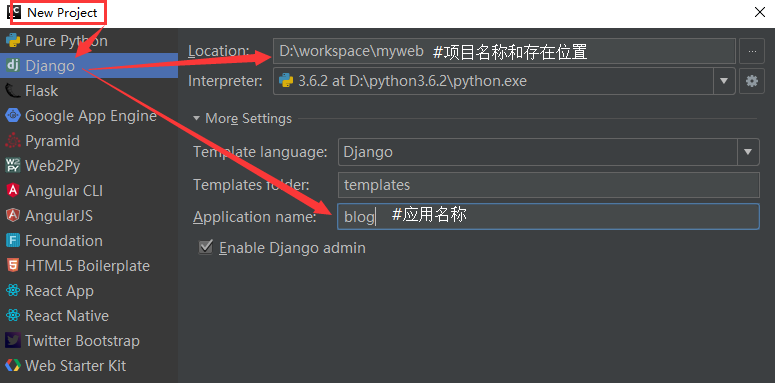


我们通过pycharm打开创建的项目包括什么吧



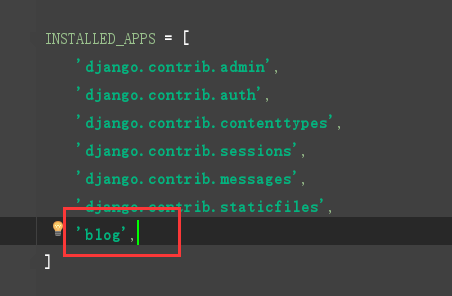
这些就是我们创建的项目和应用。

另外还有一种方法创建django项目的是：通过pycharm创建django项目



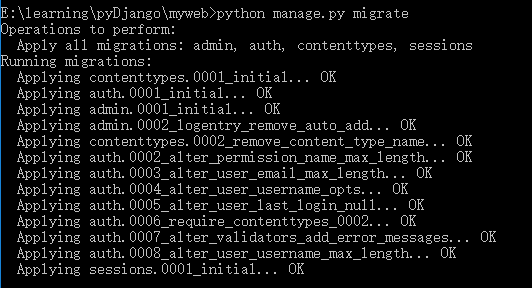
上面两种方法都可以创建我们的django项目的，通过pycharm会自动帮我们创建一个templates文件夹，后面我们需要用上的。作为学习笔记，我们就先从教程开始吧，熟悉了命令之后，后面我们通过工具创建就简单很多了。

我们进入myweb/myweb/目录下的setting.py文件下添加blog应用:



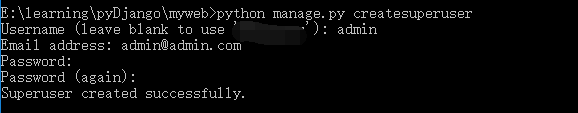
这个“blog”正是我们执行“python manage.py startapp blog”命令所创建的应用

Django中一个最强大的部分是自带了一个admin后台，为我们提供了一个强大的界面接口，让我们非常方便的操作数据库的数据

首先，我们需要进行数据库同步:  


