搭建开发环境

创建虚拟环境：

|  |
| --- |
| mkvirtualenv ForDjango |

安装Django库：

|  |
| --- |
| pip install -i https://pypi.doubanio.com/simple/ django |

安装完成之后，输入 import django 没问题则说明安装成功。

进入工作目录，在虚拟环境下，创建Django项目：

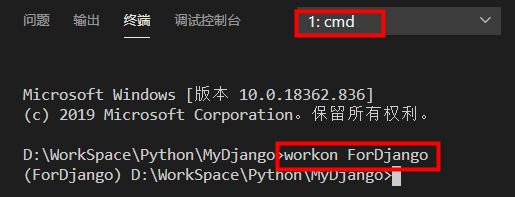
|  |
| --- |
| django-admin startproject MyDjango |

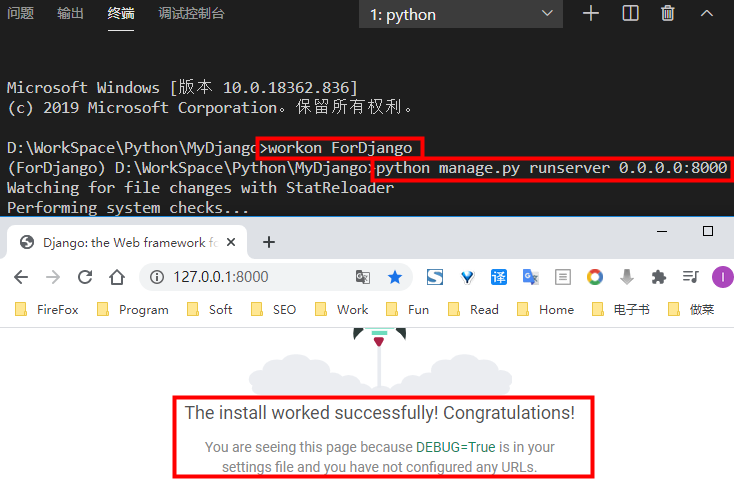
在虚拟环境下，启动 Django 项目：

|  |
| --- |
| python manage.py runserver 0.0.0.0:8000 |

即可启动服务器。然后在浏览器输入 http://127.0.0.1:8000/ 即可访问。

在VSCode中先打开cmd命令行，不能是PowerShell，然后激活虚拟环境，然后再启动Django项目，如下所示：





然后在浏览器输入<http://127.0.0.1:8000/>即可成功访问。

在虚拟环境下，创建 Django APP 应用：

|  |
| --- |
| django-admin startapp XXX |



在Django中，一个Project是由项目配置 + 一个或多个App（应用|模块）组成的。

VSCode搭建Python开发环境的官方文档：

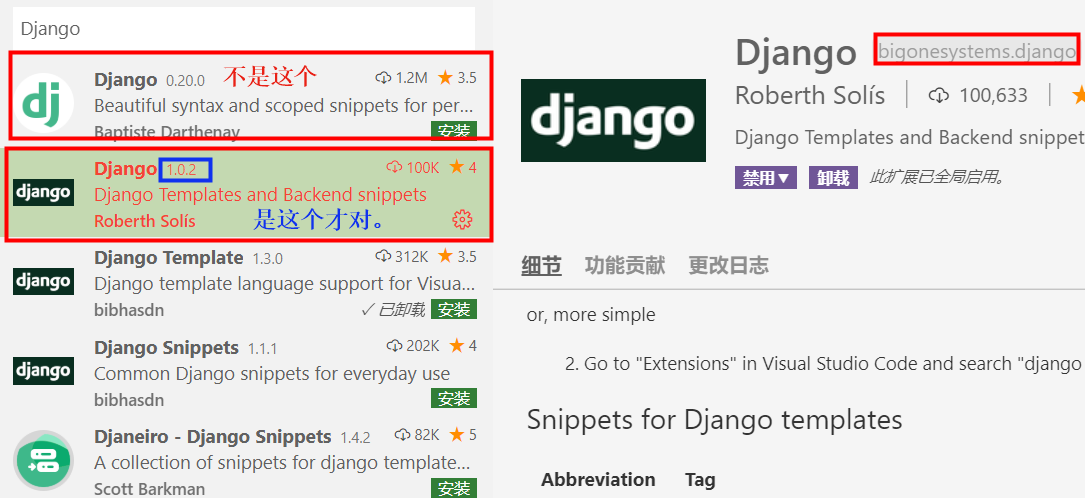
<https://code.visualstudio.com/docs/languages/python>

