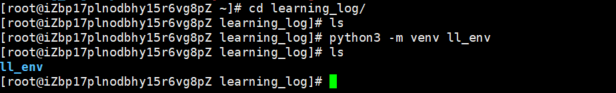
## 从Django入手

# 环境准备

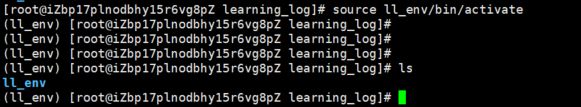
## 创建虚拟环境

python3 -m venv ll\_env



## 激活虚拟环境

source ll\_env/bin/activate



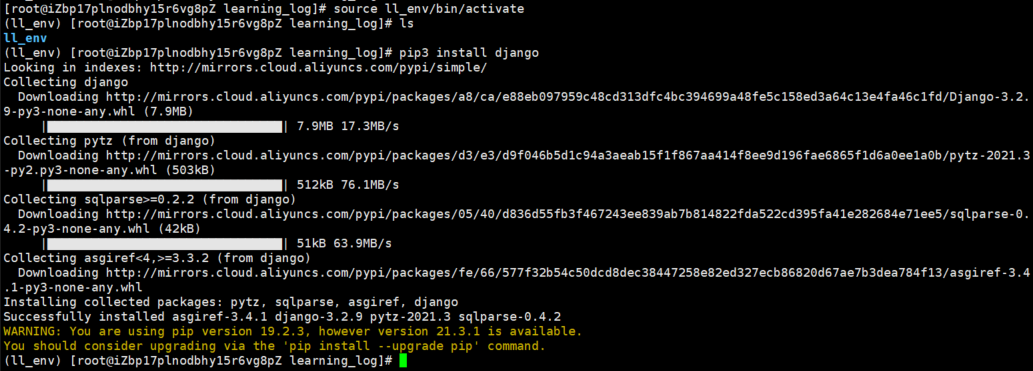
如需要关闭虚拟环境：

deactivate



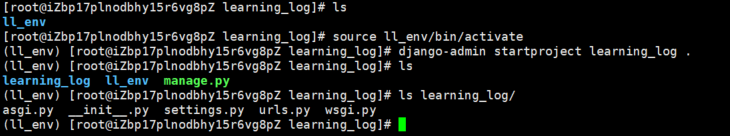
## 安装Django,创建项目

需要在激活虚拟环境后：pip3 install django



在django中创建项目：（需要在激活虚拟环境）

django-admin startproject learning\_log . # 这里有个句号



Settings.py: 指定Django如何与系统交互以及该如何管理项目

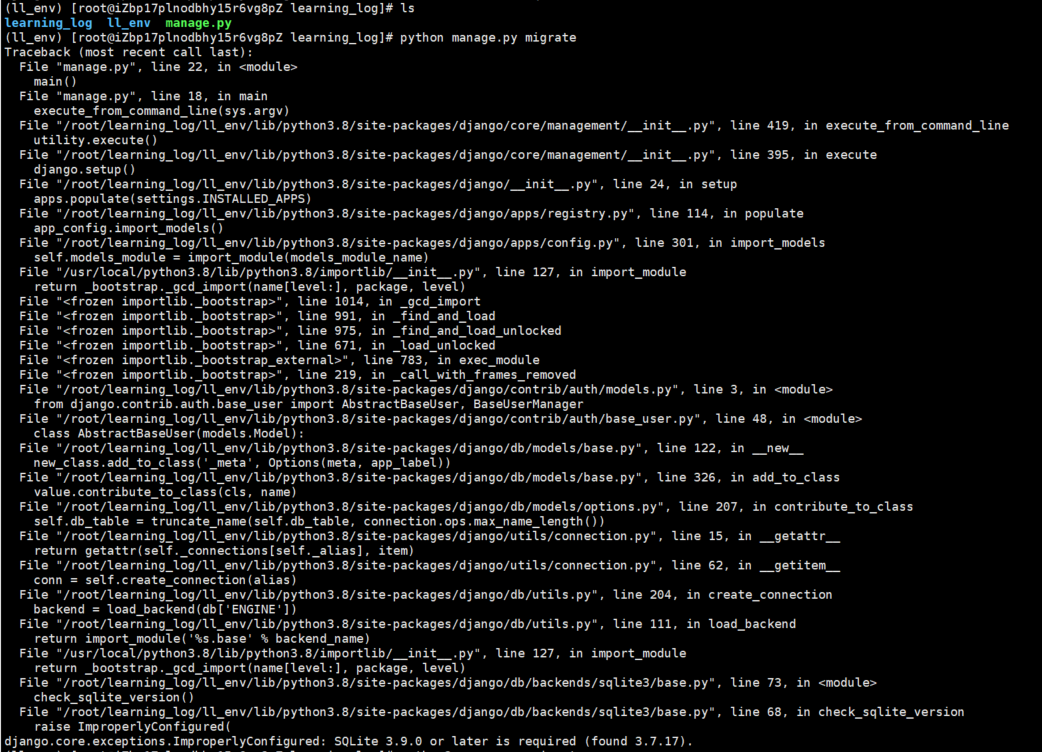
Urls.py: 告诉Django应该创建哪些页面来响应浏览器请求

Wsgi.py: 帮助Django提供它创建的文件，这个文件名是Web服务器网关接口（Web server gateway interface）

## 创建数据库

Django将大部分与项目相关的信息存储在数据库中

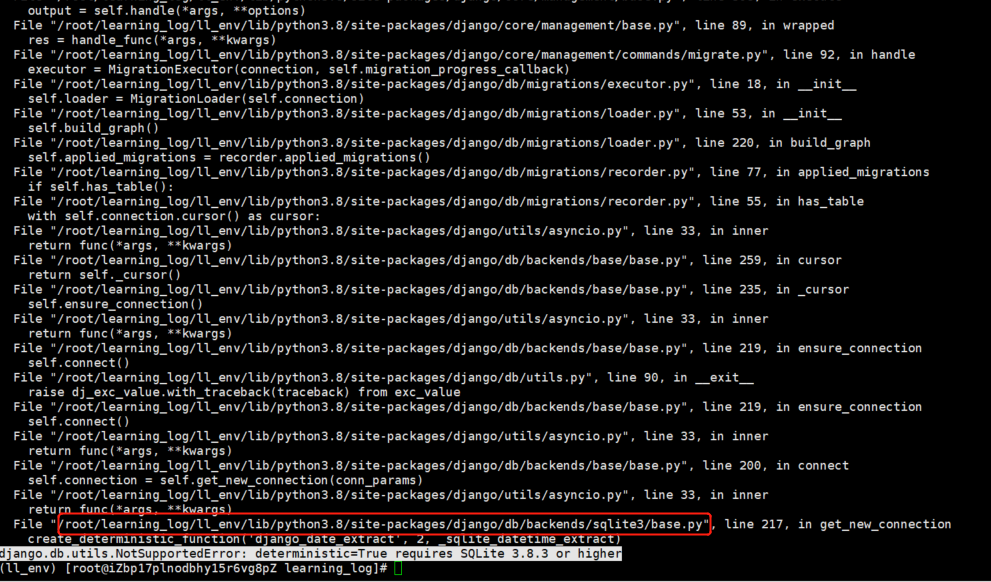
python manage.py migrate （就算安装了python3,这里也用python）



我们发现有安装报错：

django.core.exceptions.ImproperlyConfigured: SQLite 3.9.0 or later is required (found 3.7.17).

