CMFEditions

Versioning for Plone

Europython 2006, CERN, Geneva, Grégoire Weber

```
>>> pprint(document.__dict__) # green: retrieve, red: don't
   {
     '_Modify_portal_content_Permission': ('Manager', 'Owner'),
     '__ac_local_roles__': {'gregweb': ['Owner']},
     '_safety_belt': 'None',
     'creation_date': DateTime('2005/02/14 20:03 GMT+1'),
     'effective_date': None,
     'expiration_date': None,
     'modification_date': DateTime('2005/02/14 20:03 GMT+1'),
     'description': 'Description of Document',
     'id': 'index_html',
     'portal_type': 'Document',
     'rights': '',
     'text': 'Body of the document ...',
     'title': 'Title of Document',
     'workflow_history': {'plone_workflow': ({'action': None,
       'review_state': 'visible', 'actor': 'gregweb',
       'time': DateTime('2005/02/14 20:03 GMT+1')},)}
   }
```

Contents

Query – What do you expect from ... ?

(contributed by the audience)

... the ideal versioning *product*?

- Diff, retrieve fast
- Usable, simple for common users
- Audit trails (history)
- Purge support for unwanted versions

... the ideal versioning *framework*?

- Higly configurable,
- adaptable to use cases
- Easy to setup and integrate,
- pluggable

General Issues – Do we know ...?

- the future?
- all yours or your customers use cases today and in a year?
- the right balance between doing versioning:
  - correctly/perfectly and
  - in an intuitive kind of manner?

Sorry, but we don't! – Do you?

That's why CMFEditions ...

- out of the box functionality is *simple* and
- architecture is *higly extensible.*

What's CMFEditions?

- **CMFEditions adds versioning to Plone:**
  - propagandizes isolation of working copies and history
  - save current state for later retrieval
  - browse the history
  - revert to previous state
  - preview a previous state
  - version of folderish types and ATReferences
- **What you get with the upcoming version 1.0:**
  - a *product working out of the box* with a stock Plone 2.1.x site
  - a highly *extensible framework* for specific use cases:
    - save and retrieval interceptable
    - replaceable backend (version storage)

What not?

Version handling (CMFEditions), staging, multilanguage and wokflow handling are *orthogonal* functionalities. If well designed each dimension (functionality) can work on its own.

Version handling

- CMFEditions provides extensible repository functionality

not: Workflow

- workflows may control versioning (save the version on publish).

not: Staging

- staging may (or may not) be built on top of a versioning solution.

not: Multilanguage

- that's LinguaPlone for.

Proper integration of dimensions is still necessary.

Let's have a look at it! (Demo)

- history
- revert
- preview of old versions (retrieve)
- save
- configuration
- FAQ example (folderish type with tightly coupled childrens)
- silly retrieve modifier example (extensibility)

Part II: Dive into ...

Part II of the CMFEditons talk is about

- ? Goals the current release is based on
- ? Composition (assembling) and Decomposition (cutting) python objects

In Detail

- ? What the problematics are when versioning python objects
- ? How we achieved to solve the problematics
- ? The architecture of CMFEditions

Goals when Development Started

The Core has to ...

- be independent of how data is modeled
- be independent of how relations are implemented
- allow non-versioned objects and attributes

CMFEditions as a whole has to ...

- be easily adoptable to Use Cases without changing the core
- has an API that is as simple possible
- Scalable (BLOBs, speed)

Architecture and Design Goals

- componentized architecture (through CMF tools)
- minimizing the impact on existing components and Plone itself
- site should continue working after of CMFEditions beeing uninst.

What shall be versioned? (1 of 3)

```
>>> PrettyPrinter().pprint(document.__dict__) # shortened
{
  '_Modify_portal_content_Permission': ('Manager', 'Owner'),
  '__ac_local_roles__': {'gregweb': ['Owner']},
  '_safety_belt': 'None',
  'creation_date': DateTime('2005/02/14 20:03 GMT+1'),
  'effective_date': None,
  'expiration_date': None,
  'modification_date': DateTime('2005/02/14 20:03 GMT+1'),
  'description': 'Description of Document',
  'id': 'index_html', # this is the containers business
  'portal_type': 'Document',
  'rights': '',
  'text': 'Body of the Document ...',
  'title': 'Title of Document',
  'workflow_history': {'plone_workflow': ({'action': None,
    'review_state': 'visible', 'actor': 'gregweb',
    'time': DateTime('2005/02/14 20:03 GMT+1')},)}
}
```

What shall be versioned? (2 of 3)

What are we interested in getting back from the repo?