VSCode安装相关插件：

Python

Django

注意不是搜索出来的第一个，而是后面有个 1.0 版本的，如下图所示：



XML Tools

Visual Studio IntelliCode

在虚拟环境下安装如下Python库：

（检查代码规范）

pip install -i https://pypi.doubanio.com/simple/ flake8

"python.linting.flake8Enable": true,

pip install -i https://pypi.doubanio.com/simple/ pylint

（代码格式化）

pip install -i https://pypi.doubanio.com/simple/ yapf

"python.formatting.provider": "yapf",

pip install -i https://pypi.doubanio.com/simple/ pylint\_django

pip install -i https://pypi.doubanio.com/simple/ MySQL

pip install -i https://pypi.doubanio.com/simple/ django-widget-tweaks

pip install -i https://pypi.doubanio.com/simple/ markdown

markdown扩展名地址：

https://python-markdown.github.io/extensions/

pip install -i https://pypi.doubanio.com/simple/ Pygments

其它样式参考：https://github.com/richleland/pygments-css

开发文档：

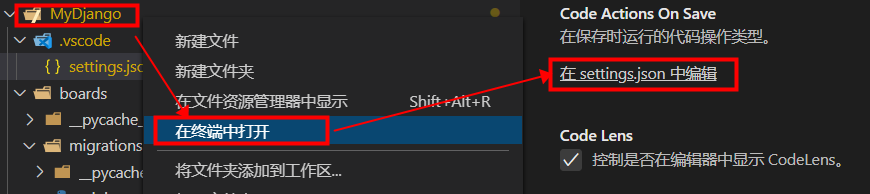
中文文档：

<https://docs.djangoproject.com/zh-hans/3.0/>

英文文档：

<https://docs.djangoproject.com/en/3.0/>

在VSCode中配置Python或者Django的配置文件：

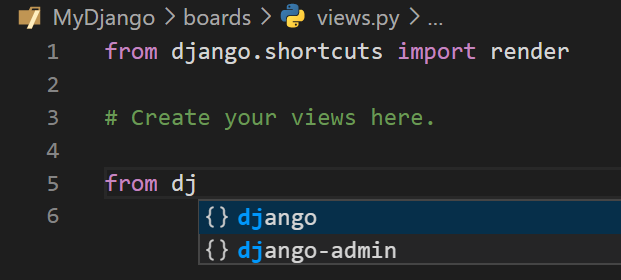


从截图可以看出打开settings.json之后，在项目目录下多出了.vscode/settings.json文件，这个结构是VSCode编辑器给每个项目配置环境时的配置文件结构。

我们在**settings.json**编写如下配置：

|  |
| --- |
| {  "editor.fontSize": 16,  "editor.fontFamily": "Consolas, 'Courier New', monospace",  "python.formatting.provider": "yapf",  "python.linting.pylintEnabled": true,  // 解決 pylint 语法检测不友好的问题  "python.linting.pylintArgs": [  "--load-plugins",  "pylint\_django",  "--disable-msg=['C0103','C0301','W0312','W0511','W0232','E1101']",  ],  "editor.maxTokenizationLineLength": 20000, // 输出自动换行  "editor.wordWrap": "on", // 配置编辑器自动换行  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 必须加 配置开始 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // 必须加  // 必须把项目的目录位置加到 虚拟环境\Lib\site-packages\pythonwork.pth 文件，一行一个  // 必须加，否则无法导入安装的第三方包  // 配置自动，必须按照以下顺序，由内而外的方式  "python.autoComplete.extraPaths": [  "D:/PythonEnv/ForDjango/Lib/site-packages",  "D:/PythonEnv/ForDjango/Scripts",  "D:/PythonEnv/ForDjango/Lib",  "D:/PythonEnv/ForDjango",  ],  // 必须加，配置项目的 Python 虚拟环境  "python.pythonPath": "D:/PythonEnv/ForDjango/Scripts/python.exe",  // 必须加，否则 Code Runner 无法找到我们的虚拟环境  "code-runner.executorMap": {  // 配置 Code Runner 使用的 Python 环境  "python": "D:/PythonEnv/ForDjango/Scripts/python.exe",  },  "code-runner.clearPreviousOutput": true, // 清除之前的输出  "code-runner.saveFileBeforeRun": true, // 运行时保存文件  "code-runner.runInTerminal": true, // 配置在终端运行，无乱码，如果设置为 false 可能乱码  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 必须加 配置结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  } |