解决完上述版本问题后，再执行python manage.py migrate



又发现有报错：

django.db.utils.NotSupportedError: deterministic=True requires SQLite 3.8.3 or higher

解决上述两个报错后，我们发现已经能成功安装数据库了

## 解决创建数据库的两个报错

### django.core.exceptions.ImproperlyConfigured: SQLite 3.9.0 or later is required (found 3.7.17).

参考：<https://www.jianshu.com/p/7dfe62ec1e54>

1. 查看sqlite3版本

sqlite3 –version

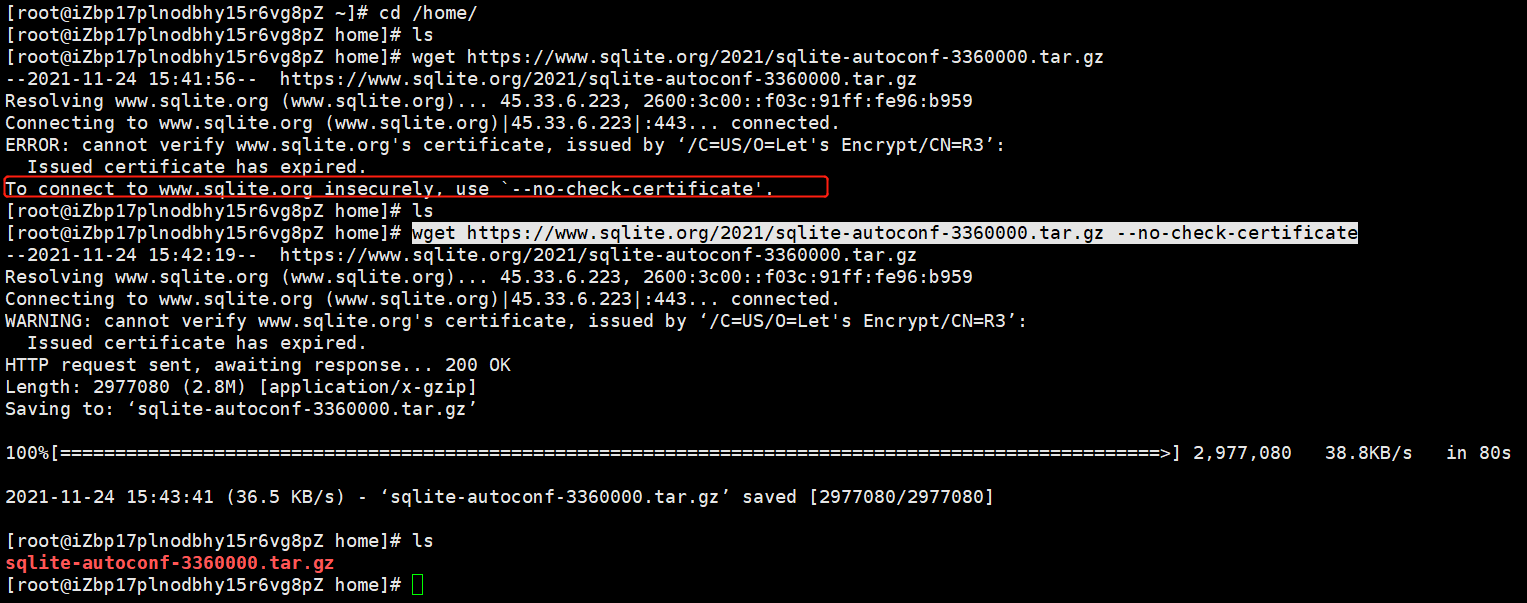


1. 下载高版本的sqlite3安装包



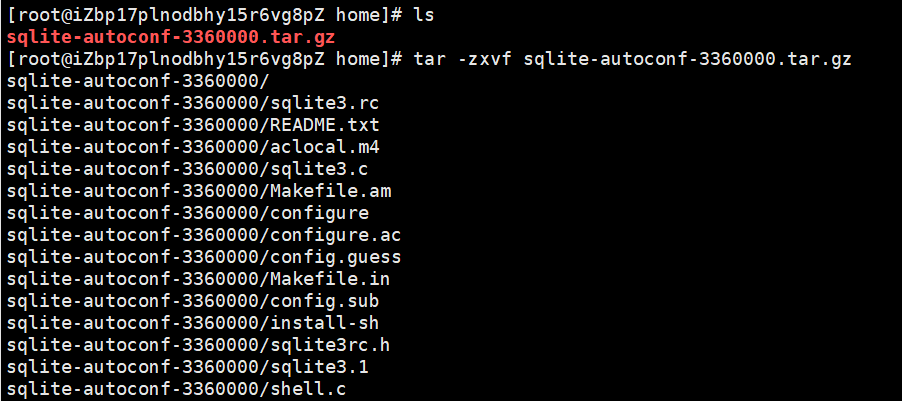
wget <https://www.sqlite.org/2021/sqlite-autoconf-3360000.tar.gz>

wget https://www.sqlite.org/2021/sqlite-autoconf-3360000.tar.gz --no-check-certificate