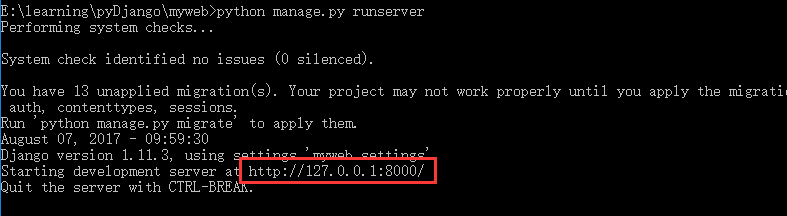
看到这些数据，你就知道数据库同步成功了

现在我们就可以为我们网站创建一个超级管理员了

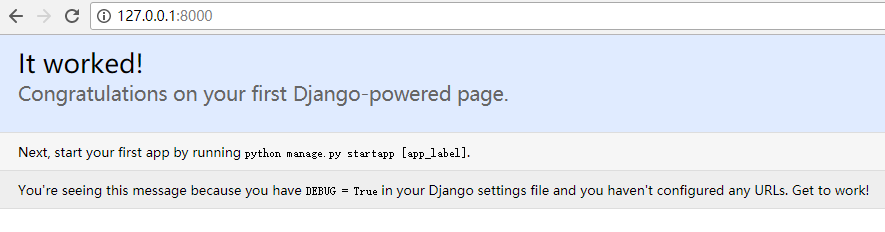


看到创建成功，就说明，我们可以使用当前管理账户登录我们的后台了

现在启动服务吧！

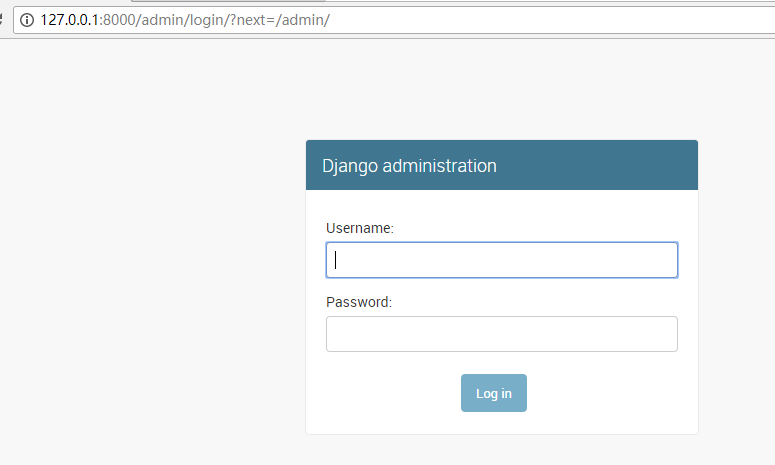


我们在浏览器中输入:127.0.0.1:8000

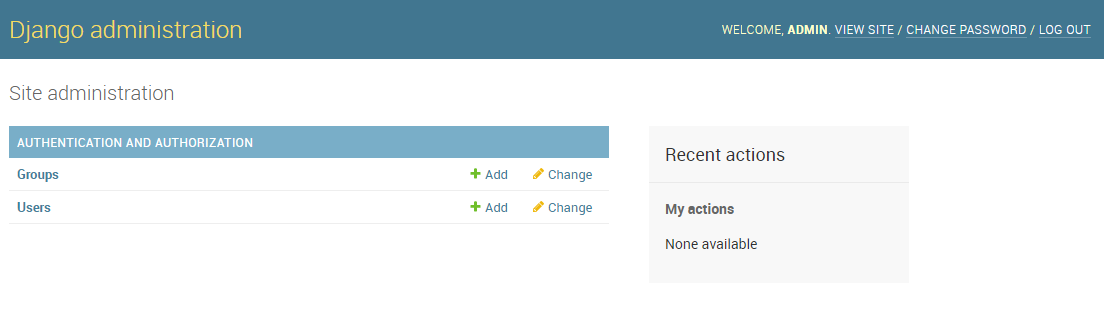


这说明我们启动服务成功了，后面8000端口号是默认的，我们可以通过启动的过程修改端口号。

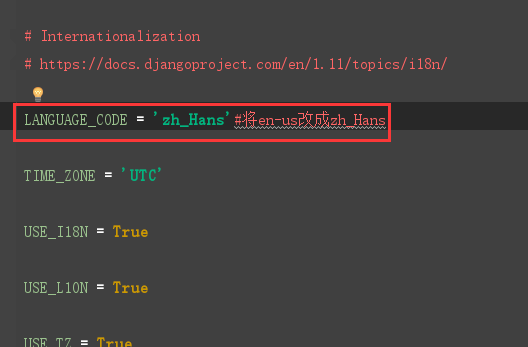
现在我们登录后台吧：127.0.0.1:8000/admin



用户名和密码就是我们刚才创建的时候输入的



我们可以看到后台的真面目了，有一点就是我们如果不喜欢看英文版的后台，我们可以回到setting.py设置



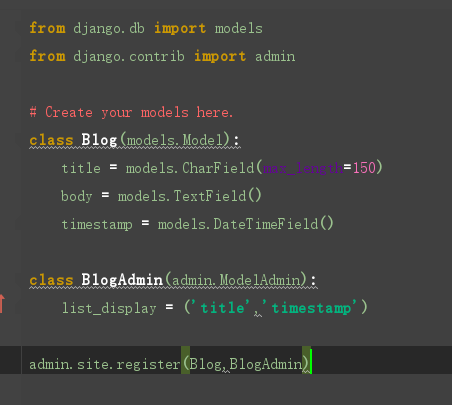
现在我们可以刷新一下页面:F5



现在看到的就是中文版的了。

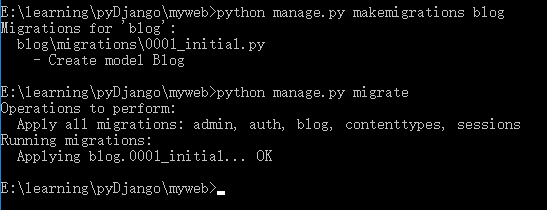
创建blog模型

打开myweb/blog/models.py文件，编写以下代码:

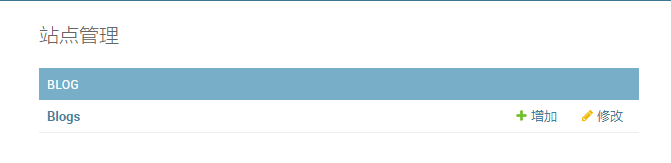