这样就有代码提示了，同时也可以查看源码了，如下图所示：



这样写代码就方便多了。另外一个配置文件为**launch.json**，用于项目启动的配置参数，比如启动端口等，不需要则不用配置。另外，如果每次**.html**文件都是以**dj**模板方式打开，则需要在用户配置文件，添加如下代码：

|  |
| --- |
| "files.associations": {  "\*\*/\*.html": "html",  }, |

这样**.html**文件就会以**html**语言模式打开了。所以，CSS和JS也同样可以这么设置。

**自定义包的导入问题：**

在虚拟环境中添加项目所在目录，比如：

|  |
| --- |
| D:\PythonEnv\ForDjango\Lib\site-packages |

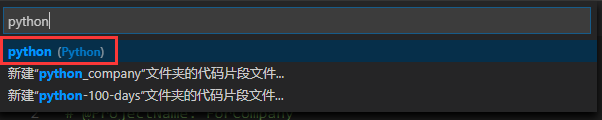
在该目录下增加如下文件：pythonwork.pth

在该文件中增加项目路径，如下所示：

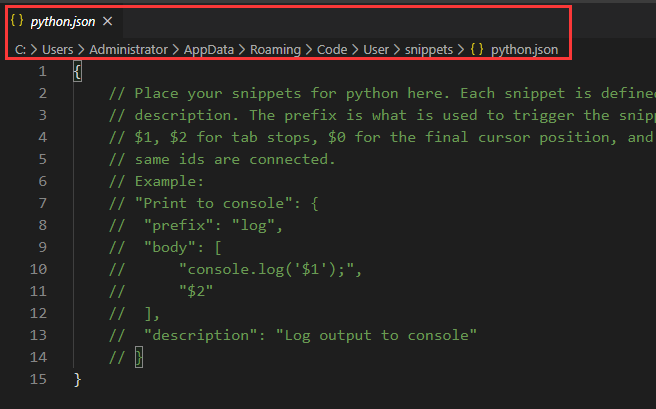
|  |
| --- |
| D:\WorkSpace\for\_actual\_combat\python\_company  D:\WorkSpace\get\_competitor\_keywords |
| 注意：一行一个项目。 |

**添加文件头代码：**

文件 → 首选项 → 用户片段 → 选择 python，如下图所示：



选择 python 后会生成 python.json ，如下 python.json 文件的代码和所在路径：



我们把代码修改成如下：

|  |
| --- |
| {  "HEADER": {  "prefix": "headerpy", // 输入前缀  "body": [  "#!/usr/bin/env python",  "# -\*- encoding: utf-8 -\*-",  "'''",  "@File : $TM\_FILENAME",  "@Time : $CURRENT\_YEAR/$CURRENT\_MONTH/$CURRENT\_DATE $CURRENT\_HOUR:$CURRENT\_MINUTE:$CURRENT\_SECOND",  "@Author : 笑看风云",  "@Contact : 276517382@qq.com",  "@License : (C)Copyright 2019-2020, ShanFeng Group",  "'''",  "",  "$0"  ],  }  } |

