PyBuilder Documentation

Release 0.10

PyBuilder Team

May 25, 2018

Zusammenfassung der Anmerkungen auf pybuilder_Lonerider2010

Seite: 1

Seite: 1

Nummer: 1 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 26.05.2018, 07:57:22

We should clarify if we handle Python 2(.7), Python 3(.5), or both.

Seite: 1

Nummer: 2 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:23:12

The main purpose of this documentation is to have all information about PyBuilder in one place, right?
Or two places, the web-based documentation, e.g. at ReadTheDocs, and this PDF.
So we should consolidate all bits and pieces of documentation, to be found anywhere in the net, into this one piece.

Seite: 1

Nummer: 3 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:31:38

Branch: feature/docs_0.12

Seite: 1

Nummer: 4 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 25.05.2018, 22:51:18

Change to 0.11

CONTENTS

Seite: 3

Seite: 3

Nummer: 5 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 12:54:27

The sequence Installation - Concepts - Complete walkthrough is a bit confusing.
If we start with the Concepts, we can omit the Installation, as it is repeated in the Complete walkthrough.

CHAPTER

ONE

INSTALLATION

PyBuilder is available on PyPI, so you can install it with

```
$ pip install pybuilder
```

1.1 Virtual Environment

We recommend installing PyBuilder into a virtual environment using pip:

```
$ virtualenv venv
```

Note: At first it might seem tempting to install PyBuilder system-wide with `sudo pip install pybuilder`, but if you work with virtualenvs then PyBuilder will see your system python (due to being installed there) instead of the virtualenv python.

1.2 Installing completions

If you are a zsh or fish shell user, we recommend installing the pybuilder-completions. These will provide tab-based completions for PyBuilder options and tasks on a per-project basis.

```
sudo pip install pybuilder-completions
```

Note: The completions can be installed system-wide since they are just files for the relevant shells.

Seite: 5

Seite: 5

Nummer: 6 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:25:15

Activation is missing:
source venv/bin/activate

Seite: 5

Nummer: 7 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:32:29

Does not work in bash. An error message occurs: Command "python setup.py egg_info" failed with error code 1 in /tmp/pip-install-xpycx7ql/pybuilder-completions/

Seite: 5

Nummer: 8 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:24:34

An introduction is missing.
The Introduction chapter of the Usage Documentation would fit here.

Seite: 5

Nummer: 9 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 13:04:11

for Python 3:
pip3 install pybuilder

Seite: 5

Nummer: 10 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 13:03:38

for Python 3:
sudo pip3 install pybuilder-completions
doesn't work either

Seite: 5

Nummer: 11 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 13:03:12

for Python 3:
virtualenv --python=python3.5 venv

CHAPTER

ONE

INSTALLATION

PyBuilder is available on PyPI, so you can install it with

```
$ pip install pybuilder
```

1.1 Virtual Environment

We recommend installing PyBuilder into a virtual environment using pip:

```
$ virtualenv venv
```

Note: At first it might seem tempting to install PyBuilder system-wide with `sudo pip install pybuilder`, but if you work with virtualenvs then PyBuilder will see your system python (due to being installed there) instead of the virtualenv python.

1.2 Installing completions

If you are a zsh or fish user, we recommend installing the pybuilder-completions. These will provide tab-based completions for PyBuilder options and tasks on a per-project basis.

```
sudo pip install pybuilder-completions
```

Note: The completions can be installed system-wide since they are just files for the relevant shells.

Seite: 5

Nummer: 12 Typ: Text Autor: Lonerider2010 Betreff: Remark Datum: 27.05.2018, 12:24:45

Should we also show the messages after executing commands?

Seite: 5

Nummer: 13 Typ: Text Autor: Lonerider2010 Betreff: Remark Datum: 26.05.2018, 10:40:53

What about the Bash users?

Seite: 5

Nummer: 14 Typ: Line Autor: Lonerider2010 Betreff: Linie Datum: 27.05.2018, 12:25:09

Should be described the other way round:
Step 1: set the virtual environment
Step 2: install pybuilder within this environment.

CHAPTER

TWO

CONCEPTS

2.1 Introduction

PyBuilder is a multi-purpose software buil 16 Most commonly it targets the building and management of software with a strong focus on Python.

2.2 Advantages for python projects

Some of th 15 bilities provided by PyBuilder out-of-the box are:

- Automatic execution of unit and integration tests on every build
- Automatic analysis of the code coverage
- Automatic execution and result interpretation of analysis tools, such as flake8
- Automatic generation of distutils script `setup.py`

The general idea is that everything you do in your continuous integration chain, you also do locally before checking in your work.

2.3 Why Another Build Tool

When working on large scale software projects based on Java and Groovy I delved into the build process using tools such as Apache Ant, Apache Maven or Gradle. Although none of these tools is perfect they all provide a powerful and extensible way for building and testing software.

When focusing on Python I looked for a similar tool and got frustrated by the large number of tools that all match some aspect of the build and test process. Unfortunately, many of those tools were not suitable for composition and there was no central point of entry.

I suddenly found myself writing "build scripts" in Python over and over again using the tools I found out to be useful.

