What is Buildout (1)

```
http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.69:/component/slapos/buildout.cfg
[buildout]
extends =
../../stack/shacache-client.cfg
../lxml-python/buildout.cfg
../python-2.7/buildout.cfg
parts = slapos
find-links =
http://www.nexedi.org/static/packages/source/slapos.buildout/
versions = versions
allow-hosts =
*.googlecode.com
*.nexedi.org
*.python.org
alastairs-place.net
code.google.com
github.com
peak.telecommunity.com
```

▼ Details

We are going to review line by line the content of a buildout file. Buildout is a mature, well documented technology. You can find many tutorials on www.buildout.org web site and learn buidlout. The lines which you can read above are a copy of the file http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.69:/component/slapos/buildout.cfg which was used during the bootstrap process.

The first section of a buildout is named [buildout]. It defines which software is going to be built and how. In the case of SlapOS buildout, we can first see that SlapOS buildout extends existing buildout definitions, namely shacache-client, lxml-python and python2.7 itself.

It then define the parts variable which lists the sections which are going to be required as part of the build process. A section in a buildout file is defined by a name inside square brackets. [buildout], [lxml-python], [slapos] and [versions] are the 4 sections of SlapOS buildout file.

The find-links variable defines a repository of eggs. Eggs is a standard distribution mechanism for python packages. Buildout itself is written in python and can be extended using python language and eggs. Since not all eggs are published in Python Packing Index (PyPI), it is sometimes necessary to provide additional repositories of eggs. One should note here that the find-links variable is a generic variable used by python setuptools, the python module in charge of managing python distributions.

We use the find-links variable in our case to override the default buildout distribution with SlapOS's own buildout. SlapOS extends buildout in various ways in order to solve some minor details of standard buildout. SlapOS patches to standard buildout are not yet integrated. However, since buildout profiles are self contained, it is possible to specify which version of buildout to use (in the [versions] section) and where to find that version through find-links, as long as SlapOS's own version of buildout is published somewhere.

For buildout experts, SlapOS buildout extensions involve mainly support of distributed network caching and small bug fixes.

The versions variable specifies which buildout section is used to define specific verions of python distributions (eggs). Being able to specify the version of each egg is required in order to make sure that the exact same software is installed on every node of a distributed Cloud and that no uncontrolled upgrade will happen.

The allow-hosts variable is used to specify explicitely which sources of python distributions (eggs) are accepted. It is a useful variable to make sure that the buildout process does not get interrupted by lost connectivity to unreliable sites containing python distributions. By specifying explicitely which sites are considered to be reliable, we can quickly circumvent temporary failures of python distribution sites.

What is Buildout (2)

```
http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.69:/component/slapos/buildout.cfg
# separate from system python
include-site-packages = false
exec-sitecustomize = false
allowed-eggs-from-site-packages =
```

▼ Details

SlapOS approach to software building is to make sure that no dependency remains to anything but glibc. This bold design decision is based on years of experience with different GNU/Linux distributions and open source projects. We found that besides glibc, very few open source projects really care about upward compatibility. Libraries provided by GNU/Linux distributions often contains so many patches that it is impossible to garantee any result or portability.

We thus need to make sure that during the build process, python system libraries are not used. The statement "include-site-packages = false" makes sure that packages which are already installed with the system python which is used to run buildout will not be used and that buildout will use its owns.

The statement "exec-sitecustomize = false" makes sure that sitecustomize.py is not called and that python default behaviour will not be affected by any changes defined in sitecustomize.py.

The statement "allowed-eggs-from-site-packages =" defines an empty list of packages (eggs) to reuse from system python. It thus makes sure that not a single egg already present in the system is going to be used by buildout and that buildout will use its owns. It seems redundant with include-site-packages but prevents, through the extension mechanism of buildout, to acquire any non empty list.