在 .py 文件上面输入 headerpy 回车就会自动生成文件头。

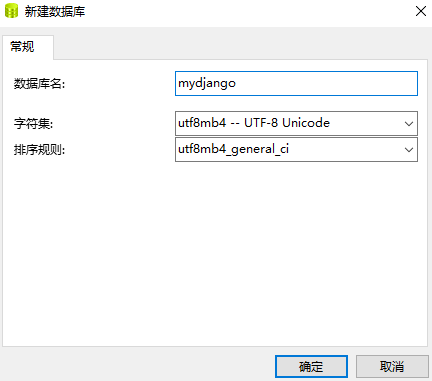
参考地址：

<https://code.visualstudio.com/docs/editor/userdefinedsnippets>

<https://blog.csdn.net/maokelong95/article/details/54379046>

创建和配置数据库：

创建名为mydjango的数据库，如下图所示：



打开settings.py修改如下配置：

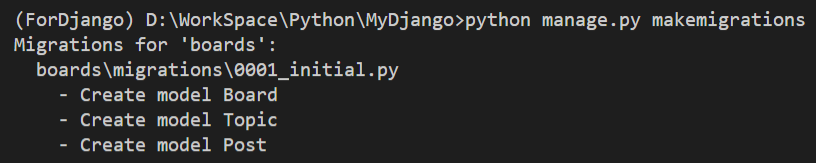
|  |
| --- |
| DATABASES = {  'default': {  # 'ENGINE': 'django.db.backends.sqlite3',  # 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),  'ENGINE': 'django.db.backends.mysql',  'NAME': 'mydjango',  'USER': 'root',  'PASSWORD': 'root',  'HOST': '127.0.0.1',  'PORT': '3306',  }  } |

然后打开boards\models.py，编写如下代码：

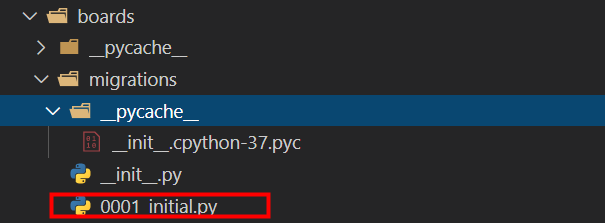
|  |
| --- |
| from django.db import models  # Create your models here.  from django.contrib.auth.models import User  class Board(models.Model):  name = models.CharField(max\_length=30, unique=True)  description = models.CharField(max\_length=100)  class Topic(models.Model):  subject = models.CharField(max\_length=255)  last\_updated = models.DateTimeField(auto\_now\_add=True)  board = models.ForeignKey(Board,  related\_name='topics',  on\_delete=models.CASCADE)  starter = models.ForeignKey(User,  related\_name='topics',  on\_delete=models.CASCADE)  class Post(models.Model):  message = models.TextField(max\_length=1000)  topic = models.ForeignKey(Topic,  related\_name='posts',  on\_delete=models.CASCADE)  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(models.NullBooleanField((True)))  created\_by = models.ForeignKey(User,  related\_name='posts',  on\_delete=models.DO\_NOTHING)  updated\_by = models.ForeignKey(User,  related\_name='+',  on\_delete=models.DO\_NOTHING) |