PyBuilder was born on the attempt to create a reusable tool that should:

- Make simple things simple
- Make hard things as simple as possible
- Let me use whatever tool I want to integrate
- Integrate these tools into a common view
- Let me use Python (which is really great) to write my build files

Seite: 7

Seite: 7

Nummer: 15 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:28:23

Are there more capabilities?
Why don't we mention them?

Seite: 7

Nummer: 16 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:25:41

Is it a CI tool? Then we should mention it here.

2.4 Design

PyBuilder executes build logic that is organized into tasks and actions.

Tasks are the main building blocks of the build logic. A task is an enclosed piece of build logic to be executed as a single unit. Each task can name a set of other tasks that it depends on. PyBuilder ensures that a task gets executed only after all of its dependencies have been executed.

Actions are smaller pieces of build logic than tasks. They are bound to the execution of task. Each action states that it needs to be executed before or after a named task. PyBuilder will execute the action if and only if the named task is executed, either directly or through another tasks' dependencies.

Actions as well as tasks are decorated plain old functions. Thus, you can structure your code the way you like if you provide a single point of entry to a build script.

Both task and action functions can request parameters known to PyBuilder through dependency injection by parameter name.

Seite: 8

Seite: 8

Nummer: 17 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:29:06

Would be clearer with an example.

Seite: 8

Nummer: 18 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:28:54

The difference between tasks and actions is not quite clear to me!

CHAPTER

THREE

COMPLETE WALKTHROUGH FOR A NEW PYBUILDER PROJECT

3.1 Installing PyBuilder

We'll start by creating a folder for our new project:

```
mkdir myproject
cd myproject
```

Then, onto creating a virtualenv and install PyBuilder inside it:

```
virtualenv venv
source venv/bin/activate
pip install pybuilder
```

22

3.2 Scaffolding

Now we can use PyBuilder's own scaffolding capabilities

```
(venv) mriehl@isdeblnnl084 myproject $ pyb
Project name (default: 'myproject') :
Source directory (default: 'src/main/python') :
Docs directory (default: 'docs') :
Unittest directory (default: 'src/unittest/python') :
Scripts directory (default: 'src/main/scripts') :
Use plugin python.flake8 (Y/n)? (default: 'y') :
Use plugin python.coverage (Y/n)? (default: 'y') :
Use plugin python.distutils (Y/n)? (default: 'y') :
```

21 20 23

As you can see, this created the content roots automatically:

```
(venv) mriehl@isdeblnnl084 myproject $ ll --
inode Permissions Size Blocks User   Group  Date Modified Name
 1488 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46  .
 1521 .rw-r--r--   324      8 mriehl admins 28 Jul 17:46  ├── build.py
 2844 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46  ├── docs
 2143 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46  └── src
 2789 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46      ├── main
 2803 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46      │   ├── python
 2864 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46      │   └── scripts
 2827 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46      └── unittest
 2829 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46          └── python
```

19

Seite: 9

Seite: 9

Nummer: 19 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 12:55:47

'll -tree' does not work in Ubuntu. I use 'tree -L 3' instead.
The result looks like:
(venv) tester@testmachine:~/workspace/myproject$ tree -L 3

```
.
├── build.py
├── docs
├── setup.py
└── src
    ├── main
    │   ├── python
    │   └── scripts
    └── unittest
        └── python
```

(without the venv directory)

Seite: 9

Nummer: 20 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:29:37

In this case, all questions are answered with the default (= return).

Seite: 9

Nummer: 21 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 26.05.2018, 16:32:07

The user input should be emphasized. Also for the next commands.

Seite: 9

Nummer: 22 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 13:06:06

for Python 3:
virtualenv --python=python3.5 venv
source venv/bin/activate
pip3 install pybuilder

Seite: 9

Nummer: 23 Typ: Highlight Autor: Lonerider2010 Betreff: Hervorheben Datum: 26.05.2018, 16:31:09

CHAPTER

THREE

COMPLETE WALKTHROUGH FOR A NEW PYBUILDER PROJECT

3.1 Installing PyBuilder

We'll start by creating a folder for our new project:

```
mkdir myproject
cd myproject
```

Then, onto creating a virtualenv and install PyBuilder inside it:

```
virtualenv venv
source venv/bin/activate
pip install pybuilder
```

25

3.2 Scaffolding

Now we can use PyBuilder's own scaffolding capabilities:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb --start-project
Project name (default: 'myproject') :
Source directory (default: 'src/main/python') :
Docs directory (default: 'docs') :
Unittest directory (default: 'src/unittest/python') :
Scripts directory (default: 'src/main/scripts') :
Use plugin python.flake8 (Y/n)? (default: 'y') :
Use plugin python.coverage (Y/n)? (default: 'y') :
Use plugin python.distutils (Y/n)? (default: 'y') :
```

24

As you can see, this created the content roots automatically:

```
(venv) mriehl@isdeblnnl084 myproject $ ll --
inode Permissions Size Blocks User   Group  Date Modified Name
 1488 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46  .
 1521 .rw-r--r--   324      8 mriehl admins 28 Jul 17:46  ├── build.py
 2844 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46  ├── docs
 2143 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46  └── src
 2789 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46      ├── main
 2803 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46      │   ├── python
 2864 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46      │   └── scripts
 2827 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46      └── unittest
 2829 drwxr-xr-x     -      - mriehl admins 28 Jul 17:46          └── python
```


Seite: 9

Nummer: 24 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:29:38

pyb finishes with "Created 'setup.py'." More interestingly for the user is the creation of 'build.py' as seen on the next page.Should we change that?

Seite: 9

Nummer: 25 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:29:16

Same as in chapter 1?

3.3 Our new `build.py`

Let us now take a look at the `build.py` which is the centralized project description for our new project. ~~The annotated contents are:~~

```
from pybuilder.core import use_plugin, init

# These are the plugins we want to use in our project.
# Projects provide tasks which are blocks of logic executed by PyBuilder.

use_plugin("python.core")
# the python unittest plugin allows running python's standard library unittests
use_plugin("python.unittest")
# this plugin allows installing project dependencies with pip
use_plugin("python.install_dependencies")
# a linter plugin that runs flake8 (pyflakes + pep8) on our project sources
use_plugin("python.flake8")
# a plugin that measures unit test statement coverage
use_plugin("python.coverage")
# for packaging purposes since we'll build a tarball
use_plugin("python.distutils")


# The project name
name = "myproject"
# What PyBuilder should run when no tasks are given.
# Calling "pyb" amounts to calling "pyb publish" here.
# We could run several tasks by assigning a list to `default_task`.
default_task = "publish"


# This is an initializer, a block of logic that runs before the project is built.
@init
def set_properties(project):
    # Nothing happens here yet, but notice the `project` argument which is
↪automatically injected.
    pass
```

Let's run PyBuilder and see what happens:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb
PyBuilder version 0.10.63
Build started at 2015-07-28 17:55:
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task publish
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[WARN]  No unit tests executed.
[INFO]  All unit tests passed.
[INFO]  Building distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
[INFO]  Copying scripts to /tmp/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO]  Writing setup.py as /tmp/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO]  Collecting coverage information
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
```

(continues on next page)

Seite: 10

Seite: 10

Nummer: 26 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:31:28

pybuilder should generate build.py always in the annotated form. Would be very useful!

Seite: 10

Nummer: 27 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 26.05.2018, 16:14:29

The contents, expanded with annotations as comments, are:

Seite: 10

Nummer: 28 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:34:43

(venv) tester@testmachine:~/workspace/myproject$ pyb
PyBuilder version 0.12.0.dev20180422165220
Build started at 2018-05-29 08:14:50

[INFO] Building myproject version 1.0.dev0
[INFO] Executing build in /home/tester/workspace/myproject
[INFO] Going to execute task publish
[INFO] Installing plugin dependency coverage
[INFO] Installing plugin dependency flake8
[INFO] Installing plugin dependency pypandoc
[INFO] Installing plugin dependency twine
[INFO] Installing plugin dependency unittest-xml-reporting
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[WARN] No unit tests executed.
[INFO] All unit tests passed.
[INFO] Building distribution in /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0
[INFO] Copying scripts to /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO] Writing setup.py as /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO] Collecting coverage information
[WARN] coverage_branch_threshold_warn is 0 and branch

3.3 Our new `build.py`

Let us now take a look at the `build.py` which is the centralized project description for our new project. ~~The annotated contents are:~~

30

```
from pybuilder.core import use_plugin, init

# These are the plugins we want to use in our project.
# Projects provide tasks which are blocks of logic executed by PyBuilder.

use_plugin("python.core")
# the python unittest plugin allows running python's standard library unittests
use_plugin("python.unittest")
# this plugin allows installing project dependencies with pip
use_plugin("python.install_dependencies")
# a linter plugin that runs flake8 (pyflakes + pep8) on our project sources
use_plugin("python.flake8")
# a plugin that measures unit test statement coverage
use_plugin("python.coverage")
# for packaging purposes since we'll build a tarball
use_plugin("python.distutils")


# The project name
name = "myproject"
# What PyBuilder should run when no tasks are given.
# Calling "pyb" amounts to calling "pyb publish" here.
# We could run several tasks by assigning a list to `default_task`.
default_task = "publish"


# This is an initializer, a block of logic that runs before the project is built.
@init
def set_properties(project):
    # Nothing happens here yet, but notice the `project` argument which is
↪automatically injected.
    pass
```

31

29

Let's run PyBuilder and see what happens:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb
PyBuilder version 0.10.63
Build started at 2015-07-28 17:55:53
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task publish
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[WARN]  No unit tests executed.
[INFO]  All unit tests passed.
[INFO]  Building distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
[INFO]  Copying scripts to /tmp/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO]  Writing setup.py as /tmp/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO]  Collecting coverage information
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
```

(continues on next page)

coverage will not be checked
[WARN] coverage_branch_partial_threshold_warn is 0 and partial branch coverage will not be checked
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[WARN] No unit tests executed.
[INFO] All unit tests passed.
Coverage.py warning: No data was collected. (no-data-collected)
[INFO] Overall coverage is 100%
[INFO] Overall coverage branch coverage is 100%
[INFO] Overall coverage partial branch coverage is 100%

BUILD FAILED - No data to report.

Build finished at 2018-05-29 08:15:00
Build took 10 seconds (10025 ms)

Seite: 10

Nummer: 29 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:31:44

Let's run PyBuilder without command flags and see what happens:

Seite: 10

Nummer: 30 Typ: StrikeOut Autor: Lonerider2010 Betreff: Durchstreichen Datum: 26.05.2018, 16:13:43

Seite: 10

Nummer: 31 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:31:35

Instead of a line break, the comment should be a two-liner. Then the code is easier to copy.

(continued from previous page)