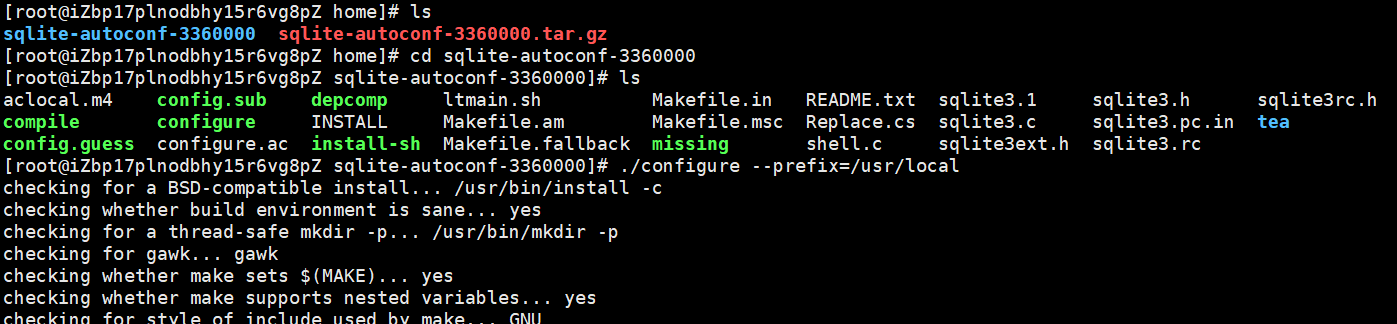
1. 解压

tar -zxvf sqlite-autoconf-3360000.tar.gz

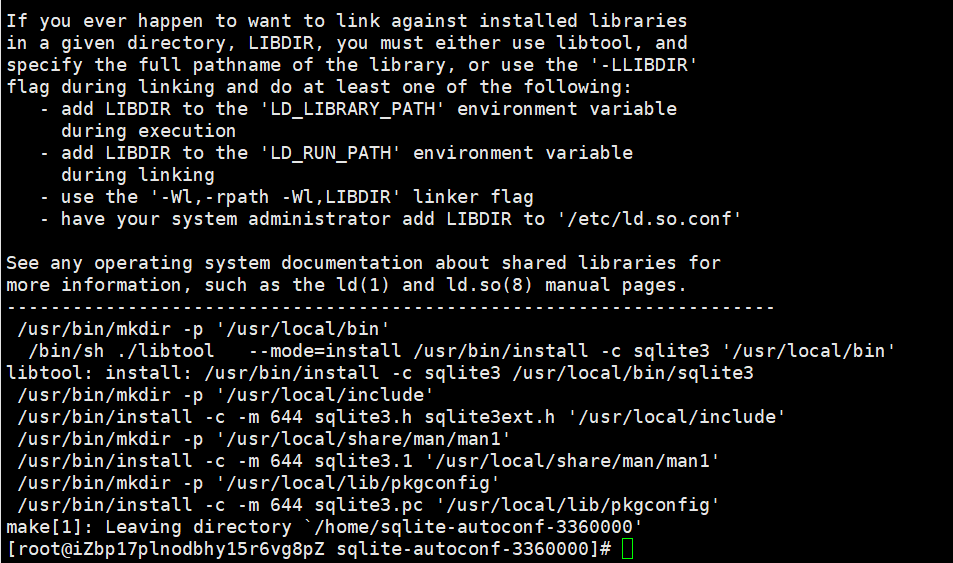


cd sqlite-autoconf-3360000

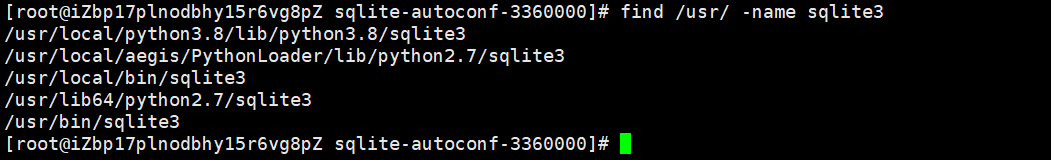
./configure --prefix=/usr/local



make && make install

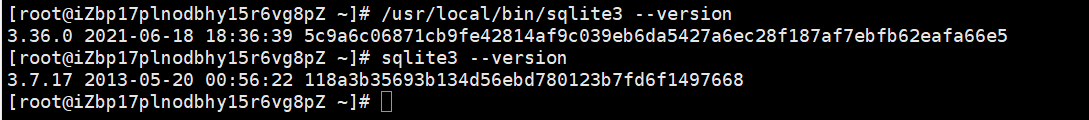


find /usr/ -name sqlite3



1. 检查版本

我们查看新安装：/usr/local/bin/sqlite3 --version



发现到sqlite3的版本还是旧版本，那么需要更新一下

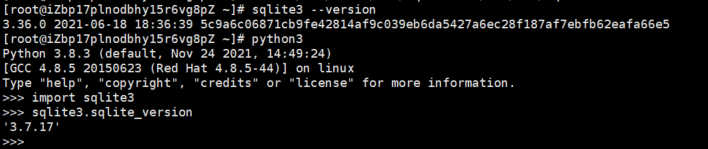
更改旧的sqlite3：

mv /usr/bin/sqlite3 /usr/bin/sqlite3\_old

软链接将新的sqlite3设置到/usr/bin目录下：

ln -s /usr/local/bin/sqlite3 /usr/bin/sqlite3

然后再查看全局sqlite3的版本，此时python3中还是显示之前的版本：

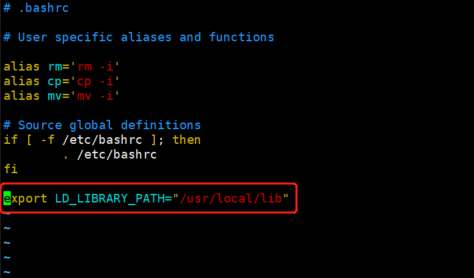


我们需要将路径传递给共享库：

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

设置开机自启动执行，可以将上面的export语句写入 ~/.bashrc 文件中

vim .bashrc



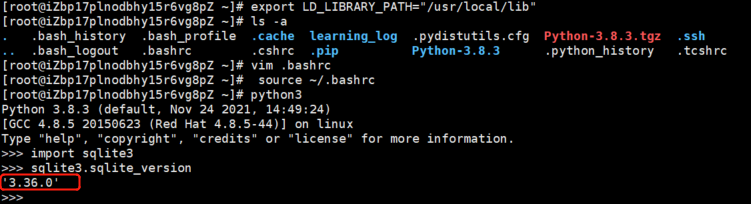
保存：source ~/.bashrc

然后再查看python3中的版本：

python3

import sqlite3

sqlite3.sqlite\_version



至此我们已经完成了sqlite的版本升级

### django.db.utils.NotSupportedError: deterministic=True requires SQLite 3.8.3 or higher

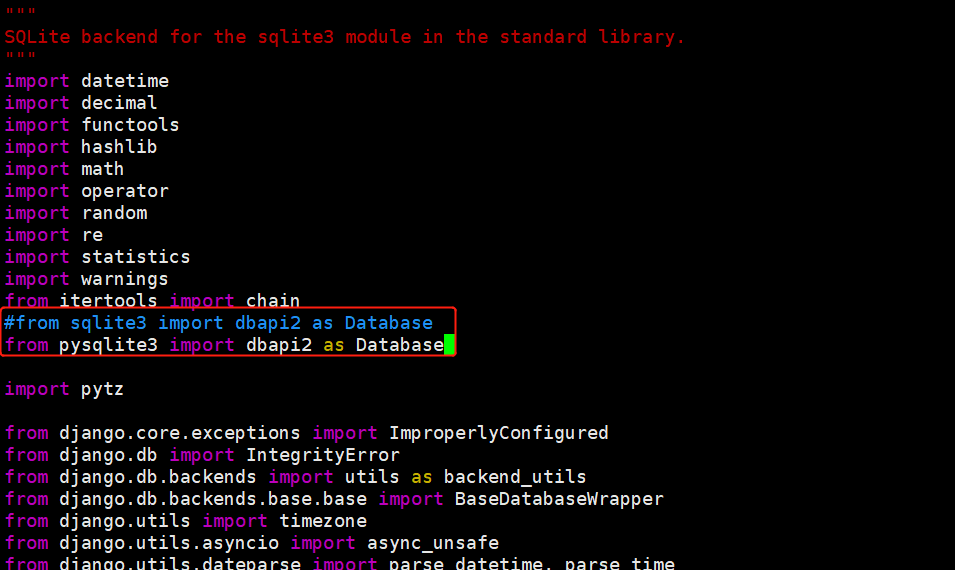
可能是不支持新的版本，最终决定更换sqlite3 为pysqlite3 和 pysqlite3-binary

按上述报错截图中的路径：

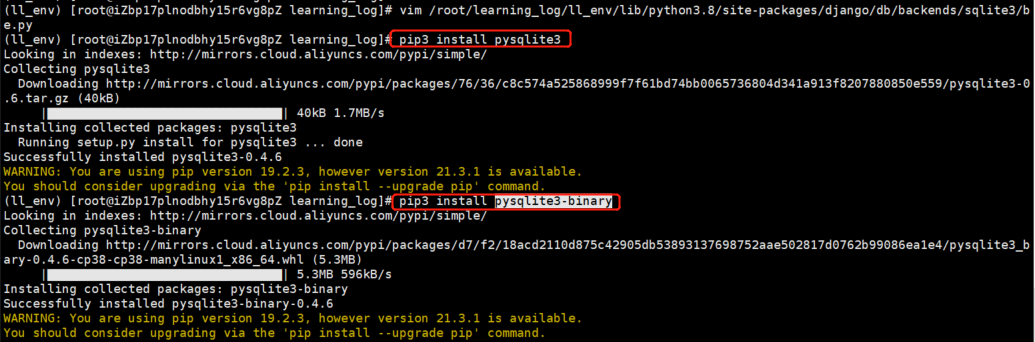
vim /root/learning\_log/ll\_env/lib/python3.8/site-packages/django/db/backends/sqlite3/base.py

# from sqlite3 import dbapi2 as Database # 注释第14 行

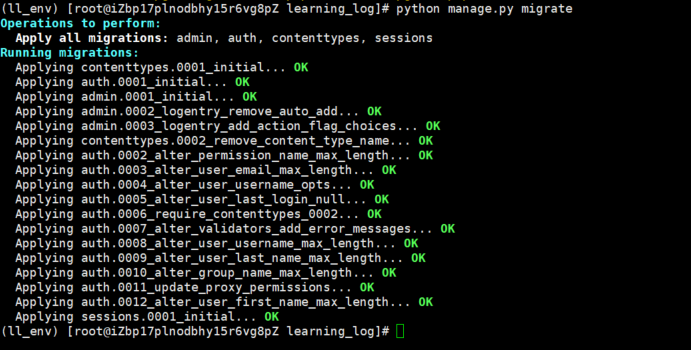
from pysqlite3 import dbapi2 as Database # 将默认数据库获取位置改为 pysqlite3



然后pip3安装pysqlite3，pysqlite3-binary：



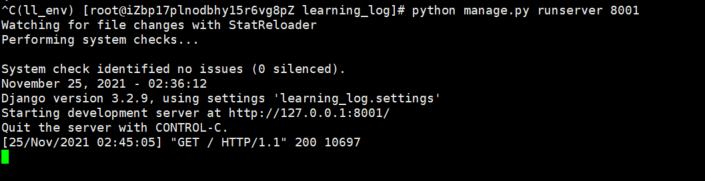
再重新安装数据库：



## 查看项目

下来来核实Django正确的创建了项目，可用命令runserver查看项目的状态：

python manage.py runserver 8001 (这里的命令也用python)



8001是端口

0 silenced表示正确的创建项目

<http://127.0.0.1:8001/>： 表示网址

我们访问url:

curl http://127.0.0.1:8001/

注意：

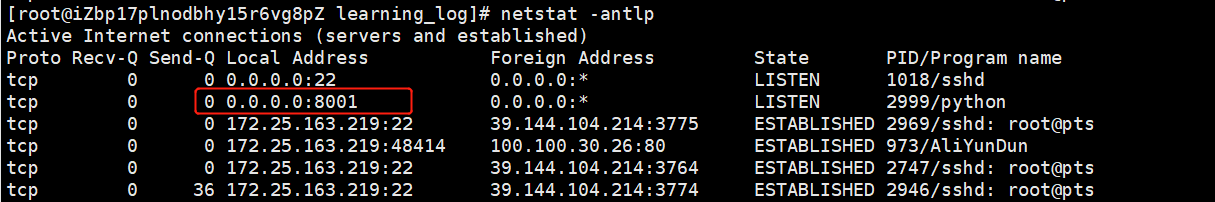
我们发现上述只是本地访问，而我们本地是linux主机，curl命令不方便展示网站内容

