Eggs, Buildouts und andere Wunderlichkeiten

DZUG Workshop Saarbrücken
September 2008

Andreas Jung
www.zopyx.de, info@zopyx.de

Agenda

- Block 1 - Vom Package zum Egg
  - Einführung und Verwendung von **setuptools**
  - Einführung und Verwendung von **virtualenv**
- Block 2 - Buildout
  - Einführung und Verwendung von **zc.buildout**
  - Verwendung von **paster** + **ZopeSkel**

zopyx

Andreas Jung

- Autor diverser Zope und Plone Produkte
- Mitgründer, 2. Vorsitzender DZUG e.V.
- Zope 2 Release-Manager
- Gründer und Inhaber ZOPYX Ltd. & Co. KG
  - Software-Entwicklung und Consulting
  - Training & Schulungen

zopyx

Motivation

- Umgang mit Python-Modulen:
- Standardisierung
- Unifizierung
- Übernahme „best-practice“ Ansätzen

Ziele

- ~~Es geht um Python Programmierung~~
- Python-Code:
  - strukturieren & distributieren
  - pflegen
  - Package-Konventionen
- „The big picture“: Python-Packages in Buildouts
- Wie werden aus Eggs eine Sandbox?

zopyx

Teil 1

- Python Packages
- Setuptools
- Eggs

Was ist ein Python Package?

- eine sinnvolle und abgeschlossene Kapselung mehrerer Python Module innerhalb einer Verzeichnisstruktur

CVS:

```
pypimirror/__init__.py
pypimirror/mirror.py
pypimirror/util.py
pypimirror/logger.py
```

zopyx

setuptools

- Das **zentrale** Standardmodul in Python bzgl. Packaging, Distribution und Package-Management in Python
- mittlerweile Standard Python-Package (>= Python 2.5)
- nachinstallierbar für ältere Python Versionen
- stellt **easy_install** zur Verfügung
- Nachfolger von **distutils**
- Autor: Philipp Eby

zopyx

Installation

- Setuptools/easy_install
- Download: http://peak.telecommunity.com/dist/ez_setup.py
- python2.4 ez_setup.py

SVN Verzeichnisstruktur für Python Packages

- cool!
- sieht ja fast aus wie ein Python Package!
- Ablage in SVN!
- neu: **setup.py**

SVN:

```
z3c.pypimirror/trunk/setup.py
z3c.pypimirror/trunk/z3c/pypimirror/__init__.py
z3c.pypimirror/trunk/z3c/pypimirror/mirror.py
z3c.pypimirror/trunk/z3c/pypimirror/logger.py
z3c.pypimirror/trunk/z3c/pypimirror/util.py


z3c.pypimirror/tags/1.0.1/...
z3c.pypimirror/branches/release-may-2008/...
```

zopyx

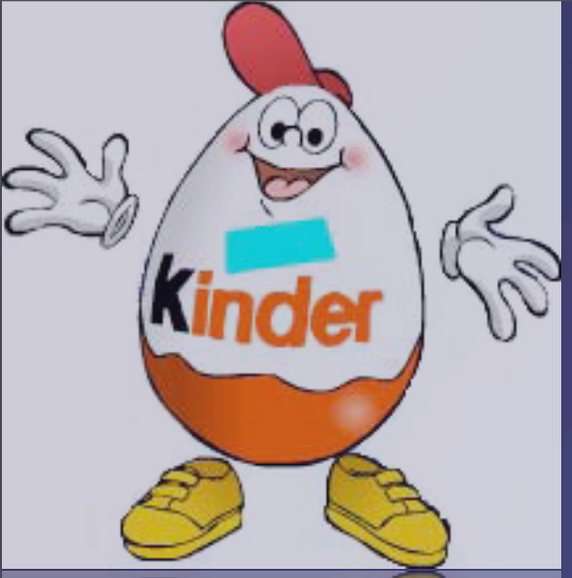
Was sind Python Eggs?

- vergleichbar mit Java JAR Dateien und Perl CPAN
- Python-Packages
- gezippt
- enthalten Metadaten
- enthalten Angaben zu Abhängigkeiten von anderen Packages
- Repository: PythonIndex http://pypi.python.org/pypi

zopyx

Package vs. Egg

- Eggs sind Hilfsmittel zur
  - Archivierung
  - Distributierung
- ...von Python Packages



setup.py eines Eggs

- Alle Metadaten sind in **setup.py/setup()** abgelegt
- Einige sind verantwortlich für:
  - Beschreibung
  - Installation
  - Archiverstellung
  - Definition von Abhängigkeiten
- Beispiel

zopyx

Namespace Packages

- eine Ebene mehr
- Namespaces zur Gliederung, Gruppierung und Branding
- Besonderheit: __init__.py
- plone.app.* zopyx.recipe.* plone.recipe.*, zope.*, zc.*, z3c.*, z3ext.*

```
z3c.pypimirror/trunk/setup.py
z3c.pypimirror/trunk/z3c/__init__.py
z3c.pypimirror/trunk/z3c/pypimirror/__init__.py
z3c.pypimirror/trunk/z3c/pypimirror/mirror.py
z3c.pypimirror/trunk/z3c/pypimirror/logger.py
z3c.pypimirror/trunk/z3c/pypimirror/util.py
```

zopyx

Namespace Packages (2)

```
z3c.pypimirror/trunk/setup.py
z3c.pypimirror/trunk/__init__.py
z3c.pypimirror/trunk/z3c/__init__.py
```

```
__import__('pkg_resources').declare_namespace(__name__)
```

```
...
namespace_packages=[,z3c‘],
...
```