```
[WARN]  No unit tests executed.
[INFO]  All unit tests passed.
[WARN]  Overall coverage is below 70%:  0%
Coverage.py warning: No data was collected.
------------------------------------------------------------
BUILD FAILED - Test coverage for at least one module is below 70%
------------------------------------------------------------
Build finished at 2015-07-28 17:55:54
Build took 0 seconds (515 ms)
```

We don't have any tests so our coverage is zero percent, all right! We have two ways to go about this - coverage breaks the build by default, so we can (if we want to) choose to not break the build based on the coverage metrics. This logic belongs to the project build, so we would have to add it to our build.py in the initializer. You can think of the initializer as a function that sets some configuration values before PyBuilder moves on to the actual work:

```
# This is an initializer, a block of logic that runs before the project is built.
@init
def set_properties(project):
    project.set_property("coverage_break_build", False) # default is True
```

With the above modification the coverage plugin still complains but it does not break the build. Since we're clean coders, we're going to add some production code with a test though!

3.4 Our first test

We'll write an application that outputs "Hello world". Let's start with a test at `src/unittest/python/myproject_tests.py`:

```
from unittest import TestCase

from mock import Mock

from myproject import greet


class Test(TestCase):

    def test_should_write_hello_world(self):
        mock_stdout = Mock()

        greet(mock_stdout)

        mock_stdout.write.assert_called_with("Hello world!\n")
```

Note: As a default, the unittest plugin finds tests if their filename ends with `_tests.py`. We could change this with a well-placed `project.set_property` of course.

3.4.1 Our first dependency

Since we're using mock, we'll have to install it by telling our initializer in `build.py` about it:

Seite: 11

Seite: 11

Nummer: 32 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:32:30

Coverage.py warning:
This message is missing the severity, right?

Seite: 11

Nummer: 33 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 10:13:51

No change in the output messages, only the build is faster (592 msvs.7498 ms)

Seite: 11

Nummer: 34 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:35:23

(venv) tester@testmachine:~/workspace/myproject$ pyb
PyBuilder version 0.12.0.dev20180422165220
Build started at 2018-05-29 08:16:36

[INFO] Building myproject version 1.0.dev0
[INFO] Executing build in /home/tester/workspace/myproject
[INFO] Going to execute task publish
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[WARN] No unit tests executed.
[INFO] All unit tests passed.
[INFO] Building distribution in /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0
[INFO] Copying scripts to /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO] Writing setup.py as /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO] Collecting coverage information
[WARN] coverage_branch_threshold_warn is 0 and branch coverage will not be checked
[WARN] coverage_branch_partial_threshold_warn is 0 and partial branch coverage will not be checked
[INFO] Running unit tests

(continued from previous page)

```
[WARN]  No unit tests executed.
[INFO]  All unit tests passed.
[WARN]  Overall coverage is below 70%:  0%
Coverage.py warning: No data was collected.
------------------------------------------------------------
BUILD FAILED - Test coverage for at least one module is below 70%
------------------------------------------------------------
Build finished at 2015-07-28 17:55:54
Build took 0 seconds (515 ms)
```

We don't have any tests so coverage is zero percent, all right! We have two ways to go about this - coverage breaks the build by default, so we can (if we want to) choose to not break the build based on the coverage metrics. This logic belongs to the project build, so we would have to add it to our build.py in the initializer. You can think of the initializer as a function that sets some configuration values before PyBuilder moves on to the actual work:

```
# This is an initializer, a block of logic that runs before the project is built.
@init
def set_properties(project):
    project.set_property("coverage_break_build", False) # default is True
```

With the above modification, the coverage plugin still complains but it does not break the build. Since we're clean coders, we're going to add some production code with a test though!

3.4 Our first test

We'll write an application that outputs "Hello world". Let's start with a test at `src/unittest/python/myproject_tests.py`:

```
from unittest import TestCase

from mock import Mock

from myproject import greet


class Test(TestCase):

    def test_should_write_hello_world(self):
        mock_stdout = Mock()

        greet(mock_stdout)

        mock_stdout.write.assert_called_with("Hello world!\n")
```

Note: As a default, the unittest plugin finds tests if their filename ends with `_tests.py`. We could change this with a well-placed `project.set_property` of course.

3.4.1 Our first dependency

Since we're using mock, we'll have to install it by telling our initializer in `build.py` about it:

[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[WARN] No unit tests executed.
[INFO] All unit tests passed.
Coverage.py warning: No data was collected. (no-data-collected)
[INFO] Overall coverage is 100%
[INFO] Overall coverage branch coverage is 100%
[INFO] Overall coverage partial branch coverage is 100%

BUILD FAILED - No data to report.

Build finished at 2018-05-29 08:16:37
Build took 0 seconds (757 ms)

Seite: 11

Nummer: 35 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:32:56

With version 0.11.17 the output is different: there comes a coverage warning: no-data-collected, but the overall coverage is shown as 100%! Seems to be an error of PyBuilder?

