[root@master wdata]# pip install --upgrade pip

[root@master wdata]# pip install django

[root@master wdata]# pip install mysqlclient

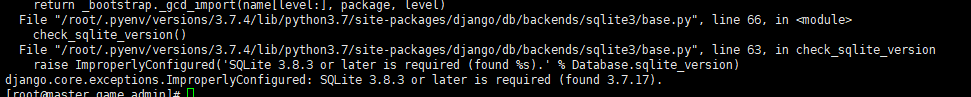
[root@master wdata]# pip install django-simple-captcha

[root@master wdata]# django-admin startproject game\_admin

[root@master wdata]# cd ./game\_admin

[root@master wdata]# python manage.py startapp login

#########################此时如果有如下报错###############################



是因为版本问题，此时我们修改sqlite的问题即可，处理方式如下

[root@master game\_admin]# cd /home/software/

[root@master software]# wget <http://www.sqlite.org/2019/sqlite-autoconf-3280000.tar.gz>

[root@master software]# tar zxvf sqlite-autoconf-3280000.tar.gz -C /usr/src/

[root@master software]# cd /usr/src/sqlite-autoconf-3280000/

[root@master sqlite-autoconf-3280000]# ./configure --prefix=/usr/local/sqlite

[root@master sqlite-autoconf-3280000]# make -j4

[root@master sqlite-autoconf-3280000]# make install

[root@master sqlite-autoconf-3280000]# mv /usr/bin/sqlite3 /usr/bin/sqlite3\_old

[root@master sqlite-autoconf-3280000]# cp /usr/local/sqlite/bin/sqlite3 /usr/bin/

[root@master sqlite-autoconf-3280000]# sqlite3 --version

IMG_257

[root@master sqlite-autoconf-3280000]# vim /etc/profile.d/sqlite.sh

export LD\_LIBRARY\_PATH="/usr/local/sqlite/lib"

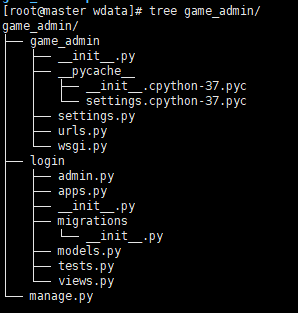
[root@master sqlite-autoconf-3280000]# bash 或者source /etc/profile.d/sqlite.sh 刷新

######################排错结束，下面继续处理###############################

此时命令顺利执行

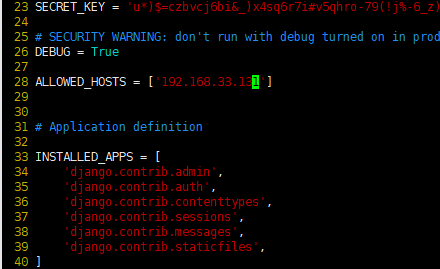
[root@master wdata]# python manage.py startapp login

完成后，初始目录结构如下：



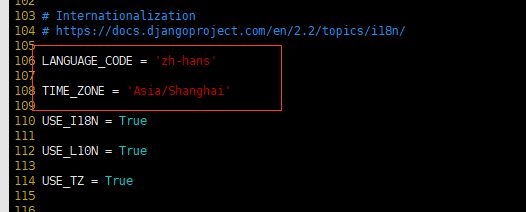
[root@master game\_admin]# vim settings.py

做如下修改（将ALLOWED\_HOSTS修改为服务server的ip）：



填入你的server服务ip地址

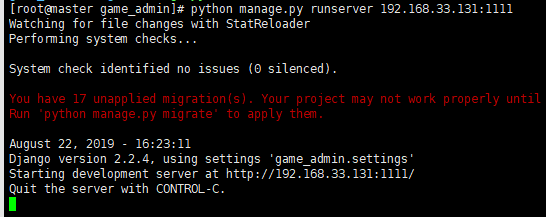
修改为中文和上海时间



然后启动服务（如图，在对应目录下）

[root@master game\_admin]# python manage.py runserver 192.168.33.131:1111

效果如图：





设置mysql数据库

[root@master wdata]# yum -y install mariadb\*

添加密码:mysql

MariaDB [(none)]> set password for root@localhost = password('123456');

增加远程访问权限：

MariaDB [(none)]> GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;

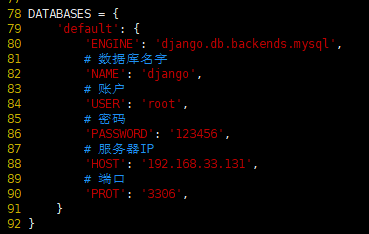
增加django库

MariaDB [(none)]> CREATE DATABASE django;

刷新数据库

MariaDB [(none)]> flush privileges;

