Installing a Python wheel

WE'LL COVER THE FOLLOWING

Let’s create a virtualenv to test with. We will use the following command to create our virtual testing environment:

```
virtualenv test
```

This assumes that **virtualenv** is on your system path. If you get an **unrecognized command** error, then you’ll probably have to specify the full path (i.e. **c:\Python34\Scripts\virtualenv**). Running this command will create a virtual sandbox for us to play in that includes pip. Be sure to run the **activate** script from the **test** folder’s Scripts folder to enable the virtuanenv before continuing. Your virtualenv does not include wheel, so you’ll have to install wheel again:

```
pip install wheel
```

Once that is installed, we can install our wheel with the following command:

```
pip install --use-wheel --no-index --find-links=path/to/my_wheels Unidecode
```

To test that this worked, run Python from the Scripts folder in your virtualenv and try importing unidecode. If it imports, then you successfully installed your wheel!

The *.whl file is similar to an*.egg in that it’s basically a *.zip file in disguise. If you rename the extension from*.whl to *.zip, you can open it up with your zip application of choice and examine the files and folders inside at your leisure.

Wrapping Up

Now you should be ready to create your own wheels. They are a nice way to create a local repository of dependencies for your project(s) that you can install quickly. You could create several different wheel repositories to easily switch between different version sets for testing purposes. When combined with virtualenv, you have a really easy way to see how newer versions of dependencies could affect your project without needing to download them multiple times.