LEARNING

virtualenv

#virtualenv

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: virtualenv

It is an unofficial and free virtualenv ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official virtualenv.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with virtualenv

Remarks

This section provides an overview of what virtualenv is, and why a developer might want to use it.

It should also mention any large subjects within virtualenv, and link out to the related topics. Since the Documentation for virtualenv is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Virtual Environment tool (virtualenv) is used to isolate different projects and their dependencies by creating individual python environments for each of them. It's like installing a package locally (and not globally), similar to npm package installation option. Following is an example to install and test virtualenv for creating two projects (Project1-A Django application and Project2- A Flask application):

1. Initially check if virtualenv is already installed `$ virtualenv --version`
2. Run `$ pip install virtualenv` (for Mac and Linux) or `$ sudo apt-get install python-virtualenv` for Ubuntu, `easy_install` for Windows to install the python environment.
3. `$ mkdir Project1` and `$ cd Project1`
4. Run `$ virtualenv venvp1` and this would create a venvp1 folder inside Project1 directory.
5. To activate the environment run `source venvp1/bin/activate` (if Linux) and `venvp1\Scripts\activate` (if Windows) and prompt will change to `(venvp1)Your-Computer:your_project UserName$)`
6. Run `pip install Django` to install Django for project1 and `deactivate` (if needed) to return to the global environment.
7. Repeat steps 3-6 for Flask application with different directory, virtualenv names and `pip install Flask` to install Flask.

Once above steps are executed (without any errors) one could (possibly and) simultaneously work between both environments without any conflicts.

Notes:

1. virtualenvwrapper is another handy tool which is extended version of virtualenv, though the installation procedure for both is nearly same.
2. Executing `virtualenv` command with `--no-site-packages` excludes the globally installed packages.
3. To freeze current state of environment run `$ pip freeze > installedpkgp1.txt`. This text file contains list of installed packages (including their versions) in the current environment. If there comes a need to deploy same environment at different folder (or machine) simply executing the command `$ pip install -r installedpkgp1.txt` would create same

environment.

4. Useful commands:

- `lsvirtualenv` - list of all environments
- `cdvirtualenv` - goto currently activated virtual environment
- `cdsitepackages` - like previous, but goes directly to `site-packages` directory
- `lssitepackages` - shows content of `site-packages` directory

Read Getting started with virtualenv online: https://riptutorial.com/virtualenv/topic/9733/getting-started-with-virtualenv

Credits

S. No	Chapters	Contributors
1	Getting started with virtualenv	Ayrat, Community, w1n5rx