这段代码的意思是创建一个Blog的数据表，分别包含title，body，timestamp三个字段。

我们去命令行创建对应的数据库表，执行以下命令:



我们每创建一个数据表都要数据库同步一次，要不然我们在后台是看不到我们的数据的，现在我们去查看以下我们刚才创建的数据库表吧



这时我们就可以看到创建的数据库表了，点击增加：



输入一些内容之后，就可以点击保存了

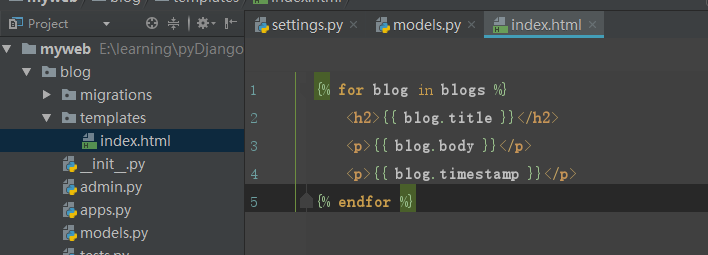


这就是我们刚才创建的一篇blog文章了

现在我们就需要把我们在后台创建的文章显示在前端的web页面了

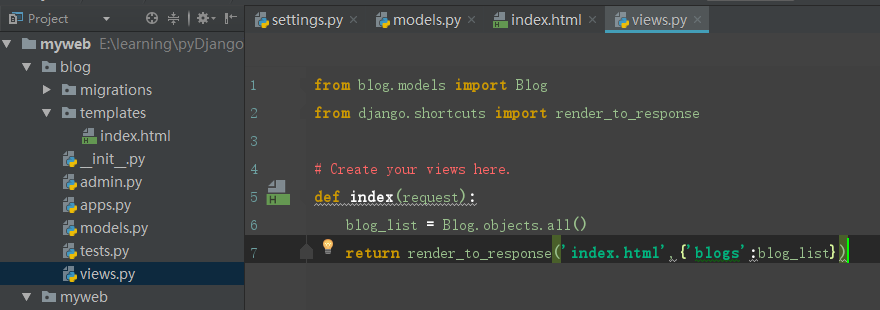
我们去myweb/blog目录下创建templates目录吧，这也是我刚才说的在pycharm创建的Django项目的时候的方便之处了。

在templates下创建一个index.html文件，可以把原来的html代码全清空，然后写以下代码:



写完之后，我们需要把这段代码放进django的views.py进行解析，然后通过url.py文件获取链接显示出来，这就是django的思路。

所以，我们现在就要去myweb/blog/views.py文件：



这段代码的意思就是通过Blog.objects.all()得到blog表中的所有数据，并赋值给blog\_list变量。最终将blog\_list发送到index.html页面

