

Zope Development Roadmap
Jim Fulton, jim@zope.com

PyCon 2004



Zope 3 is a major rewrite of the Zope application server. The motivation and major features of Zope 3 will be presented, including an overview of the Zope component architecture. The plans for transitioning from Zope 2 to Zope 3 will be presented, including an overview of the upcoming Zope 2.8, Zope 2.9 and Zope X3 releases.

Jim Fulton is the chief architect of Zope and the Zope object database, and has been a Python contributor since 1994.



Zope2

- Started as a commercial application for dynamic web applications
- Original target audience: non programmers (e.g. Content managers)
- High degree of automation
- Highly OO
- Lots of inheritance
- Acquisition

ZOPE

Acquisition is a mechanism that allows sharing of information in containment hierarchies in much the same way that inheritance allows sharing of information in class hierarchies.



App Server for Content Management

Zope 2 emphasis changed:

- Application server for creating content-management applications
- Greater emphasis on developers
- Content-management framework (CMF)
 - Content types (portal types)
 - Separation of presentation and logic
 - More content-management-specific frameworks

ZOPE



Zope 2 Strengths

- Zope 2 makes it possible for non-technical people to create sophisticated dynamic web applications.
 - Many common tasks are automated
 - Through-the-web development
- Large user community
- Lots of add-on products and applications

ZOPE

Acquisition make sharing information, scripts, and configuration easy.

The security system made it easier to delegate responsibility and enabled through-the-web management.

Through-the-web management allows people to evolve web applications quickly, with immediate feedback.

High-level components provided a variety of valuable features, such as:

- Templating
- Relational database integration
- User management
- Indexing and searching
- Email
- ...



Zope 2 weaknesses

- The automation used to make Zope usable by non-technical people made it inscrutable to developers
 - “implicit” and “automated” are synonyms
 - Too willing to be implicit
- Developers weren't important early on
- Too much reliance on inheritance
- Too little documentation = all APIs public
- Z-shaped leaning curve

ZOPE

Zope started out as an application and became an application platform.

Things we did to make it easier for end-users made it hard for developers.

We also did a number of things that made our (developers) jobs easier in the short term, but that added to the overall load of “implicitness”.

Zope provides many features and has a correspondingly rich framework. It turned out that inheritance didn't scale very well to deal with this complexity.

Because we were behind the curve on documentation, many APIs that weren't meant to become public gained widespread use, creating backward-compatibility constraints.



Zope 3

Address specific issues in Zope 2

- Focus on developers
- Better leverage of Python
- Component architecture to facilitate reuse
- More explicit APIs
- Better integration of through-the-web and file-system development methods

ZOPE

Zope 3's emphasis has been on developers, especially Python developers

- Application code uses relatively simpler APIs
- No required mix-ins
- Goal to make it easier to reuse existing Python software in Zope
- Make it easier to use Zope software outside of Zope

Complexity is spread among separate components, rather than mix-in classes.

Acquisition is primarily through explicit location-aware component-lookup APIs.



Some Zope 3 Features

- Component architecture
- Software internationalization and localization
- Schemas
- Improved security model
- More configurable
- Events
- Workflow
- More explicit forms of acquisition
- Easier object references

ZOPE

Zope 3 has built-in support for I18n and L10n:

- All text is Unicode
- Translation framework
- Text extraction tools

Schemas are interfaces that support modeling information as well as behavior. They support automated data management features, most importantly, automated form generation.

The Zope 3 security protection scheme makes much less use of code manipulation, relying most heavily on security proxies. This provides a much simpler and robust protection scheme. It also guards against Trojan attacks, in which untrusted code tricks trusted code into performing otherwise disallowed operations.

Zope 3 has a much more pluggable security policy, allowing a variety of security models.

An XML-based configuration language that allows site-managers and deployment specialists to override product configuration without modifying product files.

An event system allows processing to be extended in ways that components alone don't.

An integrated workflow system combines event- and activity-based models and supports content-centric and process-centric workflow management.

Object location is modeled using explicit object parent attributes rather than context wrappers. This makes inter object references much simpler.