修改settings.py文件，将sqlite3数据库改为mysql数据库

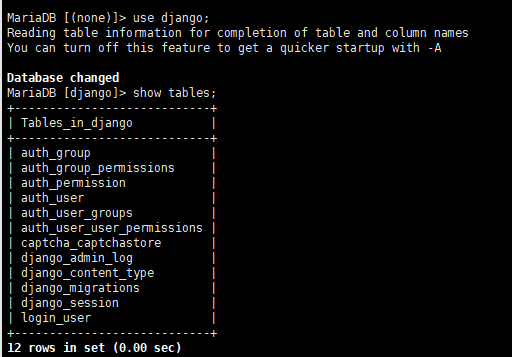


数据库迁移

[root@master game\_admin]# python manage.py makemigrations

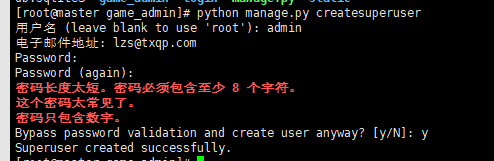
[root@master game\_admin]# python manage.py migrate

验证迁移是否成功，如图



此时，可以创建超级管理员权限了哦

[root@master game\_admin]# python manage.py createsuperuser



管理员账户：admin

管理员密码：123456

以上页面出现,说明django是没有问题了,现在我们开始做登录页面

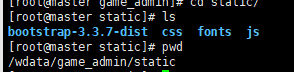
从[https://v3.bootcss.com/getting-started/#download](https://v3.bootcss.com/getting-started/" \l "download)

下载到对应的资源备用

建立资源存放目录:

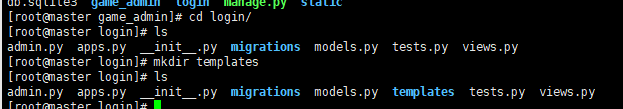
[root@master game\_admin]# mkdir static

将下载好的资源及压缩后放入static目录中



如果后面有确实东西的话,可以根据提示去下载对应的资源

在login的app中建立templates目录,备用



[root@master login]# mkdir templates

建立如下网页文件备用

[root@master templates]# touch base.html form.html index.html login.html register.html

IMG_258

具体代码如下

[root@master templates]# cat base.html

{% 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 '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="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 'js/bootstrap.min.js' %}"></script>

</body>

</html>

[root@master templates]# cat form.html

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

{% csrf\_token %}

<h2 class="text-center">欢迎登录</h2>

<div class="form-group">

<!-- <label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">

-->

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

{{ login\_form.username}}

</div>

<div class="form-group">

<!--<label for="exampleInputPassword1">Password</label>

<input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">

-->

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

{{ login\_form.password }}

</div>

<div class="form-group">

<label for="exampleInputFile">File input</label>

<input type="file" id="exampleInputFile">

<p class="help-block">Example block-level help text here.</p>

</div>

<div class="checkbox">

<label>

<input type="checkbox"> Check me out

</label>

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

[root@master templates]# cat form.html

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

{% csrf\_token %}

<h2 class="text-center">欢迎登录</h2>

<div class="form-group">

<!-- <label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">

-->

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

{{ login\_form.username}}

</div>

<div class="form-group">

<!--<label for="exampleInputPassword1">Password</label>

<input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">

-->

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

{{ login\_form.password }}

</div>

<div class="form-group">

<label for="exampleInputFile">File input</label>

<input type="file" id="exampleInputFile">

<p class="help-block">Example block-level help text here.</p>

</div>

<div class="checkbox">

<label>

<input type="checkbox"> Check me out

</label>

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

[root@master templates]# ls

base.html form.html index.html login.html register.html

[root@master templates]# cat index.html

{% 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 %}

[root@master templates]# cat login.html

{% extends 'base.html' %}

{% load staticfiles %}

{% block title %}登录{% endblock %}

{% block css %}

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

{% 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>

<div class="form-group">

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

{{ login\_form.username}}

</div>

<div class="form-group">

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

{{ login\_form.password }}

</div>

<div class="form-group">

{{ login\_form.captcha.errors }}

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

{{ login\_form.captcha }}

</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>

</div>

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

{% endblock %}

[root@master templates]# cat register.html

{% extends 'base.html' %}

{% block title %}注册{% endblock %}

{% block content %}

<div class="container">

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

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

{% if message %}

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

{% endif %}

{% csrf\_token %}

<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">

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

{{ register\_form.sex }}

</div>

<div class="form-group">

{{ register\_form.captcha.errors }}

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

{{ register\_form.captcha }}

</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>

</div>

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

{% endblock %}

现在开始对login这个app里面的模块进行修改

我们先写一个form表单的文件

[root@master login]# vim forms.py

from django import forms

from captcha.fields import CaptchaField

class UserForm(forms.Form):

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

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

captcha = CaptchaField(label='验证码')

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')

