Buildout

But first ... eggs

- Self-contained zero-installation distributions
- Dependencies
- Index and automatic download
  - Includes downloading source distributions and converting to eggs
- Eggs can advertise features
  - e.g. scripts to be installed

ZOPE

Eggs!

Eggs are revolutionary!

IMO,
the advent of eggs
is the most important thing
happening with Python now!

Pay attention to eggs!

ZOPE

Eggs are simple

Eggs are directories

that can be included in sys.path.

(The fact that they can be installed as zip files is incidental)

ZOPE

The hard part

The trick is actually including the eggs on sys.path

- Deciding which eggs to include
- Constructing sys.path

ZOPE

Setuptools and easy_install

- Setuptools
  - Access egg dependencies and other meta data
  - Analyze dependencies and compute compatible sets of eggs
  - Building eggs
- Easy install
  - Find distributions on the net
  - Install build into specified locations
  - Manage .pth files

ZOPE

Goal of easy_install

Be to Python as yum or apt are to linux

- Automatic download of distributions and dependencies
- Don't make users think -- be easy
- Install eggs into Python library
- Install scripts into Python/system bin/scripts

easy_install meets these goals well.

ZOPE

I need something else

- Don't modify python installation
- Don't provide configuration at run time (no PYTHONPATH required)
- No accidental upgrades
- Custom script generation
- Greater control over eggs used
  - Specify specific versions
  - Newest of everything

ZOPE

Buildout

- Create an assembly of parts
- Control how each part is created
- Single specification of assembly, composed from multiple sources
- Support for change, through automated part uninstall and reinstall.

ZOPE

Buildout History

- Needed to assemble systems with multiple processes (databases, ZEO servers, app servers, etc.) over multiple machines
- Initial version(s) were make based
- make does more than we want and is a terrible scripting language
- Python prototype evolved over two years
- 1.0 version based on experience and leveraging eggs

ZOPE

Buildout and eggs

Buildout gives me the control
to get what I need from eggs!

ZOPE

Parts

- Can be almost anything:
  - programs
  - libraries
  - configuration files or changes to configuration files
  - etc.
- Name
- Recipe
- Configuration options

ZOPE

Recipes

- Implement parts
- Small Python classes or methods
- Different kinds:
  - install
  - uninstall (planned)
  - modified (planned)

ZOPE

Buildout is Egg-Centric

- Recipes managed as eggs
- Support for developing Python software using develop eggs
- APIs to support retrieval and installation of eggs for use by recipes

ZOPE

Configuration Database

- Each buildout has a configuration database.
- Based on Python ConfigParser format with extensions
- Configuration files can load other files
- Command-line options:
  `section:option=value`
- Variable substitutions

ZOPE

Sample Configuration

```
[buildout]
develop = mkdir
parts = data_dir
log-level = INFO
```

```
[data_dir]
recipe = mkdir
path = mystuff
```

ZOPE

Sample Recipe (in mkdir egg)

```
import logging, os

class Mkdir:

    def __init__(self, buildout, name, options):
        self.name = name
        self.options = options
        options['path'] = os.path.join(
            buildout['buildout']['directory'],
            options['path'])

    def install(self):
        path = self.options['path']
        if not os.path.isdir(path):
            logging.getLogger(self.name).info(
                'Creating directory %s', os.path.basename(path))
            os.mkdir(path)
        return path
```

ZOPE

Uninstallation

- The install method can return one or more paths.
- If the configuration changes, or if a part is no longer used, the paths returned in by the last part install are removed at the beginning of buildout processing

ZOPE

Real-world example

- Developing a new security-policy, zc.sharing, component for Zope 3
- Uses another component, zc.security, that is being developed developed at the same time
- Subversion project for zc.sharing with svn:external to zc.security
- buildout.cfg specified how to build out a zope install and zope instance using zc.security and it's dependencies

ZOPE

buildout.cfg (1)

```
[buildout]
develop = . zc.security

parts = zope3 data instance

find-links = http://download.zope.org/distribution/

[zope3]
recipe = zc.recipe.zope3checkout
url = svn://svn.zope.org/repos/main/Zope3/trunk

[data]
recipe = zc.recipe.filestorage
```

ZOPE

buildout.cfg (2)

```
[instance]
recipe = zc.recipe.zope3instance
database = data
user = jim:123
eggs = zc.sharing

zcml =
  zc.resourcelibrary zc.resourcelibrary-meta
  zc.sharing-overrides:configure.zcml zc.sharing-meta
  zc.sharing:privs.zcml
  zc.sharing:zope.manager-admin.zcml
  zc.security
  zc.table
  zope.app.securitypolicy-meta
  zope.app.twisted
  zope.app.authentication
```

ZOPE

Installing zc.buildout

- Use easy_install to install zc.buildout
  - You'll get a buildout script in your Python bin/ scripts directory
- Use the bootstrap script http://dev.zope.org/Buildout/bootstrap.py
  - Typically checked into your project
  - Downloads and installs setuptools and zc.buildout into your buildout
  - Get bin/buildout (and bin/py_zc.buildout, which is handy for running setup.py manually)

ZOPE

Status

- Under active development
- Ready for production
- Near term goals
  - Better error reporting
  - Windows support
  - egg extras
  - better control over egg versions

ZOPE

Questions?

http://dev.zope.org/Buildout

ZOPE