Zope 3 Knowledge Base

Sreeram R Nudurupati

Gulf Data Systems

raghav.sreeram@gmail.com

Abstract

The motivation behind this work is to document the process of installing Zope3 on a Fedora Core 5 system and building a knowledge base in Zope3. The paper explains the changes from Zope2 to Zope3 and migration issues. The paper also documents the entire process required to build the knowledge base and to build a search engine to search through it. The document extensively makes use of UML wherever necessary to represent the various software processes.

Introduction

The document first introduces, Zope3 and explains the differences from zope2, then it explains the detailed Zope3 installation instructions, followed by details for building the knowledge base. The content is organized as follows:

1. Zope2 Vs Zope3
2. Zope3 Setup and Installation Instructions
3. Zope3 Advantages
4. Knowledge Base
5. Software Process UML Representation

Zope2 Vs Zope3

Zope in general stands for “The Z Object Publishing Environment”. Till Zope2, Zope was merely a ‘web application server’ and a ‘content management system’. Zope2 was just like any other application server simply gluing together a web server, a database management system and a scripting language. Though Zope2 had the advantages of an object database and an excellent object – oriented scripting language Python[1]. But Zope2 lacked a any sort of underlying architecture. Thus application development in Zope2 was more or less procedural and haphazard.

Starting from Zope3, Zope has strong architecture called the ‘Component Architecture’[2], and application development on Zope3 is completely object oriented. The Component Architecture makes applications in Zope3 structured and the components are reusable. Also in Zope3 applications the three-layered structure of programming is followed i.e. the database, business and presentation layers are distinctly separated. All these features render Zope3 to be an excellent platform for development of complex systems such as the “Knowledge Base”.

Migration from Zope2 to Zope3 might be somewhat be trying and frustrating. Zope2 was heavily dependent on ZMI and ZPT and DTML and application development was through the web. But Zope3 application development entirely depends on Python. All the classes and Interfaces are designed and built using Python. All the business logic is written in Python and ZPT, HTML and DTML are used only for presentation purposes.

[1] http://www.python.org
[2] Zope 3 Developer’s HandBook by Stephan Richter

Zope3 Setup and Installation Instructions [3]

This part of the document explains the basic steps of downloading and installing Zope3 on a Fedora Core 5 machine.

1. Download the source version of Zope 3.2 from the link:

   http://www.zope.org/Products/Zope3/3.2.1/Zope-3.2.1.tgz

2. Unpack the **'.tgz'** file using the command:

   # ***gzip Zope-3.x.x.tgz***

3. The result of the above action is a file named *Zope-3.x.x.tar*. Unpack the file using the command:

   # tar Zope-3.x.x.tar

   Which creates a folder with the name *Zope-3.x.x.*

4. Now enter the folder using the command:

   # Cd Zope-3.x.x

5. Type the series of commands:

   # ***./configure***

   # make

   # make install

6. If no error message is shown, Zope3 is successfully installed. Now we need to create an instance home for Zope3 to start the Zserver.

[3] http://www.zope.org/Products/Zope3/3.2.1

7. To create an instance home in '/disk/' partition, enter the following commands:

   # Cd /usr/local/Zope3.x.x/bin

   # ./mkzopeinstance /disk/zopeinstance

   Unername: *admin*

   Password: *xxxxx*

8. Now the Zope3 instance home is created in *'/disk/zopeinstance'*. The next step is to start Zope3, which can be done as:

   # Cd /disk/zopeinstance/bin

   # ./runzope

9. If no error is shown then the Zope3 server should be up and running on the localhost on port 8080.

10. To change the server preferences, one can change the settings in the */disk/zopeinstance/etc/zope.conf* file.

11. Now the Zope3 management interface can be accessed through the web browser as:

    http://serveripaddress:8080/manage/

12. Login with the username and password provided while making the instance home.

13. For more details about installing Zope3 and troubleshooting, visit the link on the Zope3 website:

    http://www.zope.org/Products/Zope3/3.2.1/README.txt

Setting up the Knowledge Base

Zope3 just as Zope2 contains all the infrastructures to setup the knowledge base and to make it searchable. The only problem here is that Zope3 does not index PDF files by default and it being relatively new no custom built plug-ins exist yet. So the following steps need to be carried out to build a searchable knowledge base of PDF files:

Considering a plain text file, this is how the search mechanism works:

- Every thing in Zope is an object.
- A text file is a subclass of a generic object *File*.
- The text file has an interface called *IText* which in turn implements an interface called *IsearchableText.*
- *IsearchableText* has a method called *getSource* which gets fulltext of the object implementing it.
- Thus whenever a text object is uploaded into the Zope object base, and *Catalog* with a *TextIndex* already exists, the *IsearchableText's getSource* is invoked and the fulltext is passed on to the *Catalog* to be indexed.

