Python 虚拟环境管理

日期：2020/01/16

virtualenv / virtualenvwrapper：venv 管理

虚拟环境是什么？为什么要用虚拟环境？如何创建、激活?

为什么需要虚拟环境?

参考资料

https://www.jb51.net/article/114933.htm

https://www.jianshu.com/p/2fdb53825d35

https://www.jianshu.com/p/e17263be54ea

“python虚拟环境，是一个隔离/独立的python开发环境。和系统python环境可以完全隔离，互不相关，相当于多了一个python开发环境。

而且在python虚拟环境中的开发过程，和使用系统python没有差别。可以在创建的python虚拟环境中，使用pip工具安装任何你需要的模块，该模块和系统python环境完全不相关。“

常用步骤

安装 - 创建 - 激活 - 在虚拟环境中安装包

1. 在虚拟环境中安装项目依赖（必备的一些包）
2. 迁移数据库： python manage.py migrate
3. 创建后台管理员账户： python manage.py createsuperuse
4. 运行开发服务器： python manage.py runserver

virtualenv常用命令

安装虚拟环境

pip install virtualenv

创建虚拟环境

virtualenv [OPTIONS] DEST_DIR

例 1：

// 更换当前目录

$ cd project1

//在当前目录创建虚拟环境

$ virtualenv venv

例 2：

// 在指定目录下创建虚拟环境

$ virtualenv /home/yan/env

激活虚拟环境

$ source venv/bin/activate

关闭虚拟环境

$ deactivate

删除虚拟环境

要删除一个虚拟环境，只需删除它的文件夹

$ rm -rf venv

$ rmvirtualenv my_env

列出所有虚拟环境

$ lsvirtualenv

virtualenvwrapper

virtualenvwrapper安装与使用 | https://u.nu/z35gz

Mac配置Python开发环境之virtualenvwrapper | https://www.jianshu.com/p/83ab5947bc7e

优点

为什么需要 virtualenvwrapper？而不是virtualenv?

virtualenv 的一个最大的缺点就是，每次开启虚拟环境之前要去虚拟环境所在目录下的 bin 目录下 source 一下 activate，这就需要我们记住每个虚拟环境所在的目录。virtualenvwrapper，相较于使用 virtualenv， 好处就是把所有环境都放在同一目录下管理，以便更好的管理及切换。

可行的解决方案是，将所有的虚拟环境目录全都集中起来，比如放到 ~/virtualenvs/，并对不同的虚拟环境使用不同的目录来管理。virtualenvwrapper 正是这样做的。并且，它还省去了每次开启虚拟环境时候的 source 操作，使得虚拟环境更加好用。

virtualendvwrapper 的安装及配置

1. 进入命令行界面安装virtualenvwrapper： pip install virtualenvwrapper
2. 其次，还需要对 virtualenvwrapper进行配置。
3.
    1. 它需要指定一个环境变量，叫做WORKON_HOME，要用来存放各种虚拟环境目录的目录，这里我们可以设置为 ~/.workspaces
    2. 需要运行一下它的初始化工具 virtualenvwrapper.sh，这个脚本在 /usr/local/bin/ 目录下。

操作如下：首先创建两个文件夹，一个用来存放虚拟环境（.virtualenvs），另一个是工作空间（workspace）

$mkdir$HOME/.virtualenvs

$mkdir$HOME/workspace

需要说明的是，文件夹的名称自己可以随意更改，相应的代码就要进行更改。

1. 在shell配置文件中添加如下几行代码 sudo vi ~/.bash_profile
2. 编辑界面 export WORKON_HOME=$HOME/.virtualenvsexportPROJECT_HOME=$HOME/workspace source /usr/local/bin/virtualenvwrapper.sh
3. 然后在当前终端（shell）中执行如下命令让更改之后的终端配置生效 $ source ~/.bash_profile

成功配置后出现如下界面：

```
(base) wangyanfandeMBP:~ wangyangfan$ source /usr/loca
l/bin/virtualenvwrapper.sh
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/premkproject
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/postmkproject
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/initialize
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/premkvirtualenv
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/postmkvirtualenv
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/prermvirtualenv
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/postrmvirtualenv
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/predeactivate
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/postdeactivate
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/preactivate
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/postactivate
virtualenvwrapper.user_scripts creating /Users/wangyan
gfan/workspaces/get_env_details
```

