
CHAPTER 0

PREFACE

The preface will be a brief introduction into Zope 3 and its capabilities as well as into Python , the programming language Zope is written in.

What is Zope?

What is Zope? While this sounds like a simple question that should be answered in a line or two, I often find myself in situations where I am unable to simply say: “It is an Open-Source Application Server.” or “It is a Content Management System.”. Both of these descriptions are true, but they are really putting a limit on Zope that simply does not exist. So before I will give my definition of Zope, let’s collect some of the solutions Zope has been used for. As mentioned above, many people use Zope as a Content Management System, which are usually Web-based (browser managed) systems. Basically the users can manage the content of a page through a set of Web forms, workflows and editing tools. However, there is an entirely different CMS genre, for which Zope also has been used. Other companies, such as struktur AG, used Zope successfully to interface with the XML Database Tamino (from software AG). The second common use is Zope as a Web-Application server, where it is used to build Web-based applications, such as online shops or project management tools. Of course, Zope is also suitable for regular Web sites.

And yet, there is a usage that we neglected so far. Zope can also be used as a reliable backend server managing the logistics of a company’s operations. In fact, *bluedynamics.com* in Austria built a logistic software based on Zope 2 ZClasses and a relational database that was able to handle hundreds of thousands transactions each day from taking credit card information and billing the customer up to ordering the products from the warehouse using XML-RPC. In my opinion this is the true strength of Zope, since it allows not only Web-familiar protocols to talk to, but also any other network protocol you can imagine. Zope 3, with its component architecture, accelerates even more in this area, since third party products can be easily plugged in or even replace some of the defaults. For example the Twisted framework can replace all of ZServer (the Zope Server components).

Now that we have seen some of the common and uncommon uses of Zope it might be possible to formulate a more formal definition of Zope, just in case you are being asked at one point. *Zope is an application and backend server framework that allows developers to quickly implement protocols, build applications (usually Web-based) and function as glue among other net-enabled services.*

Before Zope was developed, Zope Corporation was reviewing many possible programming languages to develop the framework, such as Java, C/C++, Perl and Python. After extensive research they found that only Python would give them the competitive advantage in comparison to the other large framework providers, such as IBM, BEA and others.

Powerful Python

Python is a high-level object-oriented scripting language producing – by design – clean code through mandatory indentation. While Perl is also an interpreted scripting language, it lacks the cleanliness and object-orientation of Python. Java, on the other hand, provides a nice object-oriented approach, but fails to provide powerful tools to build applications in a quick manner. So it is not surprising that Python is used in a wide variety of real world situations, like NASA, which uses Python to interpret their simulation data and connect various small C/C++ programs. Also, Mailman, the well-known mailing list manager, is being developed using Python. On the other hand, you have academics that use this easy-to-learn language for their introductory programming courses.

Since Python is such an integral part of the understanding of Zope, you should know it well. If you are looking for some introductory documentation, you should start with the tutorial that is available directly from the Python homepage <http://www.python.org/doc/current/tut/tut.html>. Also, there are a wide variety of books published by almost every publisher.

In the beginning there was...

Every time I am being asked to give a presentation or write a survey-like article about Zope, I feel the need to tell a little bit about its history, not only because it is a classic Open-Source story, but also because it gives some background on why the software behaves the way it does. So it should definitely not be missing here.

Before Zope was born, Zope Corporation (which was originally named Digital Creations) developed and distributed originally three separate products called Bobo, Principia and Aqueduct. Bobo was an object publisher written in Python, which allowed one to publish objects as pages on the web. It also served as object database and object request broker (ORB), converting URLs into object paths. Most of this base was implemented by Jim Fulton in 1996 after giving a frustrating Python CGI tutorial at the International Python Conference. Even though Bobo was licensed under some sort of “free” license, it was not totally Open-Source and Principia was the commercial big brother.

In 1998, Hadar Pedhazur, a well-known venture capitalist, convinced Digital Creations to open up their commercial products and publish them as Open Source under the name Zope. Zope stands for the “Z Object Publishing Environment”. The first Zope release was 1.7 in December 1998. Paul Everitt, former CEO, and all the other people at Zope Corporation converted from a product company into a successful consultant firm. Alone the story how Zope Corporation went from a proprietary, product-based to a service-based company is very interesting, but is up to someone else to tell.

