

Assignments

Assignments give you the ability to set section options via the command line. Assignments are in the form of:

```
section_name:option_name=value
```

Here are some examples:

Set the log-level of the buildout section
(This is equivalent to bin/buildout -vvvv):

```
$ bin/buildout buildout:log-level=50
```

Turn on debug mode for the instance:

```
$ bin/buildout instance:debug-mode=on
```

Links

Details about how to pin a dependency:

<http://peak.telecommunity.com/DevCenter/setuptools#declaring-dependencies>

The official documentation pages for Zope:

<http://buildout.zope.org>

The Cheese Shop (pypi) page for buildout:

<http://pypi.python.org/pypi/zc.buildout>

Martin Aspeli's buildout tutorial on plone.org:

<http://plone.org/documentation/tutorial/buildout>

Vocabulary

buildout - A set of parts that describe how to assemble an application

part - A set of options that allow you to build a piece of the application

recipe - The software used to create a part based off of its options

Getting Started With a Plone Buildout

You can add a default.cfg into your \$HOME/.buildout directory to set up some user defaults for any part of the buildout:

```
[buildout]
eggs-directory = /path/to/home/.buildout/eggs
download-cache = /path/to/home/.buildout/downloads
zope-directory = /path/to/home/.buildout/zope
```

NOTE: These only provide defaults, they do not override settings in your buildout!

How to get started with a Plone buildout:

To start from scratch, you can use the ZopeSkel collection of templates:

```
$ easy_install -U ZopeSkel
$ paster create -t plone3_buildout
```

This will ask you a series of questions about your new buildout.

Once you have your buildout, you can now bootstrap it:

```
$ cd path/to/buildout
$ python2.4 bootstrap.py
$ bin/buildout
```

Now you have everything you'll need to start your site
(assuming the part names are zeoserver and instance).

If you're using a Zope Storage Server:

```
$ bin/zeoserver start
```

Now you can start your Zope instance:

```
$ bin/instance start
```

NOTE: Multiple instances are typically incremented by number
(e.g. instance1, instance2, etc.)



six feet up

Download more cards at:
www.sixfeetup.com/swag

Configuration

Reserved characters that shouldn't be used in part or option names:

```
: $ % ( )
```

Buildout configuration uses a variable substitution syntax:

```
${<part_name>:<option_name>}  
${buildout:parts-directory}
```

Options that take a list of items are done with spaces or one per indented line:

```
# base.cfg  
[part-one]  
option1 = foo bar baz  
  
option2 =  
    foo  
    bar  
    baz
```

Options can be added and subtracted from using += and -=.
In this example we are extending the above config:

```
[buildout]  
extends = base.cfg  
  
[part-one]  
option1 += bang  
option2 -= bar
```

Versions

Versions can be pinned in various ways:

```
[buildout]  
# use our list of versions to pin  
versions = release-versions  
  
[release-versions]  
plone.recipe.plone = 3.1.5.1  
archetypes.schemaextender = 1.0  
SQLAlchemy = 0.4.6  
  
[plone]  
# use the latest 3.1.x release  
recipe = plone.recipe.plone < 3.2-dev  
  
[instance]  
# use archetypes.schemaextender 1.0 to 1.4  
# explicitly use SQLAlchemy 0.4.6  
  
eggs =  
    archetypes.schemaextender >= 1.0, < 1.5  
    SQLAlchemy == 0.4.6
```

Buildout Command Line Usage

Buildout command syntax:

```
buildout [options and assignments] [command [command arguments]]
```

NOTE: Options and assignments can be interspersed.

The bin/buildout command has several options. Use this command to see them:

```
$ bin/buildout -h
```

Options

Some options can also be set through assignments, see the info below.

- v** Increase verbosity (log-level) by 10, use multiple times to increase more. (Default: 100)
- q** Decrease verbosity (log-level) by 10, same semantics as -v.
- U** Don't read in the user's default configuration (located in ~/default.cfg).
- o** Run in 'offline' mode. Buildout will not access the outside world to get its needed parts, packages, etc.
- O** Run in 'online' mode. Buildout will be allowed to access the outside world to get its needed parts, packages, etc. (Default)
- n** Run in 'newest' mode. Buildout will check each distribution to see if it is the latest version. (Default)
- N** Run in 'non-newest' mode. Buildout will not check for the latest distribution. If a distribution requires a newer version, it will still be retrieved.
- t socket_timeout** Timeout after *n* seconds of trying to download a package. (Default: none)
- c config_file** The path to an alternate configuration. (Default: buildout.cfg)
- D** Use post mortem debugging if buildout encounters an error.

Example: Run in non-newest mode, increase verbosity by 30 and timeout after 60 seconds.

```
$ bin/buildout -Nvvv -t 60
```

Commands

Buildout has several built-in commands; the most useful will be the install command.

install [parts]

If no parts are given, the buildout config's parts will be used. Otherwise the space separated list of parts will be installed:

```
$ bin/buildout install instance
```