# 注册登录项目操作步骤

前提：已安装python,pycharm,mysql,pymysql

## 一、登录功能

1. django-admin.py startproject mysite //创建项目 CMD
2. python manage.py startapp login //创建APP CMD
3. create database csvt; //mysql client
4. 修改setting.py；

* INSTALLED\_APPS中添加’login’,
* 修改时区TIME\_ZONE=’Asia/Shanghai’;
* 修改语言LANGUAGE\_CODE = ‘zh-hans’

**验证：**python manage.py runserver //终端输入,打开链接<http://127.0.0.1:8000/>

**数据库后端：**

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'csvt',

'USER': 'test',

'PASSWORD': 'test123',

'HOST':'localhost',

'PORT':'3306',

}

}

login/\_init\_.py 文件写入

import pymysql  
pymysql.install\_as\_MySQLdb()

1. 模型设计（数据库设计）

用户登录和注册项目，需要一种用户表User保存用户信息：

* 用户名
* 密码
* 性别
* 创建时间

Login/Models.py

from django.db import models  
  
class User(models.Model):  
  
 gender = (  
 ('male',"男"),  
 ('female',"女"),  
 )  
  
 name = models.CharField(max\_length=128,unique=True)  
 password = models.CharField(max\_length=256)  
 email = models.EmailField(unique=True)  
 sex = models.CharField(max\_length=32,choices=gender,default='男')  
 c\_time = models.DateTimeField(auto\_now\_add=True)  
   
 def \_\_str\_\_(self):  
 return self.name  
   
 class Meta:  
 ordering = ["-C\_time"]  
 verbose\_name = "用户"  
 verbose\_name\_plural = "用户"

 name： 必填，最长不超过128个字符，并且唯一，不同名

 password： 必填，最长不超过256个字符

 sex: 使用了一个choice，只能选择男/女，默认为男

 **str**，人性化显示对象信息

 元数据中定义是按照用户创建时间的反序排列

6、创建记录和数据表：

python manage.py makemigrations # 让 Django 知道我们在我们的模型有一些变更

python manage.py migrate TestModel # 创建表结构

注意：进入mysql客户端验证结果，用命令 show databases; use csvt; show tables;

7、admin后台

在admin.py中注册模型：

from . import models

admin.site.register(models.User)

创建超级管理员：

python manage.py createsuperuser

输入用户名，邮件，密码

8、启动开发服务器

http://127.0.0.1:8000/admin

9、路由设置

url->视图->前端模板

|  |  |  |  |
| --- | --- | --- | --- |
| **URL** | **视图** | **模板** | **说明** |
| /index/ | login.views.index() | index.html | 主页 |
| /login// | login.views.login() | login.html | 登录界面 |
| /register/ | login.views.register() | register.html | 注册界面 |
| /logout/ | login.views.logout() | 不需要 | 退出界面 |

mysite/urls.py

from django.conf.urls import url

from django.contrib import admin

from login import views

urlpatterns = [

url(r’^admin/’,admin.site.urls),

url(r’^index/’,views.index),

url(r’^login/’,views.login),

url(r’^register/’,views.register),

url(r’^logout/’,views.logout),

]

视图

login/views.py

from django.shortcuts import render

from django.shortcuts import redirect

def index(request):

pass

return render(request,’login/index.html’)

def login(request):

pass

return render(request,’login/login.html’)

def register(request):

pass

return render(request,’login/register.html’)

def logout(request):

pass

return redirect(’/index/’)

10、前端页面设置

* + 引入bootstrap-3.3.7-dist;含js,css,font
  + 下载地址[https://v3.bootcss.com/getting-started/#download](https://v3.bootcss.com/getting-started/" \l "download)
  + 下载jquery.js文件 <http://www.jq22.com/jquery-info122>
  + 在Django根目录下创建static整体导入

静态文件设置

用于指定静态文件的搜索目录

setting.py

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, "static"),

]

创建base.html模板

引入bootstrap模板导航条，引入bootstrap和jquery静态文件

{% 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">

<li><a href="/login/">登录</a></li>

<li><a href="/register/">注册</a></li>

</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>

注意：django3.0以上用<% loadstatic %>

11、登录视图

用户通过login.html中的表单来填写用户名和密码，并且以POST的方式发送到服务器的/login/地址。服务器通过/login/views.py中的login()视图函数，接收并且处理这一请求。

/login/views.py:

def login(request):

if request.method == "POST":

username = request.POST.get('username')

password = request.POST.get('password')

print(username, password)

return redirect('/index/')

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

CSRF验证：

Login.html

<form class='form-login' action="/login/" method="post">

{% csrf\_token %} <!--加这句-->

<h2 class="text-center">欢迎登录</h2>

<div class="form-group">

......