In the summer of 1999, Zope Corporation published version 2.0, which will be the base for the stable release until Zope 3.0 will outdate it in the next few years. Zope gained a lot of popularity with the 2.x series; it is now included in all major Linux distributions and many books have been written about it. Originally I was going to write this book on the Zope 2.x API, but with the beginning of

the Zope 3.x development in late 2001, it seemed much more useful to do the documentation right this time and write an API book parallel to the development itself. In fact when these lines were originally written, there was no Zope Management Interface , and the initial security had just been recently implemented.

Zope 3 Components

Zope 3 will make use of many of the latest and hottest development patterns and technologies, and that with “a twist” as Jim Fulton likes to describe it. But Zope 3 also reuses some of the parts that were developed for previous versions. Users will be glad to find that Acquisition (but in a very different form) is available again as well as Zope Page Templates and the Document Template Markup Language - DTML (even though with less emphasis). Also, there is the consensus of a Zope Management Interface in Zope 3 again, but is completely developed from scratch in a modular fashion so that components cannot be only reused, but the entire GUI can be altered as desired.

But not only DTML, ZPT and Acquisition received a new face in Zope 3; external data handling has been also totally reworked to make external data play better together with the internal persistence framework, so that the system can take advantage of transactions, and event channels. Furthermore, the various external data sources are now handled much more generically and are therefore more transparent to the developer. But which external data sources are supported? By default Zope 3 comes with a database adaptor for Gadfly , but additional adapters for PostgreSQL and other databases already exist and many others will follow. Data sources that support XML-RPC, like the very scalable XML database Tamino, could also be seamlessly inserted. However, any other imaginable data source can be connected to Zope by developing a couple of Python modules, as described in various chapters.

During the last five years (the age of Zope 2) not only Zope was developed and improved, but also many third party products were written by members of the very active Zope community for their everyday need. These products range from Hot Fixes, Database Adaptors and Zope objects to a wide range of end user software, such as e-commerce, content management and e-learning systems. However, some of these products turned out to be generically very useful to a wide variety of people; actually, they are so useful, that they were incorporated into the Zope 3 core. The prime examples are the two internationalization and localization tools Localizer (by Juan David Ibáñez Palomar) and ZBabel (by me), whose existence shaped the implementation of the internationalization and localization support Zope 3 significantly. Another great product that made it into the Zope 3 core was originally written by Martijn Faassen and is called Formulator. Formulator allows the developer to define fields (representing some meta-data of a piece of content) that represent data on the one side and HTML fields on the other. One can then combine fields to a form and have it displayed on the Web. The second great feature Formulator came with was the Validator, which validated user-entered data on the server side. Formulator’s concepts were modularized into schemas and forms/widgets and incorporated in Zope 3.

Altogether, the framework is much cleaner now (and more pythonic) and features that failed to make it into the Zope 2 core were incorporated.

Goals of this book

The main target audience for this book are developers that would like to develop on the Zope 3 framework itself; these are referred to as Zope developers in this book. But also Python programmers will find many of the chapters interesting, since they introduce concepts that could be used in other Python applications as well. Python programmers could also use this book as an introduction to Zope.

In general the chapters have been arranged in a way so that the Zope 3 structure itself could be easily understood. The book starts out by getting you setup, so that you can evaluate and develop with Zope 3. The second part of the book consists of chapters that are meant as introductions to various important concepts of Zope 3. If you are a hands-on developer like me, you might want to skip this part until you have done some development. The third and fourth part are the heart of the book, since a new content component with many features is developed over a course of 12 chapters. Once you understand how to develop content components, part five has a set of chapters that introduce other components that might be important for your projects. The fifth part is intended for people that wish to use Zope technologies outside of Zope 3. The emphasis on testing is one of the most important philosophical transitions the Zope 3 development team has undergone. Thus the last chapter is dedicated to various ways to write tests.

Last but not least this book should encourage you to start helping us to develop Zope 3. This could be in the form of enhancing the Zope 3 core itself or by developing third party products, reaching from new content objects to entire applications, such as an e-commerce system. This book covers all the modules and packages required for you to start developing.