Versionierungsschemata

- Unterstützt das übliche Versionierungsschemta:
  - X.Y.Z, X.Y.Za1, X.Y.Zb2, X.Y.Zrc3,
  - pre-Releases: alpha, beta, a, c, dev < final
  - post-Releases: alles > „final“
- Details: Link

```
>>> from pkg_resources import parse_version

>>> parse_version('1.9.a.dev') == parse_version('1.9a0dev')
True
>>> parse_version('2.1-rc2') < parse_version('2.1')
True
>>> parse_version('0.6a9dev-r41475') < parse_version('0.6a9')
True
```

zopyx

Package Abhängigkeiten Version-Pinning

z3c.pypimirror/trunk/setup.py...

```
...,
install_requires = ["OtherProject>=0.2a1.dev-r143,==dev",
                    „SQLAlchemy==0.3.9“,
                    „z3c.sqlalchemy“]
```

Package Abhängigkeiten (2)

- Abhängige Packages werde via easy_install nachinstalliert
- Vorsicht bei Packages, die Abhängigkeiten von zope.* haben (easy_install --no-deps)

Vom Package zum Egg

```
python2.4 setup.py <command1> <command2> ...
```

- <command>
  - „install" - Standard Python installation
  - „sdist" - Sourcecode Archiv
  - „bdist_egg" - Juhu, ein EGG
  - „upload" - Upload in einen Repositoryserver
  - „bdist_win" - für Windows
- **Livedemo** mit z3c.pypimirror

zopyx

PyPI - Python Index (aka Cheeseshop)

- zentrales Repository für die meisten Python-Packages
- Uploads werden zentral gehostet (extern nach Wahl)
- Single-Point-of-Failure
- easy_install sucht Packages per-default auf PyPI
- setuptools lädt Packages per-default auf PyPI hoch

zopyx

Entry Points

„Dynamic discovery of services and plugins"

- Registry für Dienste und Erweiterungen
- Stringifizierte Key-Value Paare
- Häufige Anwendungsfälle:
  - Generierung von Skripten bei Installation
  - Registrierung von Templates/Resourcen

Entry Points (2)

z3c.pypimirror/trunk/setup.py...

```
    entry_points = dict(console_scripts=[
        'pypimirror = z3c.pypimirror.mirror:run',
        ])
    )
```


zopyx

Generierung Skeletons

- Erzeugung der initialen Verzeichnisstruktur für neue Packages, Produkte etc. ist langweilig und fehlerträchtig -> Automatisierung und Standardisierung notwendig
- Lösung: **paster + ZopeSkel**
- **Zopeskel:** Satz von Templates für Packages, Zope-Produkte, Plone Buildouts

Generierung Skeletons (2)

- paster create --list-templates
- paster create -t basic_package
- paster create -t basic_namespace
- paster create -t nested_namespace

Die dunkle Seite der Eier

- Abhängigkeitsmanagement bedarf der Überarbeitung (Versionskonflikte)
- Installation aller Eggs in das (System)-Python
- Debugging von Eggs (ZIP-Archive) schwierig
- „site-packages“ als Egg-Klo

zopyx

- Eggs sind **ein Deploymentformat**
- Eggs sind **kein Distributionsformat**
- Source-Code (**sdist**) Upload bevorzugt gegenüber Egg Uploads (**bdist_egg**)
- http://philikon.wordpress.com/2008/06/

zopyx

virtualenv
„Virtual Python Environment Builder“

- System-weite Python Installationen:
  - Normalsterbliche dürfen keine Add-ons installieren - neue Add-Ons schwer zu testen
  - man will nicht die Python-Installation mit Add-ons zumüllen
  - Versionskonflikte
- virtualenv:
  - isolierte Python-Umgebung im Benutzerkontext
  - ideal für Prototyping und zum Rumspielen

zopyx

virtualenv (2)