- title, description, body, ...

What are we definitively <u>not</u> interested in getting back?

- workflow stuff, effective date, id, savety belt, ...
- security: permissions, local roles

Where do we not know what to do (Use Case dependent)?

- portal_type (in most cases: <u>yes</u>)
- rights (they may have been changed site wide in between)

What shall be versioned? (2 of 3)

What are we interested in getting back from the repo?

? title, description, body, ...

What are we definitively <u>not</u> interested in getting back?

? workflow stuff, effective date, id, savety belt, ...

? security: permissions, local roles

Where do we not know what to do (Use Case dependent)?

? portal_type (in most cases: <u>yes</u>)

? rights (they may have been changed site wide in between)

$\Rightarrow$ **we only want back the content!**

What shall be versioned? (2 of 3)

What are we interested in getting back from the repo?

- title, description, body, ...

What are we definitively not interested in getting back?

- workflow stuff, effective date, id, savety belt, ...
- security: permissions, local roles

Where do we not know what to do (Use Case dependent)?

- portal_type (in most cases: yes)
- rights (they may have been changed site wide in between)

$\Rightarrow$ **we only want back the content! (usually)**

What shall be versioned? (3 of 3)

Conclusions

? What to get back from the repository depends on:

- the use cases
- the point in time (before or after a site redesign, etc.)
- other frameworks used

$\Rightarrow$ **nothing can be assumed!**

$\Rightarrow$ move use case dependent stuff to outside of the core

CMFEditions implementation

? **on save**: save everything that belongs to the object (interferable)

? **on retrieve**: selective (interferable)
$\Rightarrow$ only retrieve the **wanted** information

Still here?

Wanna like to ...

- hear more about internals?
  --> additional 3 slides: one of them with a **architecture layer diagram!**
- pass to questions and discussion?

Handling References (simpified)

Questions

- Shall we deep copy on save? --> No!
  (this would probably copy most parts of a site)
- What to do on retrieve? --> Hmmm?

What to do? Simplified:

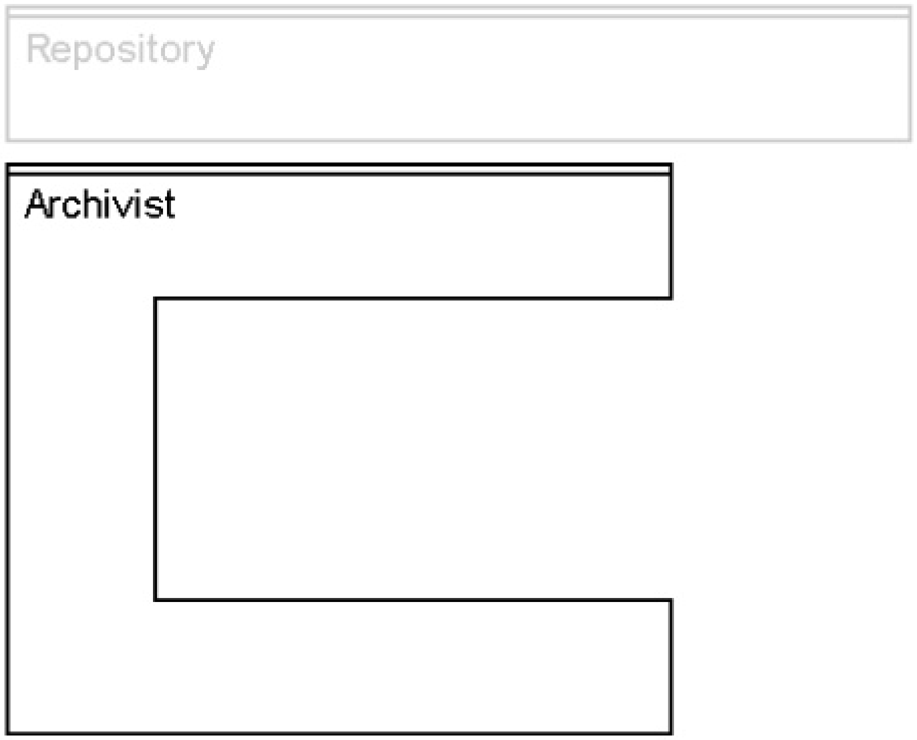
- **know the bounderies!**
- before save:
  - **decompose:** Replace the references (Zope ObjectManager, ATReferences, ...) by version aware references
- after retrieve:
  - **compose:** Rebuild references from the version aware refs