```
# This is an initializer, a block of logic that runs before the project is built.
@init
def set_properties(project):
    project.set_property("coverage_break_build", False) # default is True
    project.build_depends_on("mock")
```

We could require a specific version and so on but let's keep it simple. Also note that we declared `mock` as a build dependency - this means it's only required for building and if we upload our project to PyPI then installing it from there will not require installing `mock`.

We can install our dependency by running PyBuilder the corresponding task:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb install_dependencies
PyBuilder version 0.10.63
Build started at 2015-07-28 19:35:37
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task install_dependencies
[INFO]  Installing all dependencies
[INFO]  Installing build dependencies
[INFO]  Installing dependency 'coverage'
[INFO]  Installing dependency 'flake8'
[INFO]  Installing dependency 'mock'
[INFO]  Installing runtime dependencies
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
             Project: myproject
             Version: 1.0.dev0
      Base directory: /tmp/myproject
        Environments:
               Tasks: install_dependencies [1480 ms]
Build finished at 2015-07-28 19:35:39
Build took 1 seconds (1486 ms)
pyb install_dependencies  1.44s user 0.10s system 98% cpu 1.570 total
```

36

3.4.2 Running our test

We can run our test now:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb verify
PyBuilder version 0.10.63
Build started at 2015-07-28 19:36:41
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task verify
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[ERROR] Import error in test file /tmp/myproject/src/unittest/python/myproject_tests.
↪py, due to statement 'from myproject import greet' on line 5
[ERROR] Error importing unittest: No module named myproject
------------------------------------------------------------
BUILD FAILED - Unable to execute unit tests.
```

37 38

(continues on next page)

Seite: 12

Seite: 12

Nummer: 36 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:35:11

The dependencies coverage
flake8
pypandoc
unittest-xml-reporting
were installed already during the first pyb run.

Seite: 12

Nummer: 37 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:38:50

(venv) tester@testmachine:~/workspace/myproject$ pyb verify
PyBuilder version 0.12.0.dev20180422165220
Build started at 2018-05-29 08:18:52

[INFO] Building myproject version 1.0.dev0
[INFO] Executing build in /home/tester/workspace/myproject
[INFO] Going to execute task verify
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[ERROR] Import error in test file /home/tester/workspace/myproject/src/unittest/python/myproject_tests.py, due to statement 'from myproject import greet' on line 9
[ERROR] Error importing unittest: No module named myproject

BUILD FAILED - Unable to execute unit tests.

Build finished at 2018-05-29 08:18:52
Build took 0 seconds (437 ms)

Seite: 12

Nummer: 38 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 13:09:31

(venv) tester@testmachine:~/workspace/myproject3$ pyb verify
PyBuilder version 0.11.17
Build started at 2018-05-28 08:01:34

```
# This is an initializer, a block of logic that runs before the project is built.
@init
def set_properties(project):
    project.set_property("coverage_break_build", False) # default is True
    project.build_depends_on("mock")
```

We could require a specific version and so on but let's keep it simple. Also note that we declared `mock` as a build dependency - this means it's only required for building and if we upload our project to PyPI then installing it from there will not require installing `mock`.

We can install our dependency by running PyBuilder the corresponding task:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb install_dependencies
PyBuilder version 0.10.63
Build started at 2015-07-28 19:35:37
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task install_dependencies
[INFO]  Installing all dependencies
[INFO]  Installing build dependencies
[INFO]  Installing dependency 'coverage'
[INFO]  Installing dependency 'flake8'
[INFO]  Installing dependency 'mock'
[INFO]  Installing runtime dependencies
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
            Project: myproject
            Version: 1.0.dev0
     Base directory: /tmp/myproject
       Environments:
              Tasks: install_dependencies [1480 ms]
Build finished at 2015-07-28 19:35:39
Build took 1 seconds (1486 ms)
pyb install_dependencies  1.44s user 0.10s system 98% cpu 1.570 total
```

39

3.4.2 Running our test

We can run our test now:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb verify
PyBuilder version 0.10.63
Build started at 2015-07-28 19:36:41
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task verify
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[ERROR] Import error in test file /tmp/myproject/src/unittest/python/myproject_tests.
↪py, due to statement 'from myproject import greet' on line 5
[ERROR] Error importing unittest: No module named myproject
------------------------------------------------------------
BUILD FAILED - Unable to execute unit tests.
```

(continues on next page)

[INFO] Building myproject3 version 1.0.dev0
[INFO] Executing build in /home/tester/workspace/myproject3
[INFO] Going to execute task verify
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject3/src/unittest/python
[INFO] Executed 1 unit tests
[ERROR] Test has error:
unittest.loader._FailedTest.myproject3_tests

BUILD FAILED - There were 1 error(s) and 0 failure(s) in unit tests
---
Build finished at 2018-05-28 08:01:35
Build took 0 seconds (330 ms)

Seite: 12

Nummer: 39 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:36:48

(venv) tester@testmachine:~/workspace/myproject$ pyb install_dependencies
PyBuilder version 0.12.0.dev20180422165220
Build started at 2018-05-29 08:18:22

[INFO] Building myproject version 1.0.dev0
[INFO] Executing build in /home/tester/workspace/myproject
[INFO] Going to execute task install_dependencies
[INFO] Installing all dependencies
[INFO] Processing batch dependency 'mock'

BUILD SUCCESSFUL

Build Summary
Project: myproject
Version: 1.0.dev0
Base directory: /home/tester/workspace/myproject
Environments:
Tasks: install_dependencies [1604 ms]
Build finished at 2018-05-29 08:18:24
Build took 1 seconds (1612 ms)