</form>

数据验证：

后台需要验证前端填写的数据

def login(request):

if request.method == "POST":

username = request.POST.get('username', None)

password = request.POST.get('password', None)

#确保当数据请求中没有username键时不会抛出异常，而是返回一个我们指定的默认值None；

#添加以下两句

if username and password: # 确保用户名和密码都不为空

username = username.strip()

# 用户名字符合法性验证

# 密码长度验证

# 更多的其它验证.....

return redirect('/index/')

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

验证用户名和密码：

和数据库已注册的用户名和密码进行比较：

def login(request):

if request.method == "POST":

username = request.POST.get('username', None)

password = request.POST.get('password', None)

if username and password: # 确保用户名和密码都不为空

username = username.strip()

# 用户名字符合法性验证

# 密码长度验证

# 更多的其它验证.....

#添加try except语句

try:

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

if user.password == password:  
 return redirect('/index/')  
 else:  
 message = "密码不正确"

except:

message = “用户不存在”

return render(request,’../templates/login.html’,{”message”=message})

return render(request, '../templates/login.html')

不合法的内容前端给出提示：

login.html

<form class='form-login' action="/login/" method="post">

<!--添加以下两句做前端提示-->

{% if message %}

<div class="alert alert-warning">{{ message }}</div>

{% endif %}

{% csrf\_token %}

<h2 class="text-center">欢迎登录</h2>

<div class="form-group">

<label for="id\_username">用户名：</label>

<input type="text" name='username' class="form-control" id="id\_username" placeholder="Username" autofocus required>

</div>

<div class="form-group">

<label for="id\_password">密码：</label>

<input type="password" name='password' class="form-control" id="id\_password" placeholder="Password" required>

</div>

<button type="reset" class="btn btn-default pull-left">重置</button>

<button type="submit" class="btn btn-primary pull-right">提交</button>

</form>

12、Django表单

Django在内部集成了一个表单功能，以面向对象的方式，直接使用python代码生成HTML表单代码，专门帮助我们快速处理表单相关内容。

编写Django的form表单，非常类似我们在模型系统里编写一个模型。Django在内部集成一个表单功能，以面向对象的方式，直接使用python代码生成HTML表单代码，专门帮助我们快速处理表单的相关内容。

**创建表单模型**

新建/login/forms.py

from django import forms

class UserForm(forms.Form):

username = forms.CharField(label="用户名", max\_length=128)

password = forms.CharField(label="密码", max\_length=256, widget=forms.PasswordInput)

widget=forms.PasswordInput用于指定该字段在form表单里表现为<input type=’password’/>

也就是密码输入框。

**修改视图**

login/view.py

def login(request):

if request.method == "POST":

login\_form = forms.UserForm(request.POST) #修改此处

message = "请检查填写的内容！"

if login\_form.is\_valid(): #修改此处

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

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

try:

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

if user.password == password:

return redirect('/index/')

else:

message = "密码不正确！"

except:

message = "用户不存在！"

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

login\_form = forms.UserForm() #修改此处

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

**修改login页面**

{% extends 'base.html' %}

{% load staticfiles %}

{% block title %}登录{% endblock %}

{% block css %}<link href="{% static 'css/login.css' %}" rel="stylesheet"/>{% endblock %}

{% block content %}

<div class="container">

<div class="col-md-4 col-md-offset-4">

<form class='form-login' action="/login/" method="post">

{% if message %}

<div class="alert alert-warning">{{ message }}</div>

{% endif %}

{% csrf\_token %}

<h2 class="text-center">欢迎登录</h2>

{{ login\_form }} <!—添加此处🡪

<button type="reset" class="btn btn-default pull-left">重置</button>

<button type="submit" class="btn btn-primary pull-right">提交</button>

</form>

</div>

</div> <!-- /container -->

{% endblock %}

13、图片验证码

**安装captcha**

pip install django-simple-captcha

**注册captcha**

setting.py

INSTALLED\_APP = [

‘captcha’

]

**生成数据表**

Python manage.py migrate

**添加URL路由**

mysite/urls.py

from django.conf.urls import include

urlpatterns = [

url(r'^captcha', include('captcha.urls')) # 增加这一行

]

修改forms.py

from captcha.fields import CaptchaField #添加这一行

class UserForm(forms.Form):

captcha = CaptchaField(label='验证码') #添加这一行

修改login.html（两种方式）

使用{{login\_form}}

手动渲染修改

<div class="form-group">

{{ login\_form.captcha.errors }}

{{ login\_form.captcha.label\_tag }}

{{ login\_form.captcha }}

</div>

14、使用session

识别用户并保持用户状态

* 浏览器中的cookie存放不重要的数据信息
* 服务器中的session存放用户数据和状态
* Session依赖cookie
* Django默认启用session框架