Architecture (layers)

Repository

← Repo: Use Cases (main API)

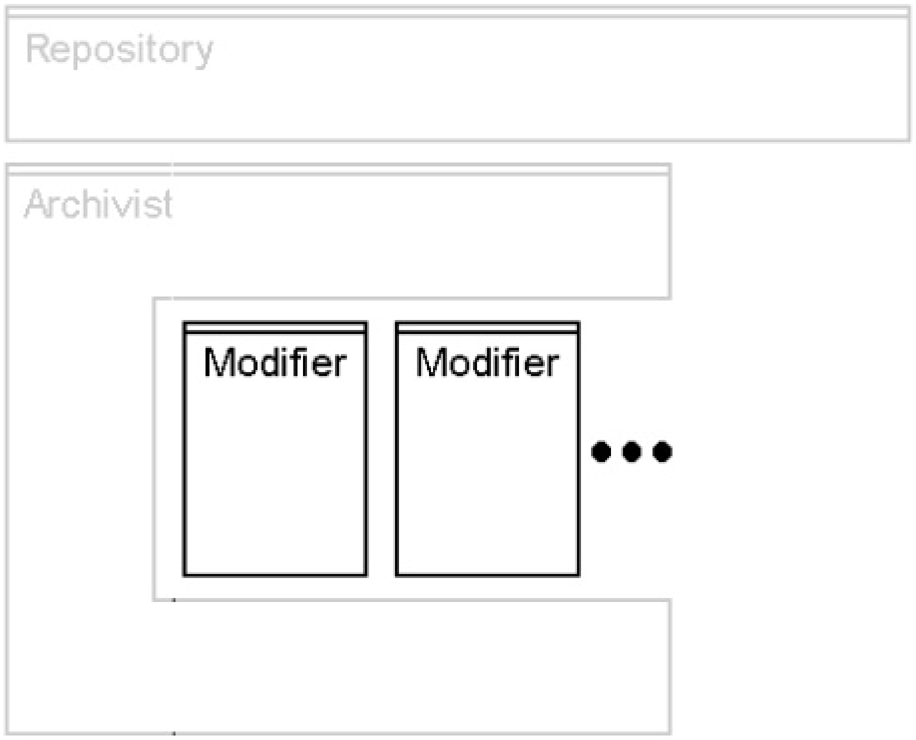
Architecture (layers)



← Repo: Use Cases (main API)

← Archivist: Python Object Model (how to copy)

Architecture (layers)