编写完模型的代码之后，我们进行迁移操作，就是生成迁移文件：

|  |
| --- |
| python manage.py makemigrations |

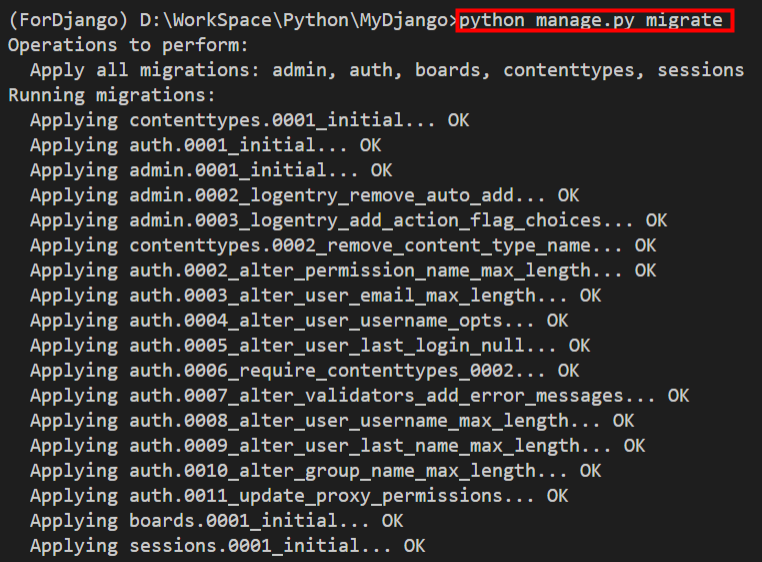


根据以上截图可以知道，生成迁移文件的位置为：

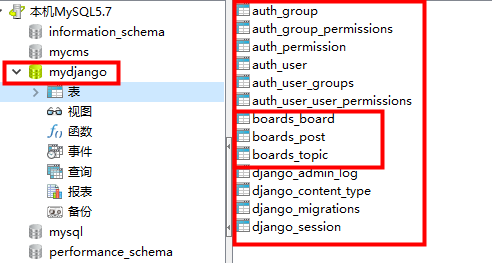


接下来我们执行迁移文件，生成对应的表结构：

|  |
| --- |
| python manage.py migrate |



此时，数据库也生成了对应的表结构，如下图所示：



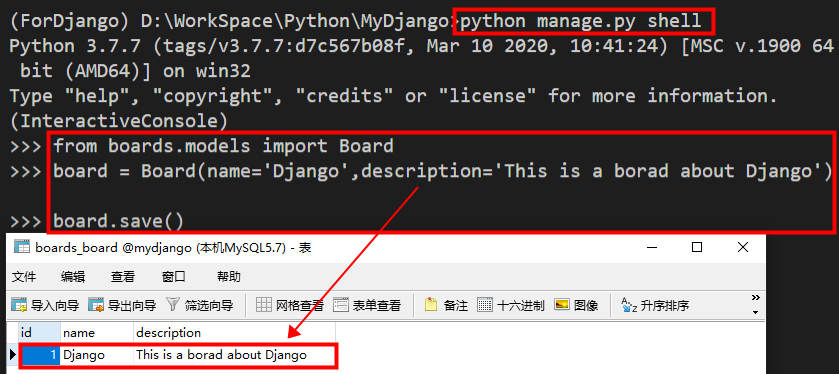
进入Python的Shell交互环境来测试数据：

|  |
| --- |
| python manage.py shell |

代码如下所示：

|  |
| --- |
| >>> from boards.models import Board  >>> board = Board(name='Django',description='This is a borad about Django')  >>> board.save() |

执行过程如下图所示：



注意：每个Django模型都带有一个特殊的属性，这个属性就是模型管理器（Model Manager），我们可以通过属性objects来访问这个管理器，它主要用于数据库操作。

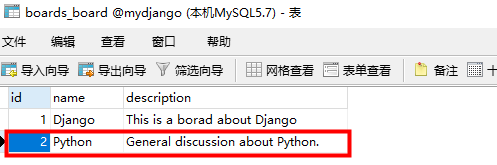
比如：

|  |
| --- |
| board = Board.objects.create(name='Python', description='Gene  ral discussion about Python.') |

过程如下：

|  |
| --- |
| (ForDjango) D:\WorkSpace\Python\MyDjango>python manage.py shell  Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 10:41:24) [MSC v.1900 64 bit (AMD64)] on win32  Type "help", "copyright", "credits" or "license" for more information.  (InteractiveConsole)  >>> from boards.models import Board  >>> board = Board.objects.create(name='Python',description='General discussion about Python.')  >>> board.save()  >>> |

数据库表增加了相应的记录：



这时，我们创建了两个板块，一个是Django，一个是Python。

Django发布时间的时区问题：

打开my\_blog/settings.py修改如下两个字段：

|  |
| --- |
| # 时区相关的两个字段  USE\_TZ = False  TIME\_ZONE = 'Asia/Shanghai' |

参考地址：

<https://www.cnblogs.com/kakawith/archive/2018/12/04/10058921.html>

设置后台管理界面的语言：

打开my\_blog/settings.py修改如下字段：

|  |
| --- |
| # 设置后台管理界面的语言  # LANGUAGE\_CODE = 'en-us'  LANGUAGE\_CODE = 'zh-hans' |