- Installation:
  - easy_install virtualenv
  - source bin/activate
- generiert unter bin:
  - lokaler Python Interpreter **python**
  - lokales **easy_install**

virtualenv (3)

```
ajung@hweb1:/tmp: virtualenv test
New python executable in test/bin/python2.4
Installing setuptools.............done.

ajung@hweb1:/tmp: cd test/
ajung@hweb1:/tmp/test: source bin/activate

(test)ajung@hweb1:/tmp/test: bin/easy_install zopeskel
Searching for zopeskel
Reading http://pypi.python.org/simple/zopeskel/
...
Couldn't find index page for 'zopeskel' (maybe misspelled?)
Installed /tmp/test/lib/python2.4/site-packages/ZopeSkel-1.8-py2.4.egg

(test)ajung@hweb1:/tmp/test: bin/python2.4
Python 2.4.4 (#3, Jul 20 2007, 07:47:18)
[GCC 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

setuptools Development Mode

- Erlaubt den Test von eigenen Modulen in der Entwicklungsphase ohne explizite Installation (python setup.py install)
- `> python2.4 setup.py develop`
- Link zur setuptools Dokumentation

Fragen?

zopyx

Fragen?

Pause

Fortsetzung folgt

zopyx

Teil 2

- Einführung in zc.buildout
- **paster + ZopeSkel**

Was ist Buildout?

- PyPI: easy_install zc.buildout
- Autor: Jim Fulton
- **Reproducable Buildouts**
  - Beschreibung aller Aspekte in einer zentralen Datei **buildout.cfg**
    - Produkte, Module, Ports, etc
  - Kein manuelles Verfrickeln von Konfigurationen etc.
  - **buildout** starten -> Kaffee holen -> fertig

Philosophie von Buildout

- standardisierte und automatisierte Erstellung von Sandboxen in Zope und vereinfachtes Deployment
- „self-contained“
- reproduzierbar

Philosophie von Buildout (2)

- agonositsch bzgl. dem Verwendungszweck
- agonostisch bzgl. der Quellen der Bestandteile:
  - Eggs
  - Checkouts (SVN, CVS)
  - Downloads
  - ....

Entwickler

erstellt
buildout.cfg

übergibt
buildout.cfg

glücklicher Kunde

zopyx

Beispiele buildout.cfg/Recipe

- Link zu Edu-buildout.cfg
- Link zu plone.recipe.zope2install

Rezepte/Recipe

- Ein **Recipe** ist ein Python-Modul mit einer minimalen API (install(), update(), uninstall())
- **Recipes** werden als Eggs verwaltet/ distributiert
- **Recipes** implementieren die korrekte Installation/Update eines **Parts**
- Repository für Recipe: PyPI (nach „recipe“ suchen)

zopyx

Rezepte/Recipe (2) Anwendungsfälle

- Zope-Installation
- SVN Checkouts
- Download/Extraktion Archive (Sourcecode-Distros)
- Setup und Konfiguration von Zope Instanzen
- Frontend-Proxy Installation
- Datenbank-Installationen
- many more.....

zopyx

Rezepte/Recipe (3)

- Buildout erkennt Änderungen an der Konfiguration eines Parts (in buildout.cfg):
  -> Part wird **gelöscht und neugeneriert**
- Part wird aus buildout.cfg gelöscht:
  -> Part wird **gelöscht**
- Part wird in buildout.cfg neu eingetragen:
  -> Part wird **installiert** (install() des Recipe)
- Part wird in buildout.cfg nicht verändert:
  -> Part wird aktualisiert (update() des Recipe)

zopyx

Einige zc.buildout Optionen

```
[buildout]

newest = True | False

prefer-final = True | False

use-dependency-links = True | False

offline = True | False

download-cache = /home/ajung/myegg-cache
```

zopyx

Einige nützliche Buildout Optionen

- buildout -c someother.cfg
- non-newest, offline-mode:
  - buildout -No
- buildout --help

Der **extends** Mechanismus

Erweiterung einer Basiskonfiguration und Kaskadierung von Konfigurationen

base.cfg

```
[buildout]
parts = part1 part2
part3

[part1]
recipe =
option = a1 a2

[part2]
recipe =
option = b1 b2 b3 b4
```

configuration1.cfg

```
[buildout]
extends = base.cfg

[part1]
option += a3 a4

[part2]
option -= b1
option += xxx
```

paster + ZopeSkel

- Tools aus der Python/Plone Welt
- Generierung von Buildout-Skeletons auf Basis von Templates:
  - paster create --list-templates
  - paster create -t <template_name>
- verwendet Cheetah-Templates (*_tmpl Dateien)

Zope Produkte als Eggs?

- Zope Produkte sind nichts anderes als Namespace Packages im **Products** Namespace!
- Einziger Unterschied: **initialize()** beim Zope Startup
- Distributierbar als Eggs
- Installierbar als Egg (Buildout)
- in Zope >=2.8 verfügbar

zopyx

Zope Produkte als Eggs? (2)

```
[buildout]

eggs =

    Products.TextIndexNG3


[instance]

zcml =

    Products.TextIndexNG3
```

Version-Pinning

```
[buildout]

eggs =

    Products.TextIndexNG3==3.2.2
#     Products.TextIndexNG3==3.1.16
#     Products.TextIndexNG3>=3.1.0,<=3.1.9999
```


zopyx

Fragen?

zopyx

Fragen?

Vielen Dank für Ihre Aufmerksamkeit

Softwareentwicklung und Beratung für Python, Zope & Plone:

ZOPYX Ltd. & Co. KG
Charlottenstr. 37/1
72070 Tübingen
Tel. 07071/793376
www.zopyx.de
info@zopyx.de