虚拟环境目录的目录workspaces：

/Users/wangyangfan/workspaces

virtualenvwrapper的使用

1 创建虚拟环境

$ mkvirtualenv py37_test

其中py37_test是我创建的虚拟环境的名字，可以自行更改。而且创建完成后自动进入创建好的虚拟环境。

mkvirtualenv也可以创建指定python版本的虚拟环境，此时要用到-p参数指出python的路径：

$ mkvirtualenv -p /usr/local/bin/python3 py37_test

2 查看虚拟环境

$lsvirtualenv$或者 workon

3 启动/切换某虚拟环境

命令格式 workon VIRTUALENV_NAME

其中VIRTUALENV_NAME是虚拟环境的名称。

测试：

1. 为了查看效果，重新打开一个终端，再创建一个新的虚拟环境py27_test

$ mkvirtualenv py27_test

1. 在该界面下使用命令 workon，则会显示出如下类容

$ py27_test

$ py37_test

也就是将创建好的两个虚拟环境都列举了出来。

1. 此时处于py27_test环境，要切换到py37_test环境，则执行命令

$ workon py37_test

4 删除虚拟环境

rmvirtualenv VIRTUALENV_NAME

$ rmvirtualenv py37_test

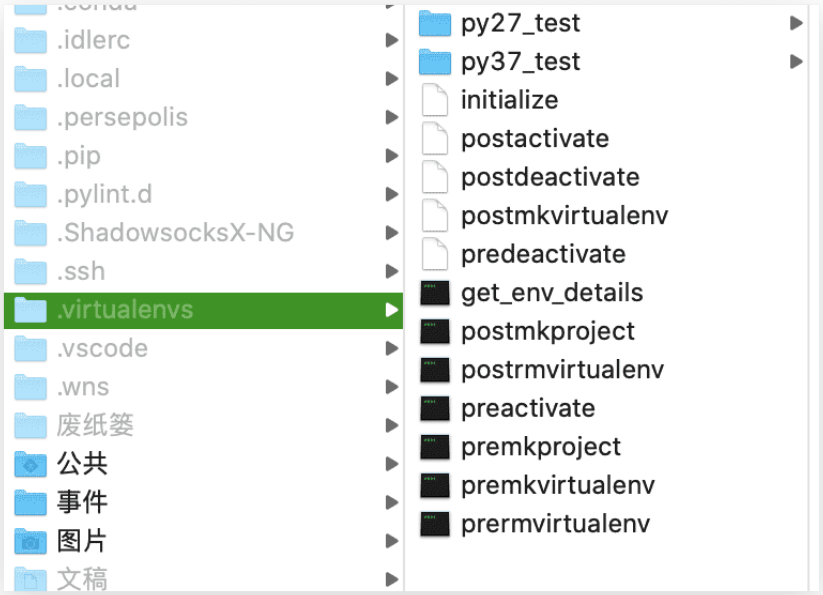
需要说明的是，无法在虚拟环境中删除虚拟环境，要先退出虚拟环境才行。

5 退出虚拟环境 deactivate

在虚拟环境中使用deactivate则退出了虚拟环境，只有就可以删除虚拟环境了。

如下可见，py27_test 和 py37_test 两个虚拟环境文件夹，虚拟环境绝对路径为：

/Users/wangyangfan/.virtualenvs



确保开发环境的一致性

1.假设我们在本地开发环境，准备好了项目+依赖包环境

2.现在需要将项目上传至服务器，上线发布

3.那么就要保证服务器的python环境一致性

解决方案：

1.通过命令保证环境的一致性，导出当前python环境的包

pip3 freeze > requirements.txt

这将会创建一个 requirements.txt 文件，其中包含了当前环境中所有包及 各自的版本的简单列表。

可以使用 “pip list”在不产生requirements文件的情况下， 查看已安装包的列表。

2.上传至服务器后，在服务器下创建virtualenv，在venv中导入项目所需的模块依赖

pip3 install -r requirements.txt