```
# This is an initializer, a block of logic that runs before the project is built.
@init
def set_properties(project):
    project.set_property("coverage_break_build", False) # default is True
    project.build_depends_on("mock")
```

We could require a specific version and so on but let's keep it simple. Also note that we declared `mock` as a build dependency - this means it's only required for building and if we upload our project to PyPI then installing it from there will not require installing `mock`.

We can install our dependency by running PyBuilder with the corresponding task:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb install_dependencies
PyBuilder version 0.10.63
Build started at 2015-07-28 19:35:37
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task install_dependencies
[INFO]  Installing all dependencies
[INFO]  Installing build dependencies
[INFO]  Installing dependency 'coverage'
[INFO]  Installing dependency 'flake8'
[INFO]  Installing dependency 'mock'
[INFO]  Installing runtime dependencies
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
             Project: myproject
             Version: 1.0.dev0
      Base directory: /tmp/myproject
        Environments:
               Tasks: install_dependencies [1480 ms]
Build finished at 2015-07-28 19:35:39
Build took 1 seconds (1486 ms)
pyb install_dependencies  1.44s user 0.10s system 98% cpu 1.570 total
```

3.4.2 Running our test

We can run our test now:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb verify
PyBuilder version 0.10.63
Build started at 2015-07-28 19:36:41
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task verify
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[ERROR] Import error in test file /tmp/myproject/src/unittest/python/myproject_tests.
↪py, due to statement 'from myproject import greet' on line 5
[ERROR] Error importing unittest: No module named myproject
------------------------------------------------------------
BUILD FAILED - Unable to execute unit tests.
```

(continues on next page)

Seite: 12

Nummer: 40 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 12:34:39

Question about the pyb cli: Why is it
pyb --start-project, but
pyb install-dependencies (without --)?

(continued from previous page)

```
------------------------------------------------------------
Build finished at 2015-07-28 19:36:41
Build took 0 seconds (249 ms)
```

It's still failing because we haven't implemented anything yet. Let's do that right now in `src/main/python/myproject/__init__.py`:

```
def greet(filelike):
    filelike.write("Hello world!\n")
```

Any finally rerun the test:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb verify
PyBuilder version 0.[illegible]3
Build started at 2015-07-28 19:39:15
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task verify
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Building distribution in /tmp/[illegible]oject/target/dist/myproject-1.0.dev0
[INFO]  Copying scripts to /tmp/myproj[illegible]target/dist/myproject-1.0.dev0/scripts
[INFO]  Writing setup.py as /tmp/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO]  Collecting coverage information
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Overall coverage is 100%
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
            Project: myproject
            Version: 1.0.dev0
     Base directory: /tmp/myproject
       Environments:
              Tasks: prepare [231 ms] compile_sources [0 ms] run_unit_tests [10 ms]␣
↪package [1 ms] run_integration_tests [0 ms] verify [255 ms]
Build finished at 2015-07-28 19:39:15
Build took 0 seconds (504 ms)
```

3.5 Adding a script

Since our library is ready, we can now add a script.

We'll just need to create `src/main/scripts/greeter`:

```
#!/usr/bin/env python
import sys
from myproject import greet
```

(continues on next page)

Seite: 13

Seite: 13

Nummer: 41 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:40:38

(venv) tester@testmachine:~/workspace/myproject$ pyb verify
PyBuilder version 0.12.0.dev20180422165220
Build started at 2018-05-29 08:20:25

[INFO] Building myproject version 1.0.dev0
[INFO] Executing build in /home/tester/workspace/myproject
[INFO] Going to execute task verify
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[INFO] Executed 1 unit tests
[INFO] All unit tests passed.
[INFO] Building distribution in /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0
[INFO] Copying scripts to /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO] Writing setup.py as /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO] Collecting coverage information
[WARN] coverage_branch_threshold_warn is 0 and branch coverage will not be checked
[WARN] coverage_branch_partial_threshold_warn is 0 and partial branch coverage will not be checked
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[INFO] Executed 1 unit tests
[INFO] All unit tests passed.
[INFO] Overall coverage is 100%
[INFO] Overall coverage branch coverage is 100%
[INFO] Overall coverage partial branch coverage is 100%

BUILD SUCCESSFUL

Build Summary

(continued from previous page)

```
------------------------------------------------------------
Build finished at 2015-07-28 19:36:41
Build took 0 seconds (249 ms)
```

It's still failing because we haven't implemented anything yet. Let's do that right now in `src/main/python/myproject/__init__.py`:

```
def greet(filelike):
    filelike.write("Hello world!\n")
```