我们需要外网访问：

1. 先在阿里云安全组中放行8001端口
2. 我们启服务的时候命令：

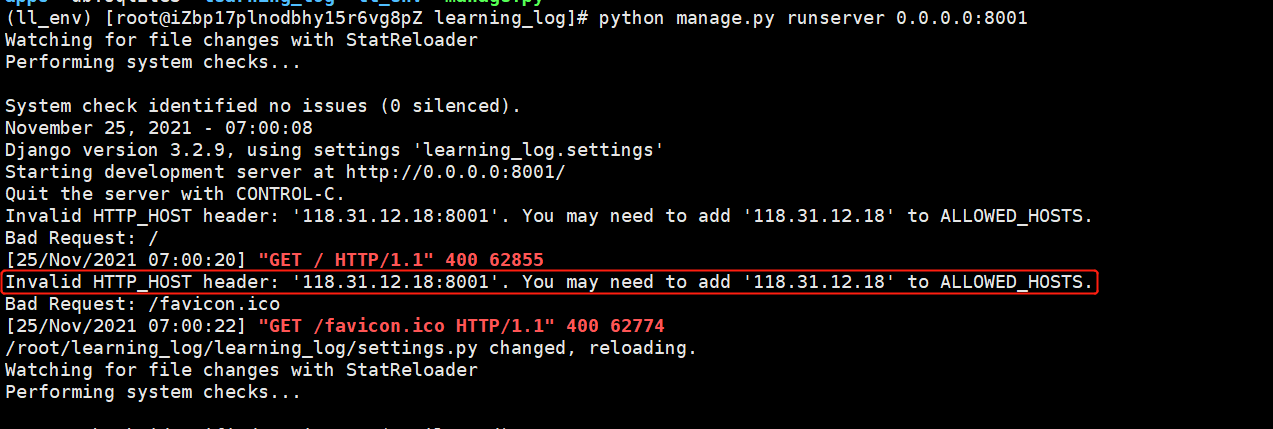
python manage.py runserver 0.0.0.0:8001

同时我们可以看到8001端口已经外网监听了



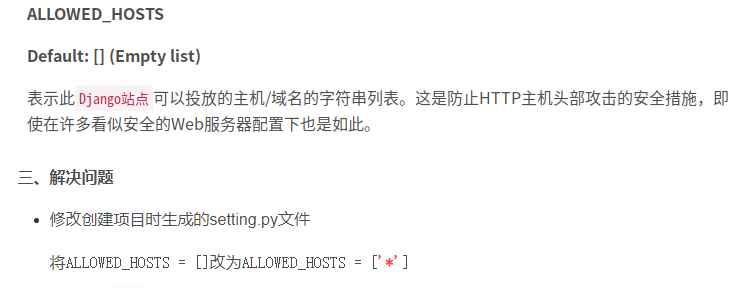
但是我们发现有如下报错：

Invalid HTTP\_HOST header: '118.31.12.18:8001'. You may need to add '118.31.12.18' to ALLOWED\_HOSTS.

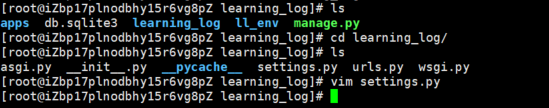


1. 解决上述报错

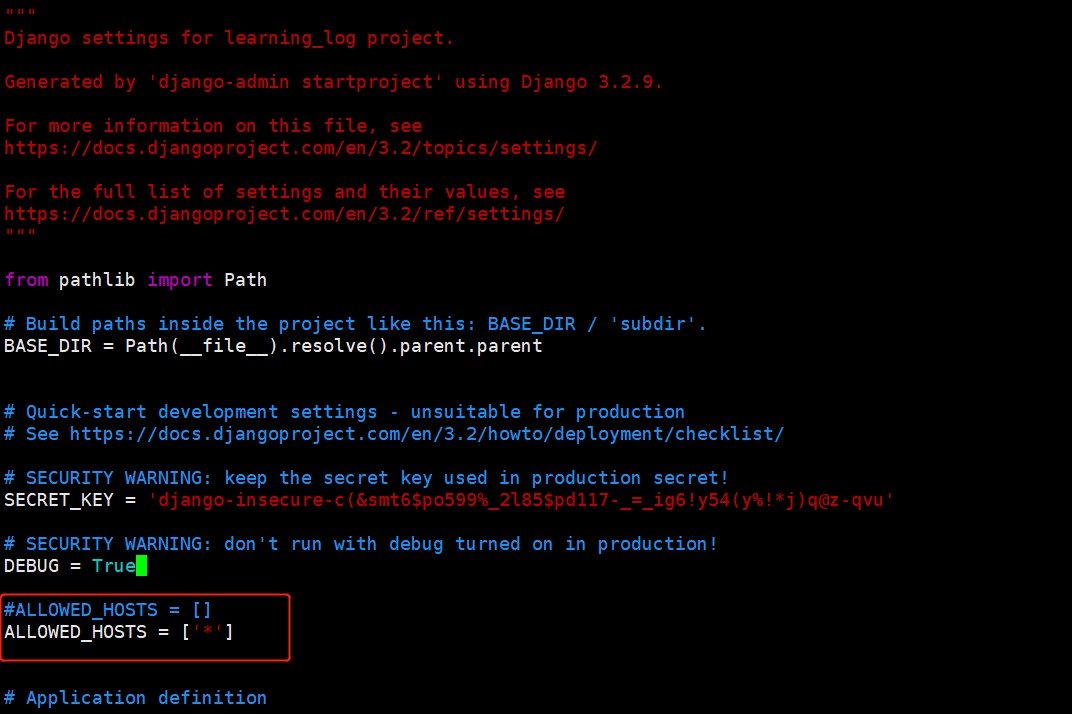
分析：



找到项目下的setting.py:

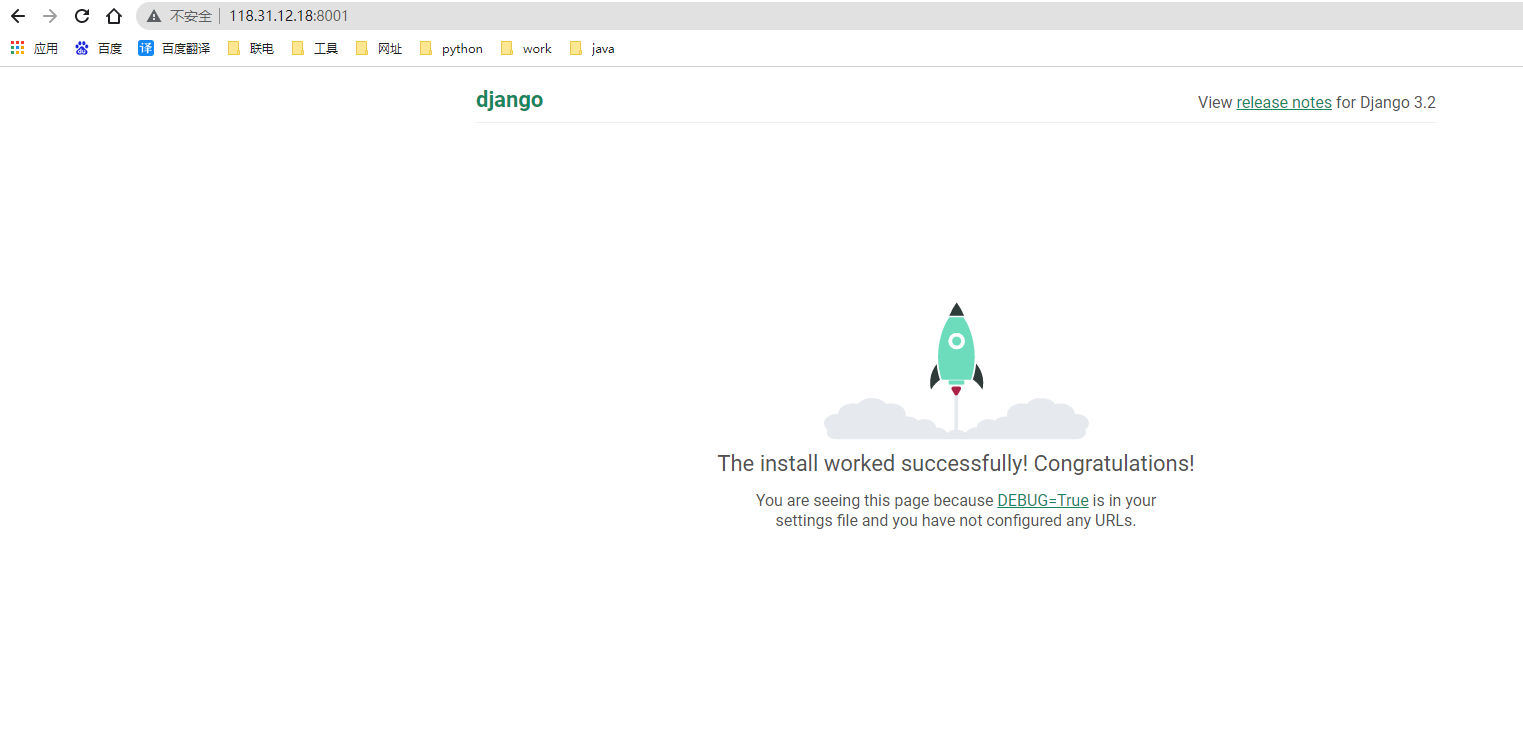


按上述的要求修改：



1. 查看

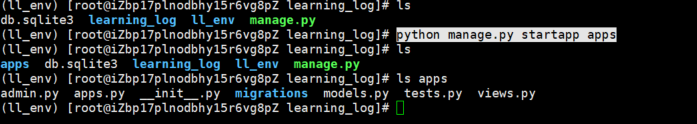
<http://118.31.12.18:8001/>



# 模型管理

## 创建应用程序

python manage.py startapp apps apps是应用程序名

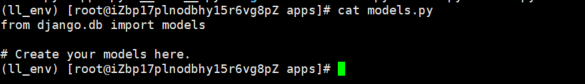


我们先介绍models.py ,我们将使用models.py来定义要在应用程序中管理的数据

## 定义模型Topic（models.py）

我们来想想涉及的数据。每位用户都需要在学习笔记中创建很多的主题；用户输入的每个条目都与特定主题相关联，这些条目都将以文本的方式显示；我们还需要存储每个条目的时间戳，以便告诉用户各个条目都是什么时候创建的

打开models.py,看看它当前包含哪些内容：



编辑models.py:



## 激活模型

要使用该模型，必须让django 将前述应用程序包含到项目中。

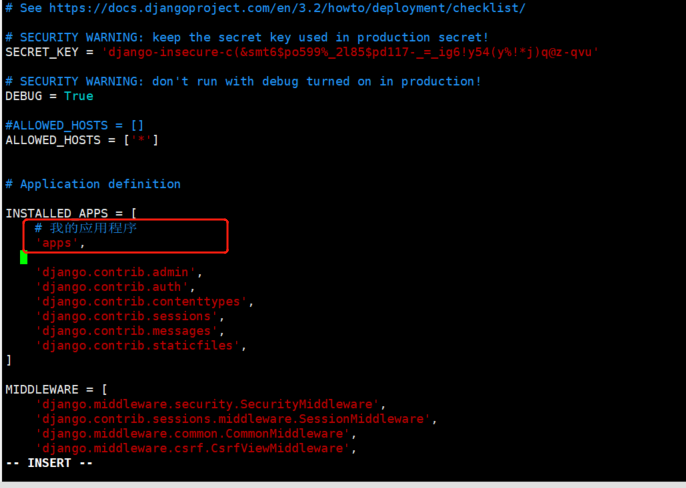
### 打开settings.py

(/root/learning\_log/learning\_log)



红框中是告知django哪些应用程序被安装到项目中并协同工作，我们需要修改INSTALLED\_APPS如下，放在默认的前面：

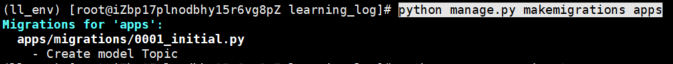
apps是我们创新的应用程序：



### django修改数据库

使其能够存储与模型topic相关的信息：

python manage.py makemigrations apps

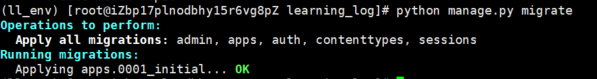


命令makemigrations让django确定该如何修改数据库，输出表明django创建了一个0001\_initial.py的迁移文件，这个文件将在数据库中为模型topic创建一个表。

### 应用迁移，修改数据库

我们需要应用这种迁移，让django替我们修改数据库

python manage.py migrate



总结：每当需要修改“学习笔记”管理的数据时，都采取如下3个步骤： 修改models.py,对apps调用makemigrations，以及让django迁移项目

# django管理网站

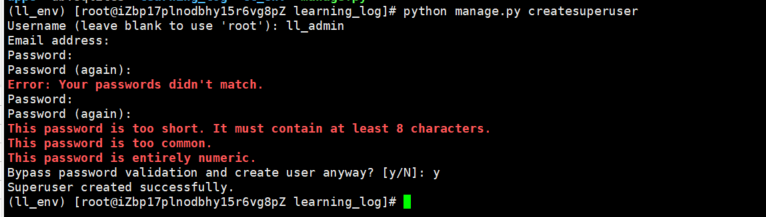
django提供的管理网站（admin site）让你能轻松处理模型。网站管理员可以使用管理网站