现在就需要去做最后一步，配置url

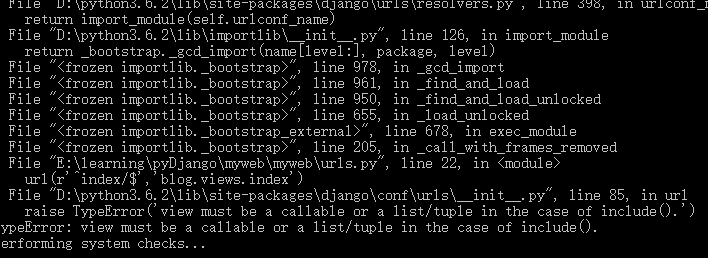
打开myweb/myweb/urls.py



在index后面添加/$是为了防止输入链接错误

现在打开浏览器访问:127.0.0.1:8000/index/

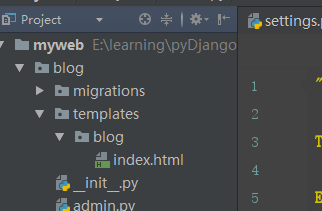
看到一个无法访问服务器页面 ，去到cmd查看以下



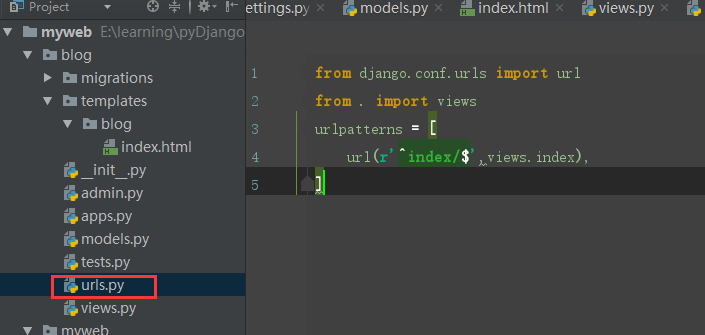
这是什么原因呢，当时自己也是有点蒙蔽了，查看以下，原来是1.10以上就再也不支持上面那种url写法了，后面自己也是改了不少地方了，同时也是为了保证安全性的。

下面就是解决的方案，跟着做:

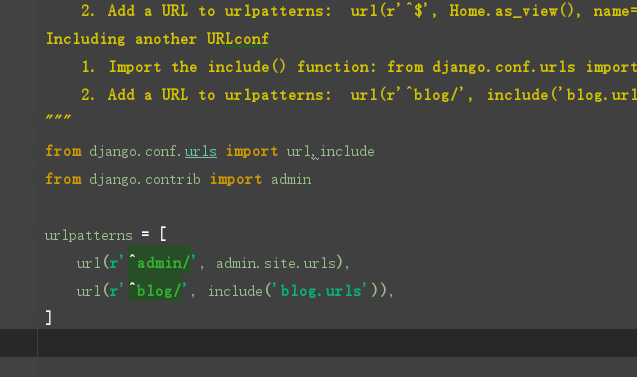
首先在templates下创建一个blog文件将index文件放进去，这样就可以避免不同链接访问同一页面的冲突



第二步就是在myweb/blog下创建一个urls.py文件，这个文件的意思是为了保证以后创建更多文件夹访问产生url错误。



好了，现在就把myweb/myweb/下的urls修改一下，



现在我们再次去访问:127.0.0.1:8000/blog/index

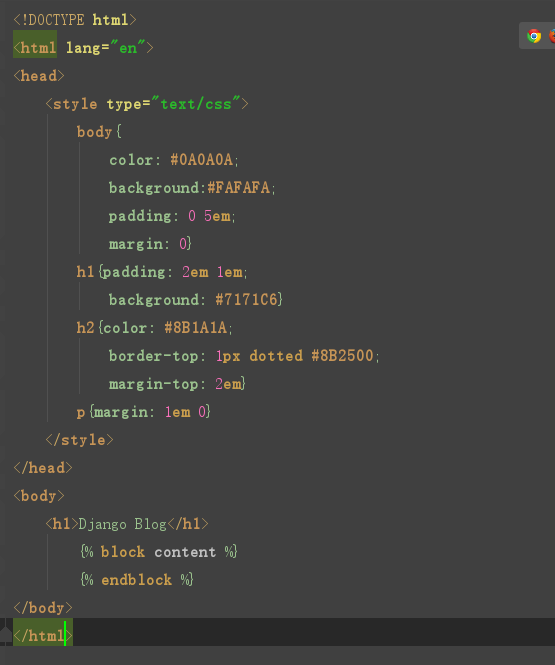
我们可以看到中间添加了blog的路径，是因为我们刚才在templates下创建的blog文件夹



