

12. 虚拟环境和包

12.1. 概述

Python应用程序通常会使用不在标准库内的软件包和模块。应用程序有时需要特定版本的库，因为应用程序可能需要修复特定的错误，或者可以使用库的过时版本的接口编写应用程序。

这意味着一个Python安装可能无法满足每个应用程序的要求。如果应用程序A需要特定模块的1.0版本但应用程序B需要2.0版本，则需求存在冲突，安装版本1.0或2.0将导致某一个应用程序无法运行。

这个问题的解决方案是创建一个 **virtual environment**，一个目录树，其中安装有特定Python版本，以及许多其他包。

然后，不同的应用将可以使用不同的虚拟环境。要解决先前需求相冲突的例子，应用程序 A 可以拥有自己的 安装了 1.0 版本的虚拟环境，而应用程序 B 则拥有安装了 2.0 版本的另一个虚拟环境。如果应用程序 B 要求将某个库升级到 3.0 版本，也不会影响应用程序 A 的环境。

12.2. 创建虚拟环境

用于创建和管理虚拟环境的模块称为 **venv**。**venv** 通常会安装你可用的最新版本的 Python。如果您的系统上有多个版本的 Python，您可以通过运行 `python3` 或您想要的任何版本来选择特定的Python版本。

要创建虚拟环境，请确定要放置它的目录，并将 **venv** 模块作为脚本运行目录路径：

```
python3 -m venv tutorial-env
```

这将创建 `tutorial-env` 目录，如果它不存在的话，并在其中创建包含 Python 解释器副本和各种支持文件的目录。

虚拟环境的常用目录位置是 `.venv`。这个名称通常会令该目录在你的终端中保持隐藏，从而避免需要对所在目录进行额外解释的一般名称。它还能防止与某些工具所支持的 `.env` 环境变量定义文件发生冲突。

创建虚拟环境后，您可以激活它。

在Windows上，运行：

```
tutorial-env\Scripts\activate.bat
```

在Unix或MacOS上，运行：

```
source tutorial-env/bin/activate
```

（这个脚本是为**bash shell**编写的。如果你使用 **csh** 或 **fish shell**，你应该改用 `activate.csh` 或 `activate.fish` 脚本。）

激活虚拟环境将改变你所用终端的提示符，以显示你正在使用的虚拟环境，并修改环境以使 `python` 命令所运行的将是已安装的特定 Python 版本。例如：

```
$ source ~/envs/tutorial-env/bin/activate
(tutorial-env) $ python
Python 3.5.1 (default, May 6 2016, 10:59:36)
...
>>> import sys
>>> sys.path
['', '/usr/local/lib/python35.zip', ...,
'~/envs/tutorial-env/lib/python3.5/site-packages']
>>>
```

12.3. 使用pip管理包

你可以使用一个名为 **pip** 的程序来安装、升级和移除软件包。默认情况下 `pip` 将从 Python Package Index <<https://pypi.org>> 安装软件包。你可以在你的 **web** 浏览器中查看 **Python Package Index**。

`pip` 有许多子命令："install", "uninstall", "freeze" 等等。（请在 [安装 Python 模块](#) 指南页查看完整的 `pip` 文档。）

您可以通过指定包的名称来安装最新版本的包：

```
(tutorial-env) $ python -m pip install novas
Collecting novas
  Downloading novas-3.1.1.3.tar.gz (136kB)
Installing collected packages: novas
  Running setup.py install for novas
Successfully installed novas-3.1.1.3
```

您还可以通过提供包名称后跟 `==` 和版本号来安装特定版本的包：

```
(tutorial-env) $ python -m pip install requests==2.6.0
Collecting requests==2.6.0
  Using cached requests-2.6.0-py2.py3-none-any.whl
Installing collected packages: requests
Successfully installed requests-2.6.0
```

如果你重新运行这个命令，`pip` 会注意到已经安装了所请求的版本并且什么都不做。您可以提供不同的版本号来获取该版本，或者您可以运行 `pip install --upgrade` 将软件包升级到最新版本：

```
(tutorial-env) $ python -m pip install --upgrade requests
Collecting requests
Installing collected packages: requests
  Found existing installation: requests 2.6.0
    Uninstalling requests-2.6.0:
      Successfully uninstalled requests-2.6.0
Successfully installed requests-2.7.0
```

`pip uninstall` 后跟一个或多个包名称将从虚拟环境中删除包。

`pip show` 将显示有关特定包的信息：

```
(tutorial-env) $ pip show requests
---
Metadata-Version: 2.0
Name: requests
Version: 2.7.0
Summary: Python HTTP for Humans.
Home-page: http://python-requests.org
Author: Kenneth Reitz
Author-email: me@kennethreitz.com
License: Apache 2.0
Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages
Requires:
```

`pip list` 将显示虚拟环境中安装的所有软件包：

```
(tutorial-env) $ pip list
novas (3.1.1.3)
numpy (1.9.2)
pip (7.0.3)
requests (2.7.0)
setuptools (16.0)
```

`pip freeze` 将生成一个类似的已安装包列表，但输出使用 `pip install` 期望的格式。一个常见的约定是将此列表放在 `requirements.txt` 文件中：

```
(tutorial-env) $ pip freeze > requirements.txt
(tutorial-env) $ cat requirements.txt
novas==3.1.1.3
numpy==1.9.2
requests==2.7.0
```

然后将 `requirements.txt` 提交给版本控制并作为应用程序的一部分提供。然后用户可以使用 `install -r` 安装所有必需的包：

```
(tutorial-env) $ python -m pip install -r requirements.txt
Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))
...
Collecting numpy==1.9.2 (from -r requirements.txt (line 2))
...
Collecting requests==2.7.0 (from -r requirements.txt (line 3))
...
Installing collected packages: novas, numpy, requests
  Running setup.py install for novas
Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0
```

`pip` 有更多选择。有关 `pip` 的完整文档，请参阅 [安装 Python 模块](#) 指南。当您编写一个包并希望它在 `Python` 包索引中使它可用时，请参考 [分发 Python 模块](#) 指南。



3.10.7	
Q	转向