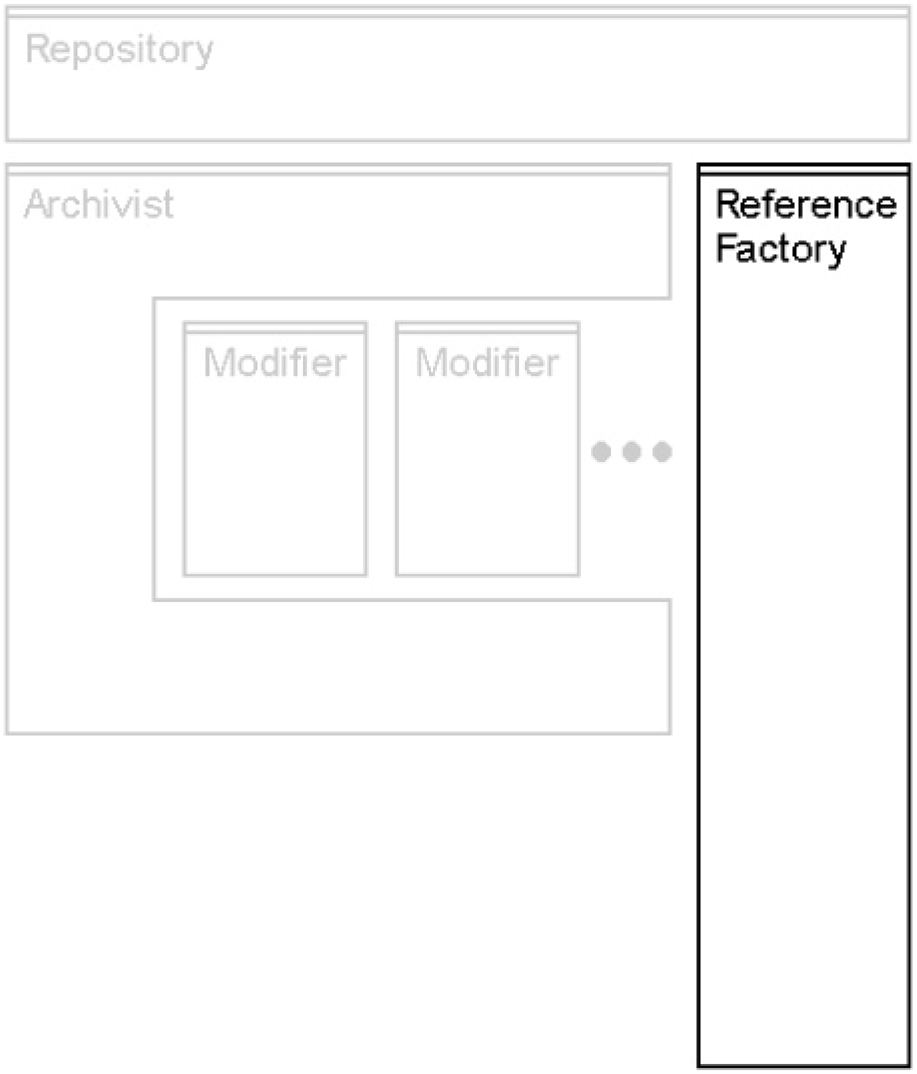


← Repo: Use Cases (main API)

← Archivist: Python Object Model (how to copy)

← Modifier: Semantics (what to copy)

Architecture (layers)



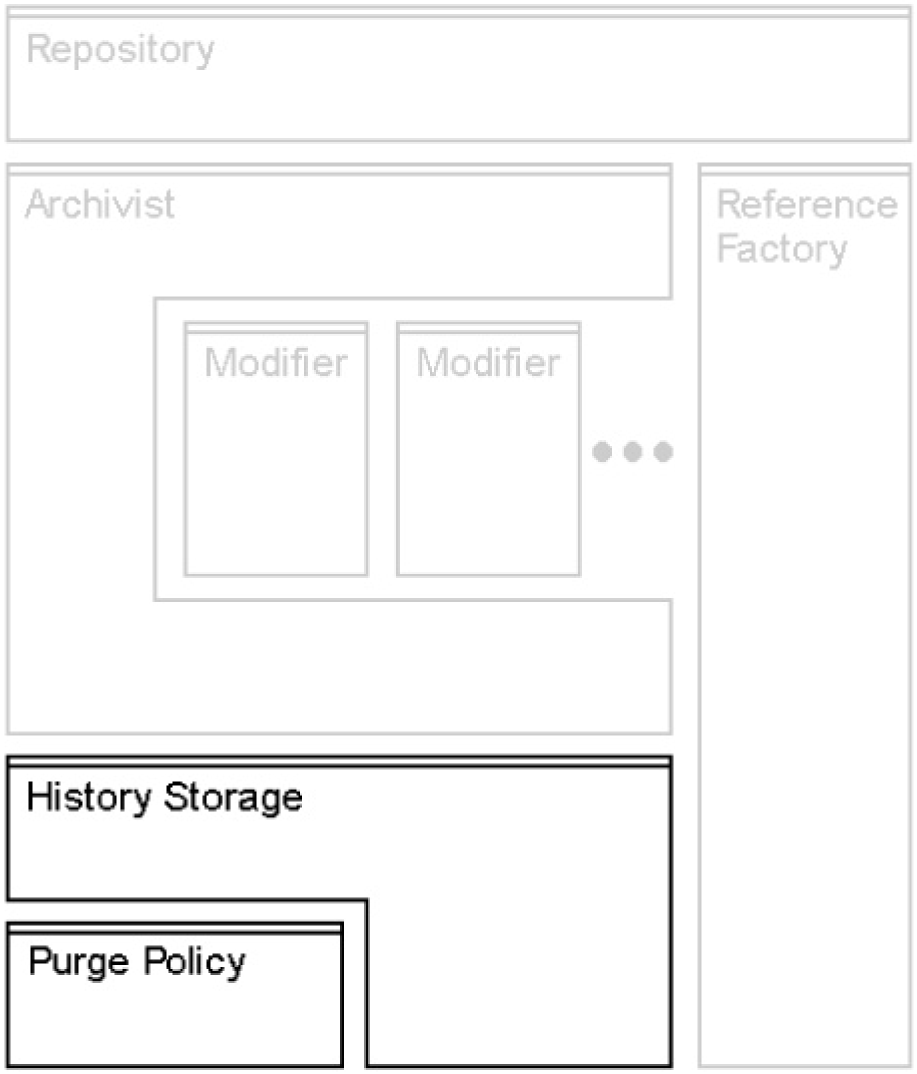
← Repo: Use Cases (main API)