。通过使用模型Topic来添加一些主题

## 创建超级用户

为django创建超级用户：

python manage.py createsuperuser



账号： ll\_admin

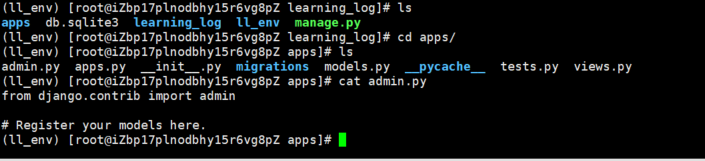
邮箱可以不填

密码： 123

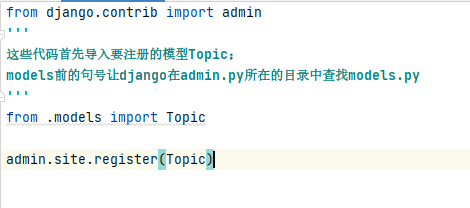
## 向管理网站注册模型（admin.py）

Django自动在管理网站中添加了一些模型，如user和group,但对于我们创建的模型，必须手工注册

我们在创建apps时，django在models.py所在目录创建了一个名为admin.py的文件：

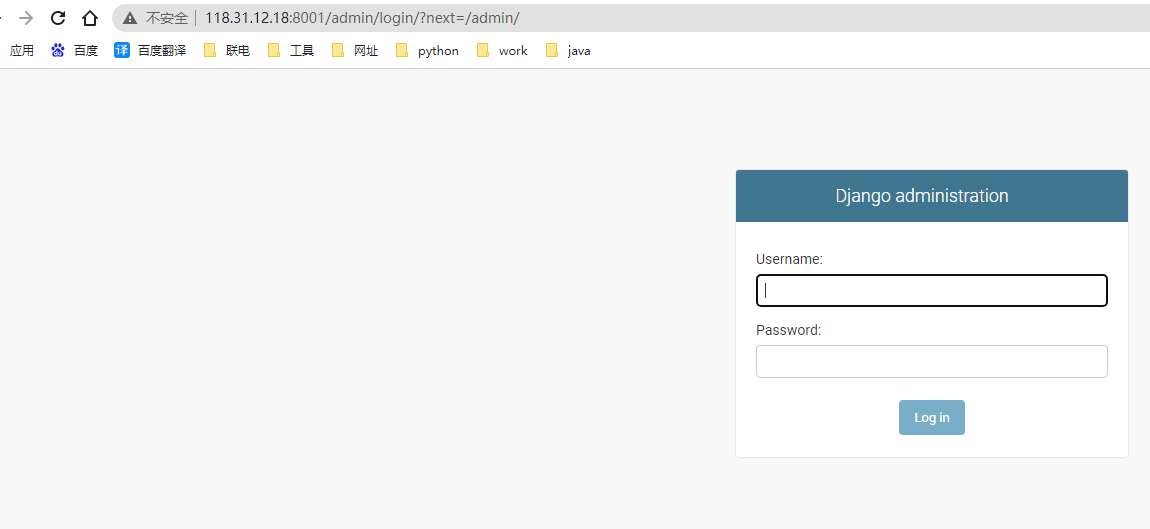


修改admin.py

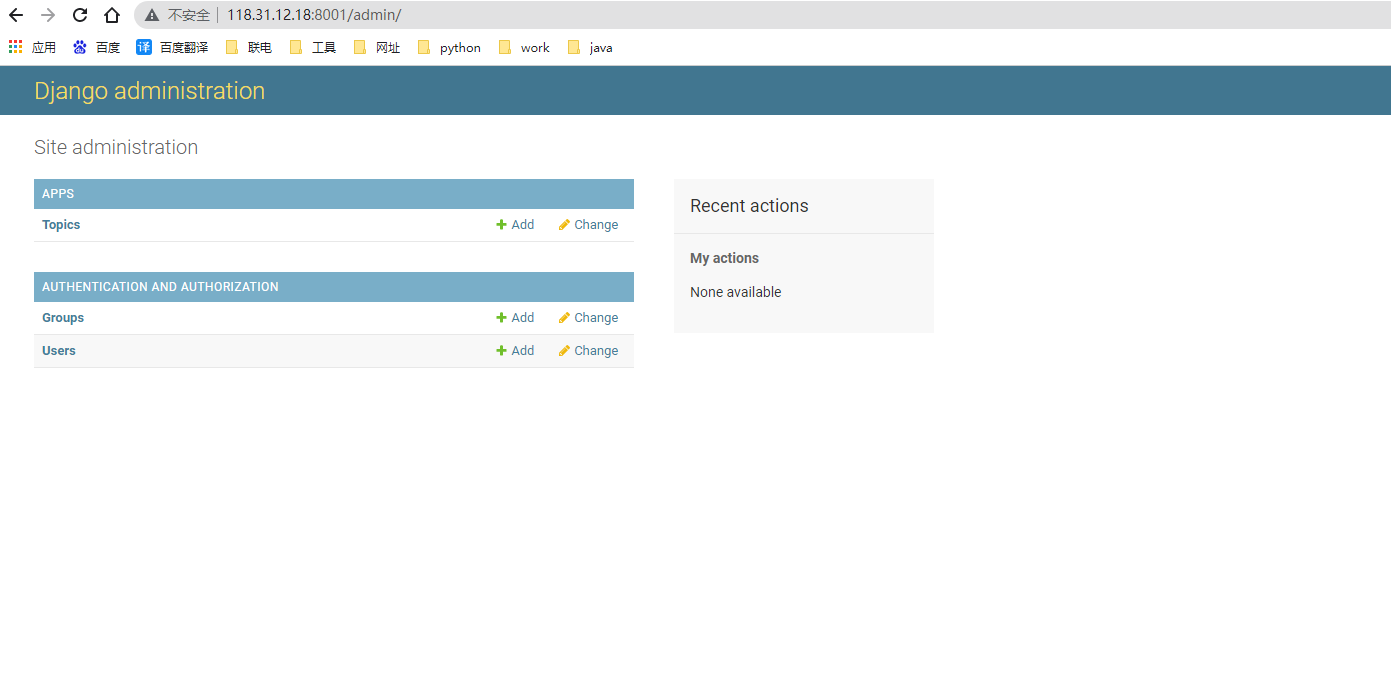


## 访问管理网站

<http://118.31.12.18:8001/admin/>



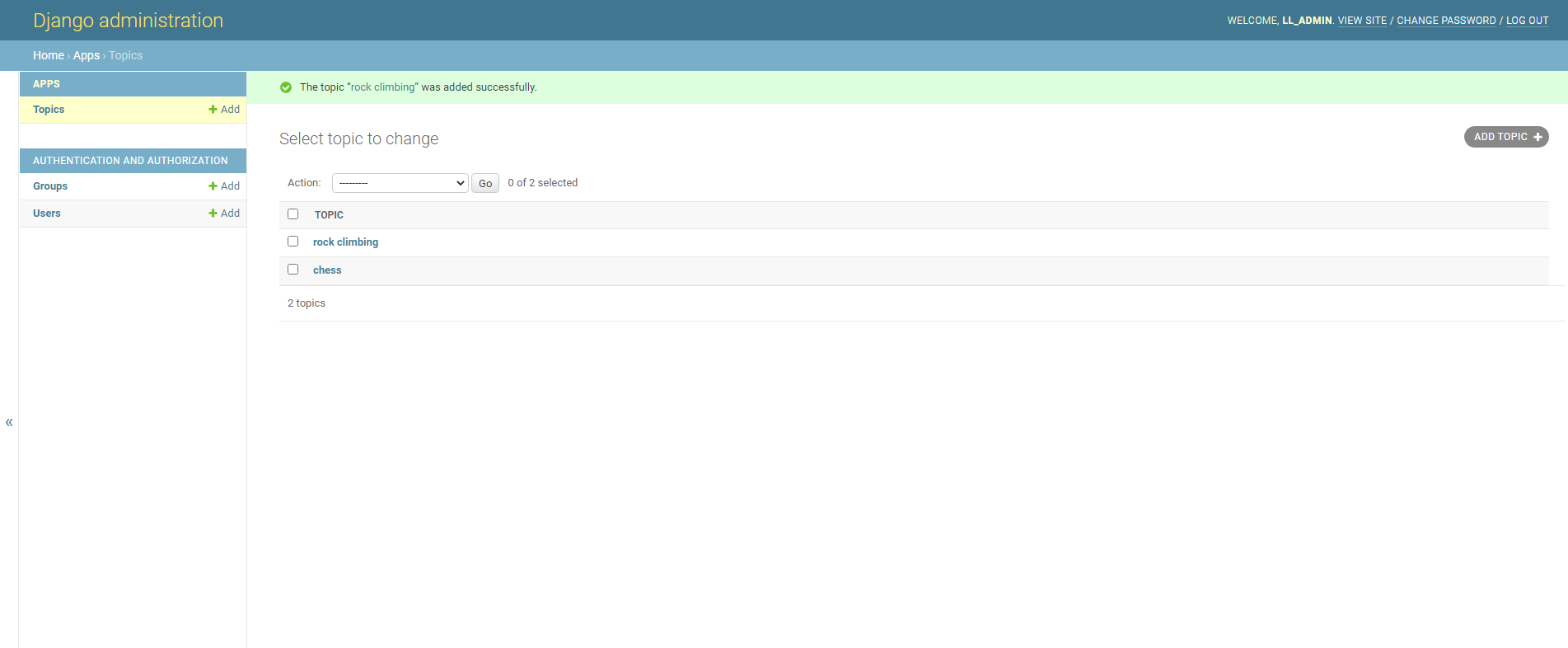
我们用超级用户登录：



## 添加主题

向管理网站注册topic后，我们来添加第一个主题：

Apps/+add/Text 添加两个主题chess,rock



目前我们发现仅能添加topic，我们还需要为新建的topic添加可以详细描述的框框

## 定义模型Entry（models.py）

用户还必须由添加条目的地方。为此，需要定义相关的模型。每个条目都与特定主题相关联，这种关系称为多对一关系，即多个条目可关联到同一主题

这些在models.py里补充：

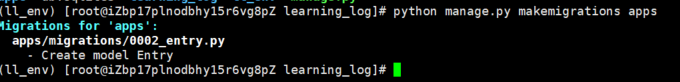


## 激活模型Entry

添加新模型后，需要再次迁移数据库

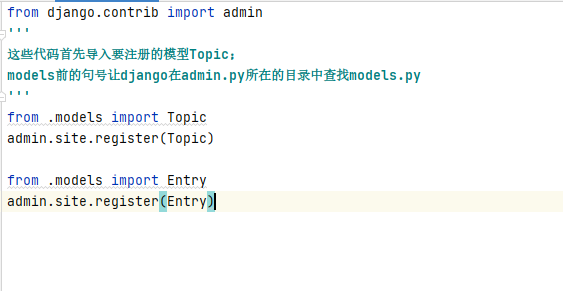
（python manage.py makemigrations app\_name）