Any finally rerun the test:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb verify
PyBuilder version 0.[illegible]3
Build started at 2015-07-[illegible]8 19:39:15
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task verify
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Building distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
[INFO]  Copying scripts to /tmp/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO]  Writing setup.py as /tmp/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO]  Collecting coverage information
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Overall coverage is 100%
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
            Project: myproject
            Version: 1.0.dev0
     Base directory: /tmp/myproject
       Environments:
              Tasks: prepare [231 ms] compile_sources [0 ms] run_unit_tests [10 ms]
↪package [1 ms] run_integration_tests [0 ms] verify [255 ms]
Build finished at 2015-07-28 19:39:15
Build took 0 seconds (504 ms)
```

3.5 Adding a script

Since our library is ready, we can now add a script.

We'll just need to create `src/main/scripts/greeter`:

```
#!/usr/bin/env python
import sys
from myproject import greet
```

(continues on next page)

Project: myproject
Version: 1.0.dev0
Base directory: /home/tester/workspace/myproject
Environments:
Tasks: prepare [287 ms] compile_sources [0 ms] run_unit_tests [91 ms] package [2 ms] run_integration_tests [0 ms] verify [694 ms]
Build finished at 2018-05-29 08:20:26
Build took 1 seconds (1084 ms)

Seite: 13

Nummer: 42 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 16:57:00

should be 'greeter.py', right?

Seite: 13

Nummer: 43 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 10:37:45

Still warnings about the coverage.

(continued from previous page)

```
greet(sys.stdout)
```

Note that there is nothing else to do. Dropping the file in `src/main/scripts` is all we need to do for PyBuilder to pick it up, because this is the convention.

Let's look at what happens when we package it up:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb publish
PyBuilder version 0.10.63
Build started at 2015-07-28 19:44:34
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task publish
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Building distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
[INFO]  Copying scripts to /tmp/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO]  Writing setup.py as /tmp/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO]  Collecting coverage information
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Overall coverage is 100%
[INFO]  Building binary distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
             Project: myproject
             Version: 1.0.dev0
      Base directory: /tmp/myproject
        Environments:
               Tasks: prepare [227 ms] compile_sources [0 ms] run_unit_tests [9 ms]
↪package [2 ms] run_integration_tests [0 ms] verify [252 ms] publish [241 ms]
Build finished at 2015-07-28 19:44:35
Build took 0 seconds (739 ms)
```

We can now simply `pip install` the tarball:

```
(venv) mriehl@isdeblnnl084 myproject $ pip install target/dist/myproject-1.0.dev0/
↪dist/myproject-1.0.dev0.tar.gz
Processing ./target/dist/myproject-1.0.dev0/dist/myproject-1.0.dev0.tar.gz
Building wheels for collected packages: myproject
  Running setup.py bdist_wheel for myproject
  Stored in directory: /data/home/mriehl/.cache/pip/wheels/89/05/9e/
↪4b035292abf39e5d6ddcf442cc7c96c2e56f5cc49c5c673d3a
Successfully built myproject
Installing collected packages: myproject
Successfully installed myproject-1.0.dev0
(venv) mriehl@isdeblnnl084 myproject $ greeter
Hello world!
```

Of course since there is a `setup.py` in the distribution folder, we can use it to do whatever we want easily, for

Seite: 14

Seite: 14

Nummer: 44 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:42:10

(venv) tester@testmachine:~/workspace/myproject$ pyb publish
PyBuilder version 0.12.0.dev20180422165220
Build started at 2018-05-29 08:22:10

[INFO] Building myproject version 1.0.dev0
[INFO] Executing build in /home/tester/workspace/myproject
[INFO] Going to execute task publish
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[INFO] Executed 1 unit tests
[INFO] All unit tests passed.
[INFO] Building distribution in /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0
[INFO] Copying scripts to /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO] Writing setup.py as /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO] Collecting coverage information
[WARN] coverage_branch_threshold_warn is 0 and branch coverage will not be checked
[WARN] coverage_branch_partial_threshold_warn is 0 and partial branch coverage will not be checked
[INFO] Running unit tests
[INFO] Executing unit tests from Python modules in /home/tester/workspace/myproject/src/unittest/python
[INFO] Executed 1 unit tests
[INFO] All unit tests passed.
[INFO] Overall coverage is 100%
[INFO] Overall coverage branch coverage is 100%
[INFO] Overall coverage partial branch coverage is 100%
[INFO] Building binary distribution in /home/tester/workspace/myproject/target/dist/myproject-1.0.dev0

BUILD SUCCESSFUL

(continued from previous page)

```
greet(sys.stdout)
```

Note that there is nothing else to do. Dropping the file in `src/main/scripts` is all we need to do for PyBuilder to pick it up, because this is the convention.