← Archivist: Python Object Model (how to copy)

← Modifier: Semantics (what to copy)

← Factory: (rebuild references)

Architecture (layers)



← Repo: Use Cases (main API)

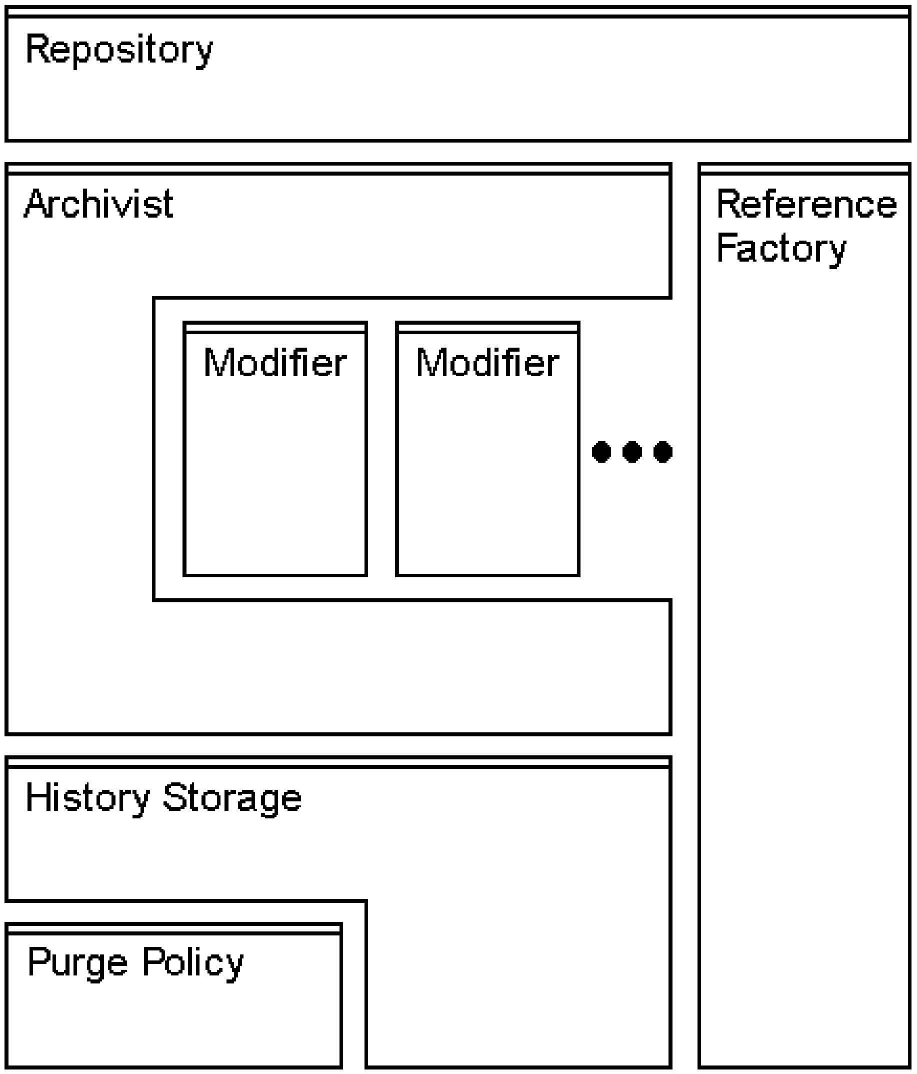
← Archivist: Python Object Model (how to copy)