python manage.py makemigrations apps

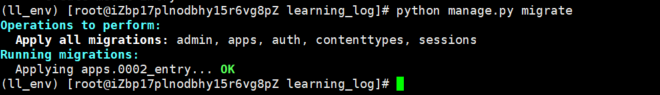


生成了002迁移文件

再次修改admin.py



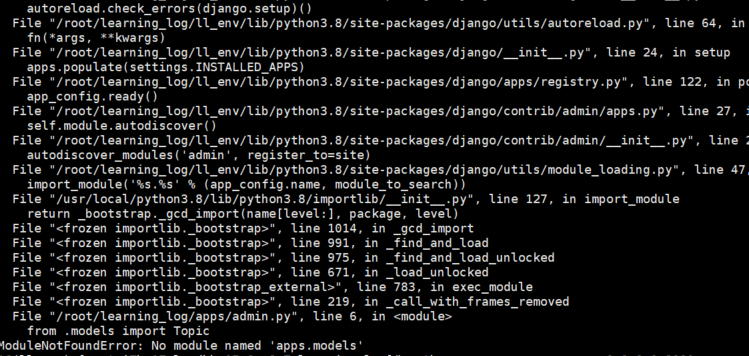
python manage.py migrate



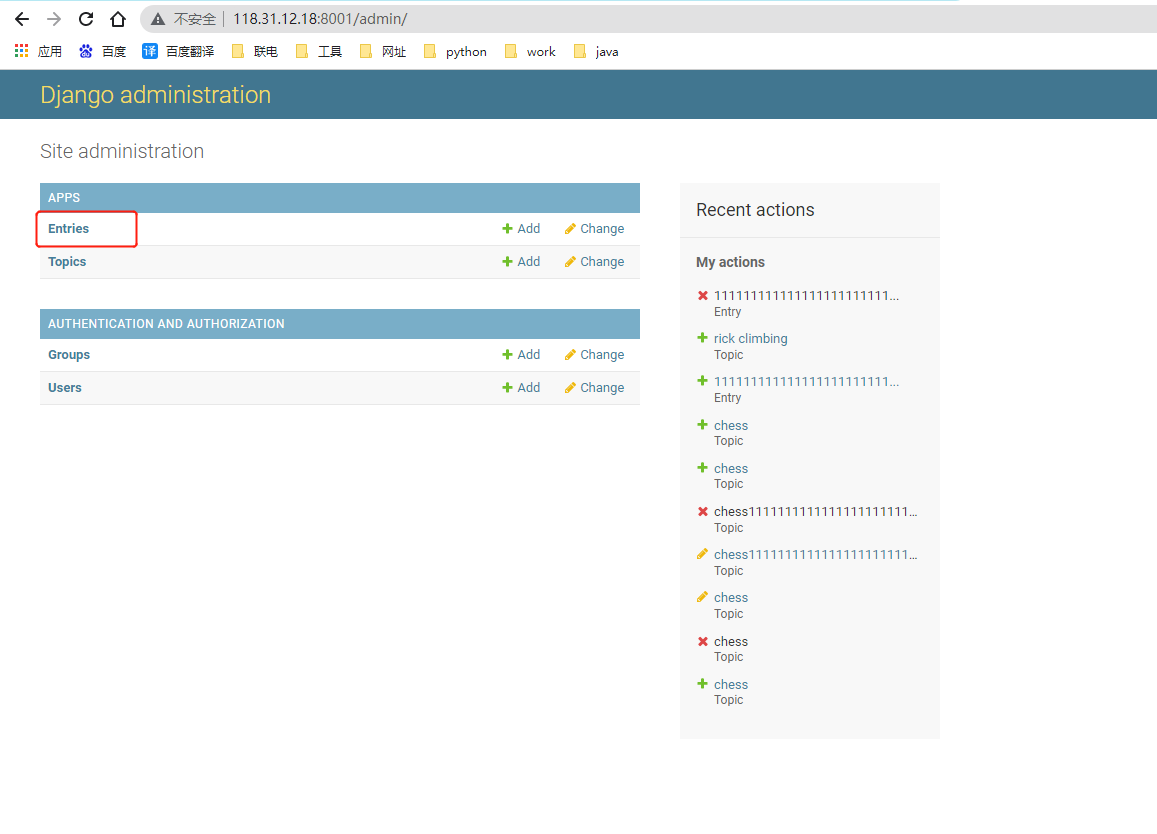
Tips:

如果我们查看runserver界面再重新迁移的出现如下报错，可能是由于models和admin.Py上传顺便导致了，可以重新再启动下：

python manage.py runserver 0.0.0.0:8001



然后可以查看浏览器是否已经增加了Entry模型:



并再Entry模型条目框中添加关于topic的部分内容：



## 回退

这个意思是，比如上述我们新增加了 Entry模型，可能我们突然想回到之前只有一个Topic模型状态

按之前的操作，我们修改models.py：

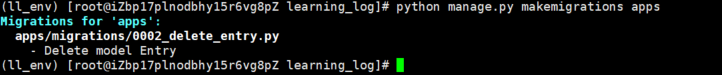


再修改admin.py:



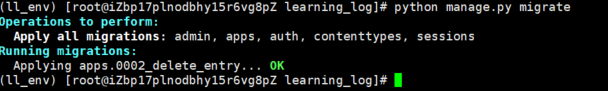
然后需要激活模型：

python manage.py makemigrations apps



最后需要迁移应用文件：

python manage.py migrate



我们发现Entry模块已经消失了

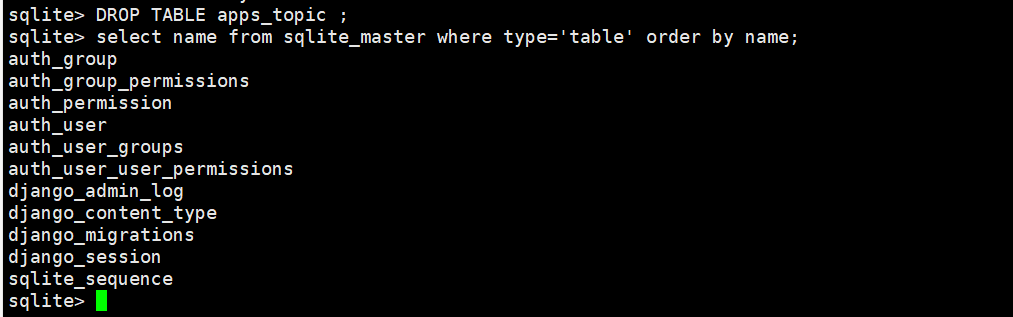
Tips: 可能要关注的地方

python manage.py dbshell # 进入sqlite数据库

select name from sqlite\_master where type='table' order by name; # 查看所有表名

select \* from django\_migrations ;

DROP TABLE apps\_topic ;



## Django shell

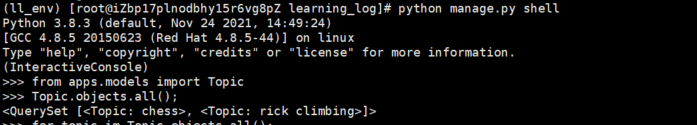
输入一些数据后，就可通过交互式终端会话以编程方式查看这些数据。这种交互式环境称为django shell，是测试项目和排除故障的理想之地

python manage.py shell

from apps.models import Topic 从应用程序的models.py导入Topic模块

Topic.objects.all(); 查看topic下的所有项目

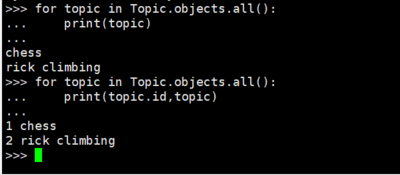
返回查询集（queryset）是一个列表



如果需要遍历输出项目：

for topic in Topic.objects.all():

print(topic.id,topic)



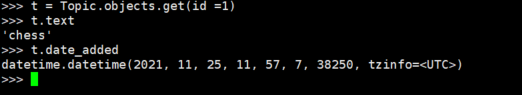
我们能获取主题(chess,rick climbing)的ID，就可用方法Topic.object.get()获取该对象并查看其属性