好了，现在页面正确的显示出来了。这是我在学习django上遇到的第一个问题以及解决的方法，希望大家提供更多方便的解决方法给本人学习。

现在我们可以给blog添加点样式了

我们在myweb/blog/templates/blog目录下创建base.html模板：



添加完base.html模板之后，我们需要在index.html中引入



再次刷新页面，我们看看新的效果：



我们看到的是这个界面，看到错误就难受，提示：不存在base.html这个文件

现在我们来解决这个问题吧：

分析一下错误信息：templates中找不到base.html文件，那我们把base.html文件放在templates下试试看吧。

果然，问题就这么解决了，这是我在学习django遇到的第二个问题以及解决方法。



这就是我们需要现实的内容页面了。

好了，现在可以登录admin后台继续添加文章了。

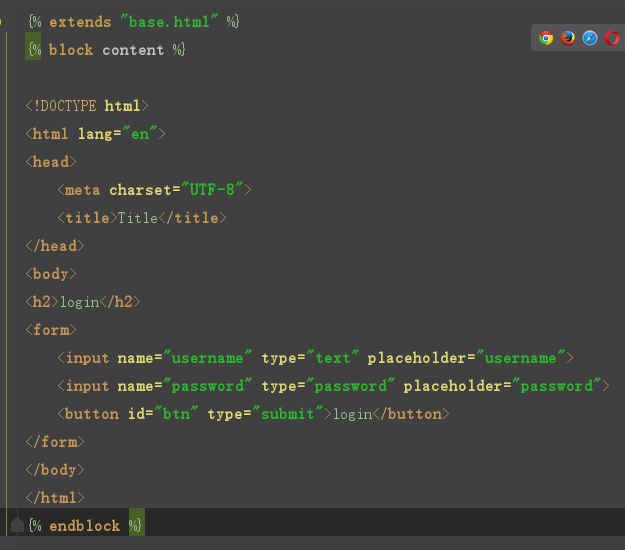
现在我们进一步对django进行学习吧，也就是django的MTV开发模式

M代表模型(Model)，即数据存取层；该层处理与数据相关的所有事务；

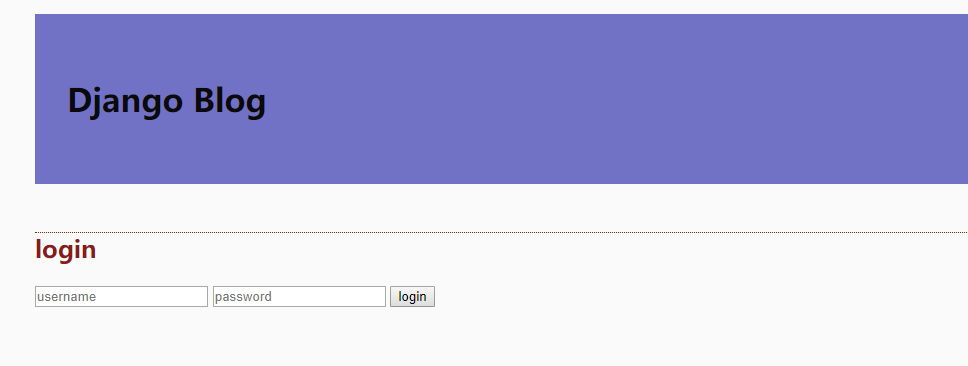
T代表模板(Template)，即表现层； 该层处理与表现相关的决定；

V代表视图(View)，即业务逻辑层；该层包含存取模型及调取恰当模板的相关逻辑；

先写个登录视图吧:



这是重新写的index.html文件，所以之前的文件内容已经删掉了。我们访问以下:127.0.0.1/8000/blog/index 看看效果吧

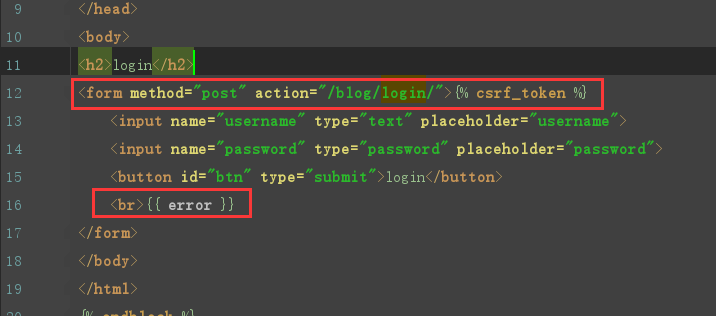


这就是我们想看到的效果了，但是目前这个页面也只是一个静态页面，我们如果要实现真正的登录功能的话，就需要继续写代码了。

当我们点击login按钮的时候，表单要以什么方式提交(Get或者Post请求)呢?

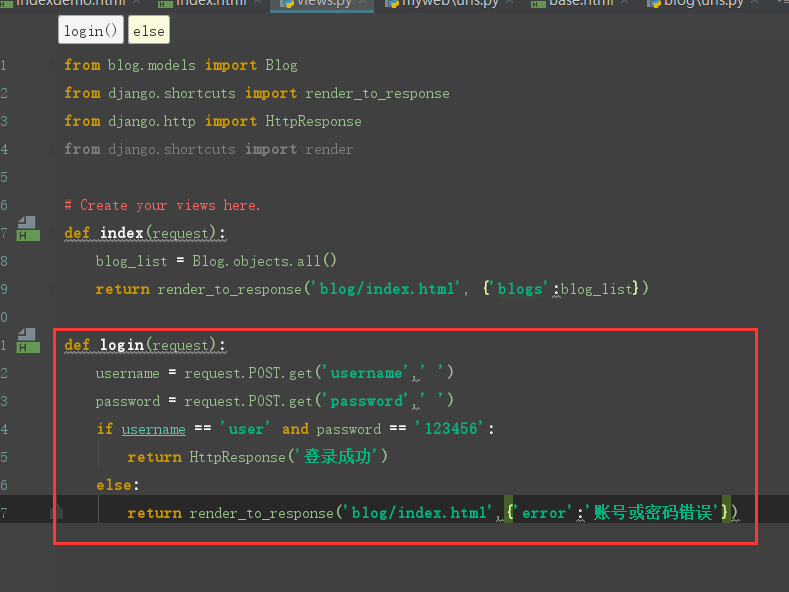
其中，Get请求可以说是一个不安全的请求方式，所以我们就不讲Get请求，直接来Post请求方式好了。

在原来登录的index.html文件中添加post方法:

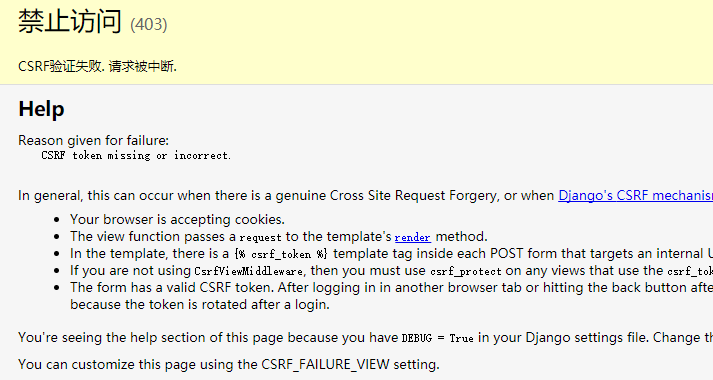


然后再到url中配置文件

打开myweb/blog/urls.py文件添加一条配置:



好了，现在我们去访问吧：

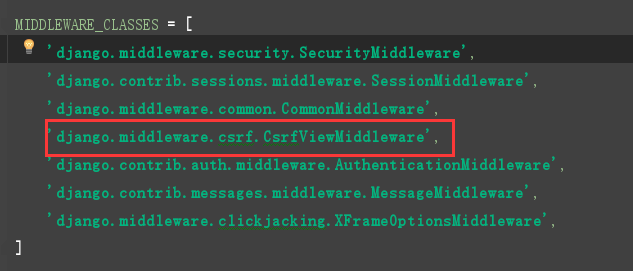


抛出了一个禁止访问的失败提示，又是一个问题。

现在我们去看看是什么错误吧:

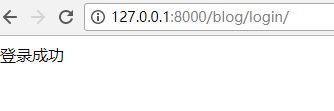
CSRF token丢失或者不正确。

针对这个问题，我们去setting文件中，查看是否存在这个文件吧,

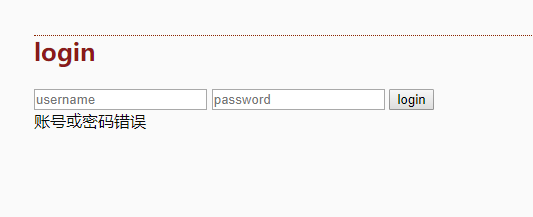


CSRF这个文件是存在的，那究竟还是什么错误的呢?

在解决问题之前，看了网上很多人都说使用,context\_instance=RequestContext(request)强制转换的方法，可是我始终是不能解决这个问题。我是先把这一段给注释掉了，后来发现成功了。研究了一下Django 403问题，发现这是一个Django为开发者提供的一个保护机制，为了防止网站被攻击，因为现在我们不用考虑这个问题，所以我把它注释掉了。后面如果在不注释的情况下可以解决问题的话，再来跟大家说。

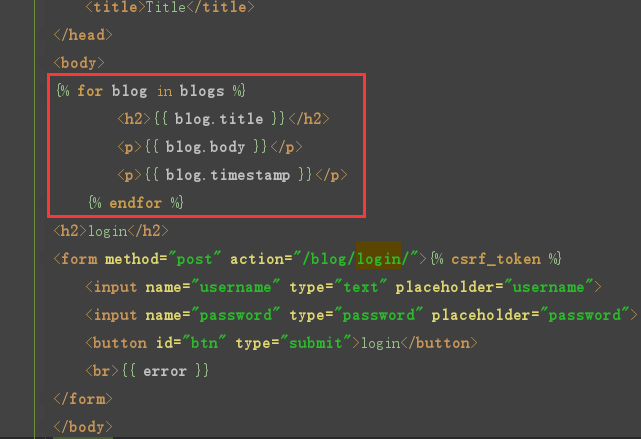


登录失败的情况:



现在我们基本的问题是得到暂时的解决了。

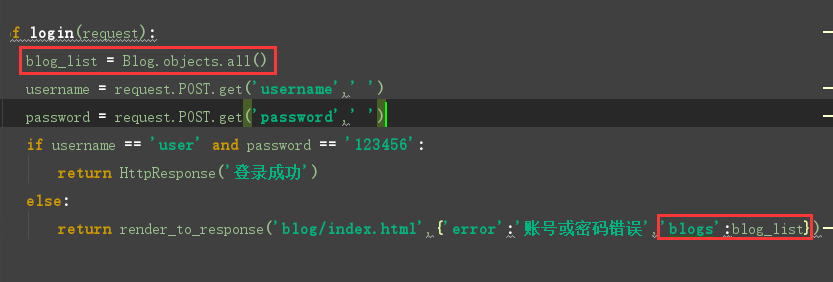
为了让我们还能显示出博客的文章，我们把之前的代码给添加上去好了



现在index页面的内容是:



当我们输入账号或者密码时，也返回当前文章:





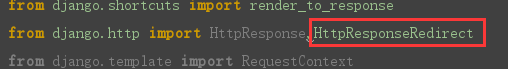
现在文章基本完成了。

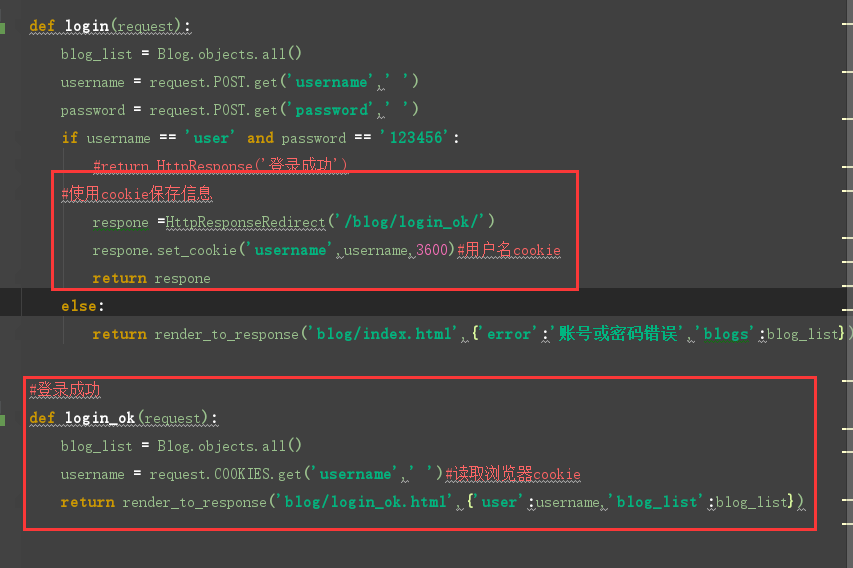
接下来，进行我们下一个页面的内容了。这就要涉及到cookie和session两个机制的知识了。关于这两个我也不多详细讲了。举个栗子，就是cookie是我们的存折，session是我们的银行卡；也就是session比cookie安全。

演示两个保存信息的操作:

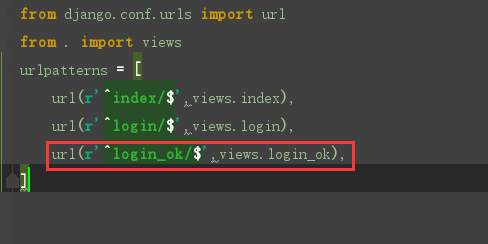
使用cookie保存信息

修改views.py文件



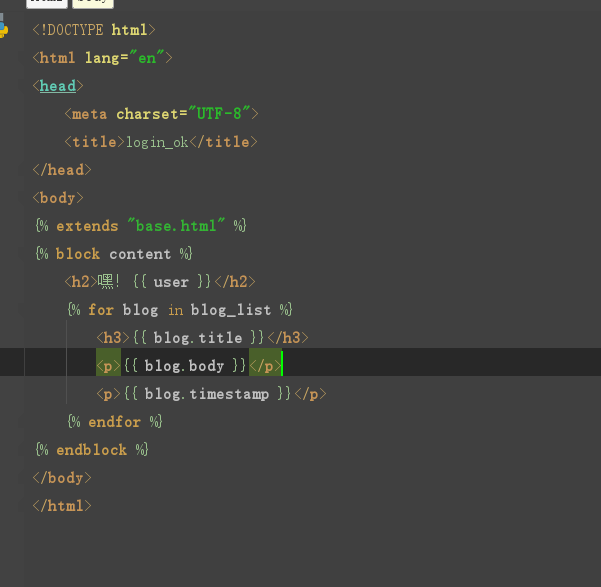


在myweb/blog/urls.py中添加login\_ok链接



现在我们就去添加login\_ok的html页面

在templates中新建一个login\_ok的html文件:



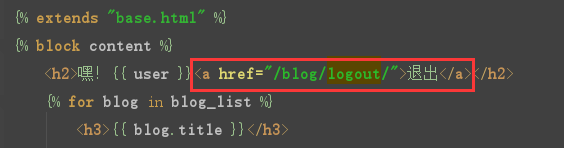
保存，我们就可以访问了，



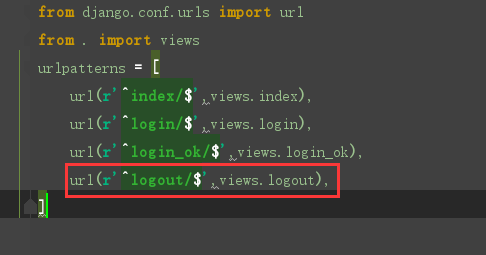
这就是我们想要的效果了。是不是越来越觉得有成就感了

现在我们就需要一个可以退出的按钮了，当我们点击退出按钮，浏览器的cookie也随之删除，也就是“安全退出”。

我们在login\_ok页面添加一个logout的按钮



再去myweb/blog/urls.py添加logout的链接



最后在views.py中添加logout函数：



退出之后，就会返回到首页了。