← Modifier: Semantics (what to copy)

← Factory: (rebuild references)

← Storage (how to store)

Architecture (layers)



← Repo: Use Cases (main API)

← Archivist: Python Object Model (how to copy)

← Modifier: Semantics (what to copy)

← Factory: (rebuild references)

← Storage (how to store)

Architecture

Layered architecture with 6 CMF tools handling different aspects of versioning:

? Tool knowing about **use cases** (CopyModifyMergeRepository)

? Tool knowing **how to decompose and compose** references during copying (ArchivistTool, using the pickle protocol)

? Tool handling plugins that know **what to cut and assemble** (ModifierRegistryTool + Modifiers) Ü ***Extension Point***

? Tool knowing how to **store versions** of objects (StorageTool) Ü ***Extension Point***

? Tool knowing how to **construct references** (e.g. Childrens in a folder or AT References, ReferenceFactoriesTool)

? Tool knowing **what to purge** and **how to find a substitute** for a purged version on retrieve. Ü ***Extension Point***

Current State / Outlook

Current State

- 1.0beta1+, stabilizing, fixing remaining bugs
- expect 1.0rc1 and 1.0final really soon
- some instances with alphas in production (e.g. http://www.openplans.org/)

Ideas for Future Features

- replace ZopeVersionControl with simpler ZODB based storage
- extend version policies on repository layer
- AT Schema and ArchGenXML support
- non ZODB storages
- allow multiple checkouts of an object (as base for staging)
- branching (server or nomades mode?)

Credits

Contributors

- Alec Mitchel (alecm)
- Alberto Berti (azy)
- Duncan Booth (duncan)
- Francesco Ciriaci (ilbestio)
- Gregoire Weber (gregweb)
- Riccardo Lemmi (rlemmi)
- Rob Miller (rafrombrc)
- Sune Broendum Woeller (sunew)
- Sylvain Thenault (syt)
- Tomek Meka (tomek)
- Varun Rastogi (varun)
- Vincenzo Di Somma (vds)

Translations

- Danish: Anton Stonor
- French: Godefroid Chapelle (godchap)
- German: Gregoire Weber (gregweb)
- Polish: Piotr Furman

(by svn blame and cvs anno statistics)

Sponsors

- Oxfam GB: www.oxfam.org.uk
- ZEA Partners: www.zeapartners.org
- Musee de l'Afrique Centrale: www.africamuseum.be
- RedCOR: www.redcor.ch
- Zehnder Group: www.zehndergroup.com
- Reflab: www.reflab.com
- Metapensiero: www.metapensiero.it
- Incept: www.incept.ch

Interested in contributing, testing or just having questions?

- **later at EPC:**
  - eye in eye
  - contribute further translations (en, de, fr, pl, dk, ...)
- **later after EPC:**
  - mailing list: **collective-versioning@lists.sourceforge.net**
  - repo: https://svn.plone.org/svn/collective/CMFEditions
  - irc: freenode.net, #cmfeditions (rarely populated)
  - e-mail: gregweb @ incept.ch
- **now!**

Merci!

Grégoire Weber
gregweb@incept.ch
http://www.incept.ch

In.cept
Information Concepts