修改login/view.py

通过IF语句，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：

if user.password == password:

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

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

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

return redirect('/index/')

登出界面

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()

# flush会一次性清空session中所有内容，可以使用下面的方法

# 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中夹带私货，也会被一并删除，这一点一定要注意。

完善界面

base.html

{% 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 %}

未登录限制访问

def index(request):  
 if not request.session.get('is\_login',None):  
 return redirect('/login/')  
 return render(request,'../templates/index.html')

------------------------以上皆为登录功能-----------------------------------

## 二、注册功能

1、创建forms：

login/forms.py

class RegisterForm(forms.Form):

gender = (

('male', "男"),

('female', "女"),

)

username = forms.CharField(label="用户名", max\_length=128, widget=forms.TextInput(attrs={'class': 'form-control'}))

password1 = forms.CharField(label="密码", max\_length=256, widget=forms.PasswordInput(attrs={'class': 'form-control'}))

password2 = forms.CharField(label="确认密码", max\_length=256, widget=forms.PasswordInput(attrs={'class': 'form-control'}))

email = forms.EmailField(label="邮箱地址",widget=forms.EmailInput(attrs={'class':'form-control'}))

sex = forms.ChoiceField(label='性别', choices=gender)

captcha = CaptchaField(label='验证码')

2、完善register.html

和登录视图类似

{% extends 'base.html' %}

{% load static %}

{% block title %}注册{% endblock %}

{% block css %}<link href="{% static 'css/login.css' %}" rel="stylesheet"/>{% endblock %}

<meta charset="utf-8">

{% block content %}

<div class="container">

<div class="col-md-4 col-md-offset-4">

<form class='form-login' action="/login/" method="post">

{% if message %}

<div class="alert alert-warning">{{message}}</div>

{% endif %}

{% csrf\_token %} <!-- 跨站伪造请求攻击验证 -->

<h2 class="text-center">欢迎注册</h2>

<div class="form-group">

{{register\_form.username.label\_tag}}

{{register\_form.username}}

</div>

<div class="form-group">

{{register\_form.password1.label\_tag}}

{{register\_form.password1}}

</div>

<div class="form-group">

{{register\_form.password2.label\_tag}}

{{register\_form.password2}}

</div>

<div class="form-group">

{{register\_form.email.label\_tag}}

{{register\_form.email}}

</div>

<div class="form-group">

{{ login\_form.sex.label\_tag }}

{{ login\_form.sex }}

</div>

<div class="form-group">

{{ login\_form.captcha.errors }}

{{ login\_form.captcha.label\_tag }}

{{ register\_form.captcha }}

</div>

<a href="/login/"><ins>直接登录</ins></a>

<button type="submit" class="btn btn-primary pull-right">注册</button>

</form>

</div>

</div> <!-- /container -->

{% endblock %}

实现注册视图

/login/views.py

def register(request):

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

# 登录状态不允许注册。你可以修改这条原则！

return redirect("/index/")

if request.method == "POST":

register\_form = forms.RegisterForm(request.POST)

message = "请检查填写的内容！"

if register\_form.is\_valid(): # 获取数据

username = register\_form.cleaned\_data['username']

password1 = register\_form.cleaned\_data['password1']

password2 = register\_form.cleaned\_data['password2']

email = register\_form.cleaned\_data['email']

sex = register\_form.cleaned\_data['sex']

if password1 != password2: # 判断两次密码是否相同

message = "两次输入的密码不同！"

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

else:

same\_name\_user = models.User.objects.filter(name=username)

if same\_name\_user: # 用户名唯一

message = '用户已经存在，请重新选择用户名！'

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

same\_email\_user = models.User.objects.filter(email=email)

if same\_email\_user: # 邮箱地址唯一

message = '该邮箱地址已被注册，请使用别的邮箱！'

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

# 当一切都OK的情况下，创建新用户

new\_user = models.User()

new\_user.name = username

new\_user.password = password1

new\_user.sex = sex

new\_user.save()

return redirect('/login/') # 自动跳转到登录页面

register\_form = forms.RegisterForm()

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

从大体逻辑上，也是实例化一个registerForm的对象，然后使用is\_valid()验证数据，再从cleaned\_data中获取数据。

重点在于注册逻辑，首先两次输入的密码必须相同，其次不能存在相同用户名和邮箱，最后如果条件满足，利用ORM的API，创建一个用户实例，然后保存到数据库内。

对于注册的逻辑，不同的生产环境有不同的要求，请跟进实际情况自行完善。

密码加密

修改login/views.py;编写一个hash函数

import hashlib

def hash\_code(s, salt='mysite'):# 加点盐

h = hashlib.sha256()

s += salt

h.update(s.encode()) # update方法只接收bytes类型

return h.hexdigest()