Let's look at what happens when we package it up:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb publish
PyBuilder version 0.10.63
Build started at 2015-07-28 19:44:34
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task publish
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Building distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
[INFO]  Copying scripts to /tmp/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO]  Writing setup.py as /tmp/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO]  Collecting coverage information
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Overall coverage is 100%
[INFO]  Building binary distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
            Project: myproject
            Version: 1.0.dev0
     Base directory: /tmp/myproject
       Environments:
              Tasks: prepare [227 ms] compile_sources [0 ms] run_unit_tests [9 ms]␣
↪package [2 ms] run_integration_tests [0 ms] verify [252 ms] publish [241 ms]
Build finished at 2015-07-28 19:44:35
Build took 0 seconds (739 ms)
```

We can now simply `pip install` the tarball:

```
(venv) mriehl@isdeblnnl084 myproject $ pip install target/dist/myproject-1.0.dev0/
↪dist/myproject-1.0.dev0.tar.gz
Processing ./target/dist/myproject-1.0.dev0/dist/myproject-1.0.dev0.tar.gz
Building wheels for collected packages: myproject
  Running setup.py bdist_wheel for myproject
  Stored in directory: /data/home/mriehl/.cache/pip/wheels/05/9e/
↪4b035292abf39e5d6ddcf442cc7c96c2e56f5cc49c5c673d3a
Successfully built myproject
Installing collected packages: myproject
Successfully installed myproject-1.0.dev0
(venv) mriehl@isdeblnnl084 myproject $ greeter
Hello world!
```

Of course since there is a `setup.py` in the distribution folder, we can use it to do whatever we want easily, for

---

Build Summary
Project: myproject
Version: 1.0.dev0
Base directory: /home/tester/workspace/myproject
Environments:
Tasks: prepare [311 ms] compile_sources [0 ms] run_unit_tests [120 ms] package [3 ms] run_integration_tests [0 ms] verify [617 ms] publish [456 ms]
Build finished at 2018-05-29 08:22:11
Build took 1 seconds (1518 ms)

Seite: 14

Nummer: 45 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 08:15:49

should be
greeter.py

Seite: 14

Nummer: 46 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 28.05.2018, 08:18:11

for Python 3:
pip3 install ...

Seite: 14

Nummer: 47 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 29.05.2018, 08:42:49

(venv) tester@testmachine:~/workspace/myproject$ pip install target/dist/myproject-1.0.dev0/dist/myproject-1.0.dev0.tar.gz
Processing ./target/dist/myproject-1.0.dev0/dist/myproject-1.0.dev0.tar.gz
Building wheels for collected packages: myproject
Running setup.py bdist_wheel for myproject ... done
Stored in directory: /home/tester/.cache/pip/wheels/37/b3/ee/535029f1c8666633b18c6612b486ba893a903a598cb9cf1804
Successfully built myproject
Installing collected packages: myproject
Successfully installed myproject-1.0.dev0

(continued from previous page)

```
greet(sys.stdout)
```

Note that there is nothing else to do. Dropping the file in `src/main/scripts` is all we need to do for PyBuilder to pick it up, because this is the convention.

Let's look at what happens when we package it up:

```
(venv) mriehl@isdeblnnl084 myproject $ pyb publish
PyBuilder version 0.10.63
Build started at 2015-07-28 19:44:34
------------------------------------------------------------
[INFO]  Building myproject version 1.0.dev0
[INFO]  Executing build in /tmp/myproject
[INFO]  Going to execute task publish
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Building distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
[INFO]  Copying scripts to /tmp/myproject/target/dist/myproject-1.0.dev0/scripts
[INFO]  Writing setup.py as /tmp/myproject/target/dist/myproject-1.0.dev0/setup.py
[INFO]  Collecting coverage information
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /tmp/myproject/src/unittest/python
[INFO]  Executed 1 unit tests
[INFO]  All unit tests passed.
[INFO]  Overall coverage is 100%
[INFO]  Building binary distribution in /tmp/myproject/target/dist/myproject-1.0.dev0
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
             Project: myproject
             Version: 1.0.dev0
      Base directory: /tmp/myproject
        Environments:
               Tasks: prepare [227 ms] compile_sources [0 ms] run_unit_tests [9 ms]␣
↪package [2 ms] run_integration_tests [0 ms] verify [252 ms] publish [241 ms]
Build finished at 2015-07-28 19:44:35
Build took 0 seconds (739 ms)
```

We can now simply `pip install` the tarball:

```
(venv) mriehl@isdeblnnl084 myproject $ pip install target/dist/myproject-1.0.dev0/
↪dist/myproject-1.0.dev0.tar.gz
Processing ./target/dist/myproject-1.0.dev0/dist/myproject-1.0.dev0.tar.gz
Building wheels for collected packages: myproject
  Running setup.py bdist_wheel for myproject
  Stored in directory: /data/home/mriehl/.cache/pip/wheels/89/05/9e/
↪4b035292abf39e5d6ddcf442cc7c96c2e56f5cc49c5c673d3a
Successfully built myproject
Installing collected packages: myproject
Successfully installed myproject-1.0.dev0
(venv) mriehl@isdeblnnl084 myproject $ greeter
Hello world!
```

Of course since there is a `setup.py` in the distribution folder, we can use it to do whatever we want easily, for

You are using pip version 9.0.3, however version 10.0.1 is available. You should consider upgrading via the 'pip install --upgrade pip' command.

Seite: 14

Nummer: 48 Typ: Text Autor: Lonerider2010 Betreff: Remark
Datum: 27.05.2018, 10:42:33

still the coverage problem