For the record, a tentative SlapOS package was created for Debian GNU/Linux. However, because it currently use Debian's default python instead of its own python, some SlapOS software buildout profiles do not compile. For the time being, we recommend to packagers of SlapOS not to use system's python, else results are unpredictable. If the distribution packaging policy prevents providing your own python, then deactivate software building in the official distribution package and put a link to SlapOS official package which includes its own python and is capable of buildout SlapOS software.

What is Buildout (3)

http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.69:/component/slapos/buildout.cfg

```
[lxml-python]
python = python2.7

[slapos]
recipe = z3c.recipe.scripts
python = python2.7
eggs =
slapos.libnetworkcache
zc.buildout
${lxml-python:egg}
slapos.core
```

▼ Details

We will start here by the [slapos] section. This section is what buildout is supposed to build according to the parts definition in [buildout] section. In order to build slapos, a recipe named "z3c.recipe.scripts" is invoked. This recipe can assemble a set of eggs and generate executable scripts and interpreters which are based on those eggs. This is exactly what we want to do since we want to combine slapos.core egg, slapos.libnetworkcache (a library which provides distributed caching of downloaded files) and buildout to generate scripts such as slapconsole, slapgrid, slapgrid-supervisorctl, etc. We also specify that the python release which we want to provide together with SlapOS scripts is python2.7, as it is defined in one of the extends of the [buildout] section.

Now comes the tricky part. The ${lxml-python:egg} defines a macro expansion in buildout system. It will try to evaluate [lxml-python] section and then access the variable named "egg" in that section. It happens that the value of this variable is already known and its value is and will be "lxml".

So, why are we doing so? In reality, to circumvent a known issue in the compilation of lxml python library. Without this trick, lxml python library would link against the system libraries of libxslt and libxml. But we want to be sure that lxml python library compiles and links againts the releases of libxslt and libxml provided by our buildout. We thus make sure through that dependency and the extends definition of the [buildout] section (../lxml-python/buildout.cfg) that lxml will be compiled without system dependencies. We also override default value of [lxml-python] section so that lxml is compiled with and for python2.7. This approach of building and providing an egg as part of the buildout process is called "develop egg".

In the end, [slapos] will simply include the "lxml" egg but this egg is going to be generated dynamically through the [lxml-python] section rather than through the default setup.py mechanism of normal egg installation (which also involves compilation).

What is Buildout (4)

http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.69:/component/slapos/buildout.cfg

```
[versions]
zc.buildout = 1.5.3-dev-SlapOS-005
Jinja2 = 2.5.5
Werkzeug = 0.6.2
hexagonit.recipe.cmmi = 1.5.0
lxml = 2.3
meld3 = 0.6.7
netaddr = 0.7.5
setuptools = 0.6c12dev-r88846
slapos.core = 0.9
slapos.libnetworkcache = 0.2
xml-marshaller = 0.9.7
z3c.recipe.scripts = 1.0.1
```

```
zc.recipe.egg = 1.3.2
...
# Required by: # slapos.core==0.9
zope.interface = 3.6.4
```

▼ Details

The [versions] section is simple to understand. It simply specifies for every python distribution which version should be used. This enforces that no regressions happens as a result of some upgrade of a software component. We sometimes call it "freezing" releases.

However versions defined in [versions] section only define versions of python distributions and not of other components. There are different ways to make a version fixed for other components. Sometimes, the URL defines implicitly a fixed revision of a component. This is the case for bison (http://git.erp5.org/gitweb/slapos.git/blob/HEAD:/component/bison/buildout.cfg?js=1). And sometimes the revision is set explicitely as in the case of ERP5 profile (http://git.erp5.org/gitweb/slapos.git/blob/HEAD:/stack/erp5.cfg#l219). Revision in the case of ERP5 is a git hash (336a8d63bdcabd92bfe3d9466685e5cd47fad716).

A good practice for complex software is to introduce revision variables in software components as well as default revisions, then let the extend machinery override those variables.