Bird's-eye view of components

- Components:
objects connected by interfaces
- Content components
- Adapters
- Presentation
- Utilities
- Services

ZOPE

What's important about components is that you can put them together. Interfaces are the mechanism for connecting things. They provide a way to reason about and automate system construction from components.

Content components model information in a system.

Adapters allow us to augment (or transform) object behavior by letting us compute new objects that provide additional interfaces for existing objects.

Presentation components are responsible for presenting objects to users.

Utilities are objects that provide functionality used by other objects. They are a bit like modules, except that they can be registered and looked up. A system may use alternate utility configuration, depending on configuration. Utilities may be named and used for specialized tools, like database connections and caches.

Services are special objects that provide a foundation for everything else. There aren't many of these. Examples include component-management services, like the adapter and utility services and application services, like the error logging and authentication services.



Zope 3 Development Strategy

- Mostly ground-up reimplementation
 - Unencumbered by backward compatibility
 - Focus on component architecture
 - Test driven
- Learn from Zope 2
- Open development with many contributors
 - Most contributors outside of ZC
 - Sprints
- Willing to make mistakes!

ZOPE



Zope 3 schedule

- Longer than expected
- For the better
- Dependent on volunteers
- Luxury of a great system in place: Zope 3
 - Don't need to rush Zope 2
- Currently in a period of consolidation

ZOPE

We've decided we're willing to wait a bit longer to have a very solid foundation for the future Zope 2 allows us this luxury.

For several months, we've been doing major restructuring and refactoring work on Zope 3 to provide as solid a foundation. There have been almost no new features, In fact, some features have been removed. The underlying architecture has become simpler, cleaner, faster and more robust.



Roadmap

- Zope X3
 - Not backward compatible
 - Platform for new applications
 - First release this July (?)
 - Already in production!
- Zope2 and Zope3 will converge
 - Already possible to use some Zope 3 in Zope 2
 - Future releases will make this easier
 - Increasing overlap

ZOPE

Someday, there'll be a Zope 3. Zope 3 will either be backward compatible, or will provide a practical transition strategy.



Zope X3.0

- Limited scope
 - Little TTW-development support
 - Enough to build file-system-based apps
- Goal:
let people work on Zope 3 in their day jobs!
- Possible schedule:
 - Alpha in April
 - Beta in May or June
 - Final in July (?)

ZOPE

There are two main motivations for Zope X3.0:

- People are already using Zope 3 in production. We need to give them a more stable platform.
- People want to use Zope 3 but don't feel they can use it in production until it's released. Zope X3.0 will allow many of these folks to use it in production

Of course, the schedule is only an estimate, but we're making a lot of progress.

.



Zope 2.8

- New-style ExtensionClass
- ZODB 3.3
 - Used by Zope 2 and Zope 3
 - MVCC
- Zope 3 interface support
- Foundation for greater Zope 2 and Zope 3 integration
- Q3 2004 (After X3.0)

ZOPE

This release represents a major architectural shift in Zope 2. Zope 2 will no longer use a specialized (advanced in it's day) class system. Zope 2 will use standard Python new-style classes with only minor enhancements.

New-style ExtensionClass, will allow Zope 2 applications to take advantage of many Python features that have been unavailable to old ExtensionClasses, such as descriptors and new operators added since (ahem) Python 1.3.

Zope 2.8 will adopt Zope 3's interface implementation, with a backward compatible interface to support existing Zope 2 interface support. This will make it much easier for those who don't want to wait until Zope 2.9 to use Zope 3 software in Zope 2.

You can help move this along by trying your software with the Zope CVS head and reporting (or fixing) bugs.



Zope 2.9

- Include many parts of Zope 3
- Support applications that run in both Zope 2 and Zope 3
- Possible features
 - Component architecture
 - Schemas
 - I18n and I10n support
 - New security protection system
 - ...

ZOPE

This release will represent a major narrowing of the differences between Zope 2 and Zope 3. It will provide a platform to start migrating existing applications to Zope 3.



You can make a difference

- How quickly Zope progresses depends on volunteers
- Many ways to contribute
 - Development (SW and Docs)
 - Testing
 - Feedback and suggestions (be nice :)
- Choose your platform: Zope 2 or Zope 3

ZOPE