然后我们修改一下admin.py这个文件

IMG_256

在这个文件中，加入一下代码：

from . import models

admin.site.register(models.User)

然后修改\_\_init\_\_.py文件，在里面加入以下代码：

import pymysql

pymysql.install\_as\_MySQLdb()

此时我们对models.py文件进行修改，完成后代码如下：

[root@master login]# cat models.py

from django.db import models

# Create your models here.

from django import forms

from captcha.fields import CaptchaField

import hashlib

# Create your models here.

class User(models.Model):

'''用户表'''

gender = (

('male','男'),

('female','女'),

)

#name必填，最长不超过128个字符，并且唯一，也就是不能有相同姓名

#注意：这里的用户名指的是网络上注册的用户名，不要等同于现实中的真实姓名，所以采用了唯一机制。如果是现实中可以重复的人名，那肯定是不能设置unique的

name = models.CharField(max\_length=128,unique=True)

#password必填，最长不超过256个字符,实际可能不需要这么

password = models.CharField(max\_length=256)

#email使用Django内置的邮箱类型，并且唯一

email = models.EmailField(unique=True)

#性别使用了一个choice，只能选择男或者女，默认为男

sex = models.CharField(max\_length=32,choices=gender,default='男')

c\_time = models.DateTimeField(auto\_now\_add=True)

#使用\_\_str\_\_帮助人性化显示对象信息

def \_\_str\_\_(self):

return self.name

#元数据里定义用户按创建时间的反序排列，也就是最近的最先显示

class Meta:

ordering = ['c\_time']

verbose\_name = '用户'

verbose\_name\_plural= '用户'

接着我们可以修改视图views.py文件

完成后，代码如下：

#from django.shortcuts import render

# Create your views here.

from django.shortcuts import render,redirect

from . import models

from .forms import UserForm,RegisterForm

import hashlib

def index(request):

pass

return render(request,'index.html')

def login(request):

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

return redirect("/index/")

if request.method == "POST":

login\_form = 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 == hash\_code(password): # 哈希值和数据库内的值进行比对

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.html', locals())

login\_form = UserForm()

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

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 = 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, '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, '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, 'register.html', locals())

# 当一切都OK的情况下，创建新用户

new\_user = models.User.objects.create()

new\_user.name = username

new\_user.password = hash\_code(password1) # 使用加密密码

new\_user.email = email

new\_user.sex = sex

new\_user.save()

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

register\_form = RegisterForm()

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

def logout(request):

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

return redirect('/index/')

request.session.flush()

return redirect('/index/')

def hash\_code(s, salt='game\_admin'):

h = hashlib.sha256()

s += salt

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

return h.hexdigest()

以上动作完成后，我们可以修改主配置文件了

[root@master game\_admin]# pwd

/wdata/game\_admin/game\_admin

[root@master game\_admin]# vim urls.py

from django.contrib import admin

from django.urls import path

##add model

from django.conf.urls import url

from django.conf.urls import include

from django.contrib import admin

from login import views

urlpatterns = [

path('admin/', admin.site.urls),

path('', views.index),

path('index/', views.index),

path('login/', views.login),

path('register/', views.register),

path('logout/', views.logout),

path('captcha/', include('captcha.urls'))

]

## add urlpatterns

#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),

# url(r'^captcha', include('captcha.urls')) # 验证码

#]

path和url方法都可以实现，具体哪个方法好，还需要研究

主配置文件的修改

[root@master game\_admin]# cat settings.py

import os

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'u\*)$=czbvcj6bi&\_)x4sq6r7i#v5qhro-79(!j%-6\_z)p%cx1n'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = ['192.168.33.131']

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions', #session功能启动

'django.contrib.messages',

'django.contrib.staticfiles',

'login', #登录页面部分app

'captcha', #验证码功能,pip install django-simple-captcha安装

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware', #session功能使用

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

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

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

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'game\_admin.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'game\_admin.wsgi.application'

# Database

# [https://docs.djangoproject.com/en/2.2/ref/settings/#databases](https://docs.djangoproject.com/en/2.2/ref/settings/" \l "databases)

DATABASES = {

'default': {

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

# 数据库名字

'NAME': 'django',

# 账户

'USER': 'root',

# 密码

'PASSWORD': '123456',

# 服务器IP

'HOST': '192.168.33.131',

# 端口

'PROT': '3306',

}

}

# Password validation

# [https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators](https://docs.djangoproject.com/en/2.2/ref/settings/" \l "auth-password-validators)

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# <https://docs.djangoproject.com/en/2.2/topics/i18n/>

LANGUAGE\_CODE = 'zh-hans'

TIME\_ZONE = 'Asia/Shanghai'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# <https://docs.djangoproject.com/en/2.2/howto/static-files/>

STATIC\_URL = '/static/'

##add

STATICFILES\_DIRS = [

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

]

效果图：





密码已经哈希



