# virtualenv

在开发Python应用程序的时候，系统安装的Python3只有一个版本：3.4。所有第三方的包都会被`pip`安装到Python3的`site-packages`目录下。

如果我们要同时开发多个应用程序，那这些应用程序都会共用一个Python，就是安装在系统的Python 3。如果应用A需要jinja 2.7，而应用B需要jinja 2.6怎么办？

这种情况下，每个应用可能需要各自拥有一套“独立”的Python运行环境。virtualenv就是用来为一个应用创建一套“隔离”的Python运行环境。

首先，我们用`pip`安装virtualenv：

```
$ pip3 install virtualenv
```

然后，假定我们要开发一个新的项目，需要一套独立的Python运行环境，可以这么做：

第一步，创建目录：

```
Mac:~ michael$ mkdir myproject
Mac:~ michael$ cd myproject/
Mac:myproject michael$
```

第二步，创建一个独立的Python运行环境，命名为`venv`：

```
Mac:myproject michael$ virtualenv --no-site-packages venv
Using base prefix '/usr/local/.../Python.framework/Versions/3.4'
New python executable in venv/bin/python3.4
Also creating executable in venv/bin/python
Installing setuptools, pip, wheel...done.
```

命令`virtualenv`就可以创建一个独立的Python运行环境，我们还加上了参数`--no-site-packages`，这样，已经安装到系统Python环境中的所有第三方包都不会复制过来，这样，我们就得到了一个不带任何第三方包的“干净”的Python运行环境。

新建的Python环境被放到当前目录下的`venv`目录。有了`venv`这个Python环境，可以用`source`进入该环境：

```
Mac:myproject michael$ source venv/bin/activate
(venv)Mac:myproject michael$
```

注意到命令提示符变了，有个`(venv)`前缀，表示当前环境是一个名为`venv`的Python环境。

下面正常安装各种第三方包，并运行`python`命令：

```
(venv)Mac:myproject michael$ pip install jinja2
...
Successfully installed jinja2-2.7.3 markupsafe-0.23
(venv)Mac:myproject michael$ python myapp.py
...
```

在`venv`环境下，用`pip`安装的包都被安装到`venv`这个环境下，系统Python环境不受任何影响。也就是说，`venv`环境是专门针对`myproject`这个应用创建的。

退出当前的`venv`环境，使用`deactivate`命令：

```
(venv)Mac:myproject michael$ deactivate
Mac:myproject michael$
```

此时就回到了正常的环境，现在`pip`或`python`均是在系统Python环境下执行。

完全可以针对每个应用创建独立的Python运行环境，这样就可以对每个应用的Python环境进行隔离。

virtualenv是如何创建“独立”的Python运行环境的呢？原理很简单，就是把系统Python复制一份到virtualenv的环境，用命令`source venv/bin/activate`进入一个virtualenv环境时，virtualenv会修改相关环境变量，让命令`python`和`pip`均指向当前的virtualenv环境。

## 小结

- virtualenv为应用提供了隔离的Python运行环境，解决了不同应用间多版本的冲突问题。