查看主题属性text和date\_add:

t = Topic.objects.get(id =1)

t.text

t.date\_added



查看与主题相关联的条目，前面给Entry定义了属性topic,这是一个ForeignKey,将条目与主题关联起来，django能获取与主题相关联的所有条目：



要想通过外键关系获取数据，t.entry\_set.all()： t\_相关模型的小写\_set

# 创建页面：学习笔记主页

使用django创建页面的过程分3个阶段： 定义urL,编写视图和编写模板

我们先定义url模式，描述了url是如何设计的，让django知道如何将浏览器请求与网站url匹配，以确认返回哪个页面

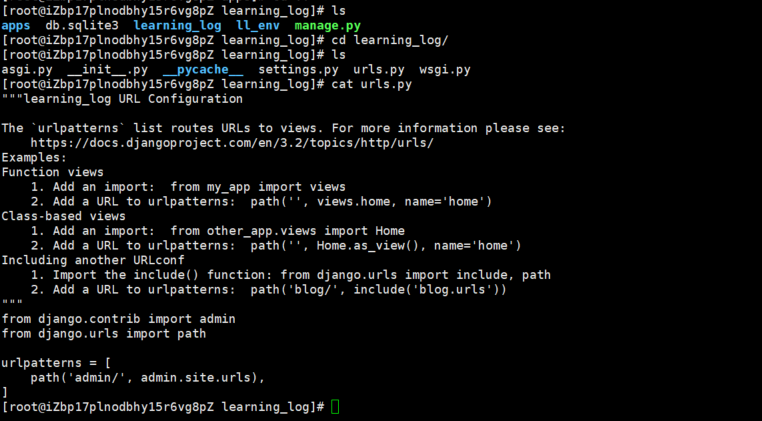
每个url都被映射到特定的视图——视图函数获取并处理页面所需的数据

视图函数通常使用模板来渲染页面，而模板定义页面的总体结构

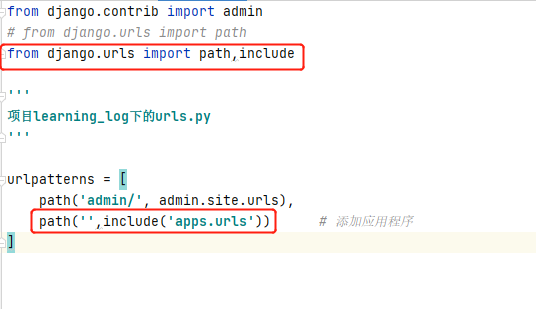
## 映射url（urls.py）

基础的url（http://118.31.12.18:8001/）返回默认的django网站，让我们知道如何正确的建立了项目。下面修改这一点，将这个基础url映射到“学习笔记”的主页

打开项目主文件夹learning\_log中的文件urls.py

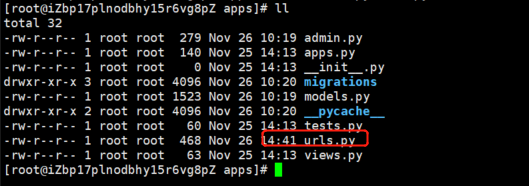


我们需要包含应用程序apps的URL，需要在urls.py添加部分代码：



默认的urls.py包含在项目中，现需要在应用程序apps里创建一个urls.py，内容如下：

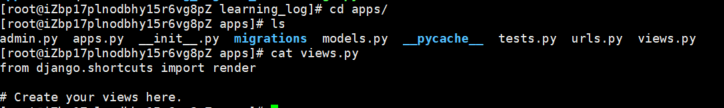




## 编写视图（views.py）

视图函数接受请求中的信息，准备好生成页面所需的数据，再将这些数据发送给浏览器

应用程序apps下views.py是执行命令python manage.py startapp时自动生成的：



修改views.py:



## 编写模板（index.html）

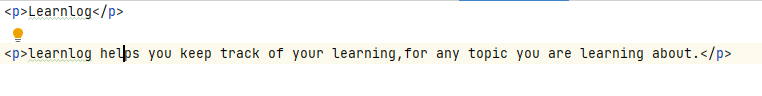
在apps中新建一个文件夹，命名为templates;

Templates中，再建立一个文件夹，命名apps(这个名称要和应用程序名相同);

apps中新建文件index.html (这个文件路径：

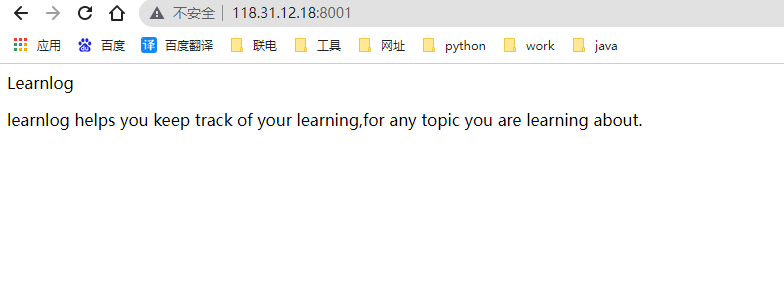
/root/learning\_log/apps/templates/apps/index.html)

编辑index.html:



标签<p></p>标识段落

现在我们请求这个项目的基础的url： <http://118.31.12.18:8001/>



注意： 如果出现django.template.exceptions.TemplateDoesNotExist: apps/index.html

我们遇到的原因是，再apps下创建文件夹templates,写成了templaters

# 创建其他页面

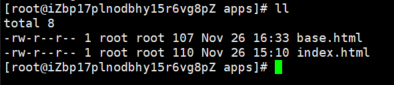
我们可以扩容“学习笔记”项目了，我们将创建两个显示数据的页面，其中一个列出所有的主题，另一个显示特殊主题的所有条目

对于每个页面我们都将指定URL模式，编写一个视图函数并编写一个模板，但在这么做之前，我们先创建一个父模板，项目中的其他模板都继承它

## 模板继承

### 父模板

下面创建一个名为（/root/learning\_log/apps/templates/apps）base.html的模板，并将其存储在index.html所在的目录中



这个模板包含所有页面都有的元素，而其他模板都继承它。

当前所有页面都包含的元素只有顶端的标题，应为每个页面都包含这个模板，所有将这个标题设置为到主页的链接：



为创造链接，使用了一个模板标签，它是用花括号和百分号（{% % }）表示的

这里的模板标签 {% url 'apps:index' %}，生成一个url,该url与在 apps/urls.py 中定义为‘index’（newurls.py中指定 *name = 'index'：将这个URL模式的名称指定为index*）的url模式匹配

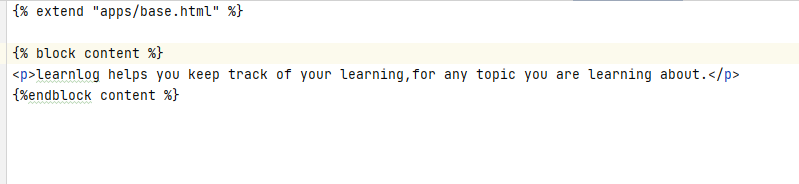
在本例中，apps是一个命名空间，而index是该命名空间中一个名称独特的URL模式

锚标签<a herf=”link\_url”>link text</a>，

块标签 ，块名为content,是一个占位符，其中包含的信息由子模版指定

### 子模板

现在需要重写index.html,使其继承base.html :



我们发现，标题 Learnlog没有了；插入了一个名为 content的{% block %}标签,以定义content块

## 2.显示所有主题的页面

### URL模式

首先定义显示所有主题的页面的URL。

我们通常使用一个简单的URL来指出页面显示的信息，这里使用单词topics，因此URL : <http://118.31.12.18:8001/topics> 将返回显示所有主题的页面

下面演示该如何修改apps/urls.py



URL与该模式匹配的请求都交给views.py中的函数topics()处理

还需要下面步骤的配合，暂时界面是报错的

### 视图

函数topics()需要从数据库中获取一些数据，并将其交给模板。需要在views.py中添加代码：



### 模板

显示所有主题的页面的模板接受字典context,以便使用topics()提供的数据；

请创建一个文件，将其命名为topics.html,并存储到index.html所在的目录：

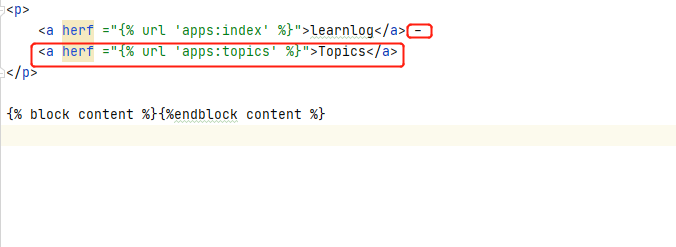


<ul></ul>: 项目列表，包含所有主题的项目列表

{% endfor %} ：表示循环的结束

<li></li>: 项目列表项

现在需要修改父模板，使其包含到显示所有主题的页面链接：base.html:



然后我们访问： <http://118.31.12.18:8001/topics>