Now, the problem with PDF files is:

- A PDF does not have a *PDF* subclass for the generic object *File.*
- So a PDF is still treated as *File,* which does not implement an *IsearchableText* interface.
- Thus an adapter called an *SearchableTextAdapter* with a method called *getSearchableText* had to be written which links an *PDF* to the *IsearchableText,* shown graphically in *Figure1*.

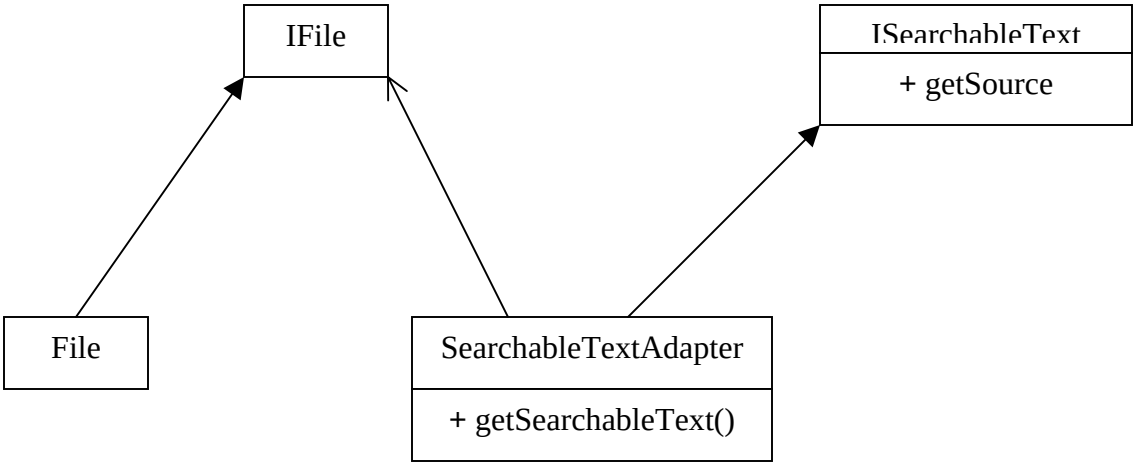

Figure1.

- The code for the module *adapter.py* is presented in Appendix – A.

Once all the links are in place, every time a PDF file is uploaded, the full text is extracted and indexed, provided a *Catalog* with a *TextIndex* already exists.

Implementing a Catalog:

- Open the Zope Management Interface, login as the manager then click on *[top]*.
- Click on *Manage Site* → *default.*
- Add an unnamed *Unique Id Utility i.e while adding it; do not provide any kind of names.*
- Then click add a *Catalog,* apply without providing a name, then click register, then provide a name and finally add, make sure the interface is set to: *zope.app.catalog.interfaces.Icatalog* access is set to *Public*.
- Now click on the newly added *Catalog*, from the left side panel add a *Text Index*, provide it with an interface of *zope.index.text.interfaces.IsearchableText,* set field name to *getSearchableText* and make sure that the field is callable by enabling on the radio button.

- Now copy the *adapter.py* the location: */usr/local/Zope3.x.x/lib/python/zope/app/file/,* and edit the *configure.zcml* in the same folder to add the following entry:

```
<adapter
    factory=".adapter.MyObjectToSearchable"
/>
```

The knowledge base is ready and every time a new PDF is added, its full text is indexed. The only thing now required is a search interface and a presentation layer object to display the results.

To add a search ineterface, the following files are required are: *__init__.py, configure.zcml, edit.pt, search.py, search.pt, form.pt.*

To add the search interface, the following steps are required:

- Create a folder called search in the directory */usr/local/Zope3.x.x/lib/python/.*
- Copy the files *__init__.py, edit.pt, search.py, search.pt, form.pt* and *configure.zcml* to the newly created folder – search.
- Create a file called *search-configure.zcml* with the content *<include package="search"/>*.
- Copy the file to */../zopeinstance/etc/package-includes/.*
- Now login in to the *Zope Management Interface* and look at the left side panel for *Search Interface.*
- Click on it to add a new search interface it presents a search form and the corresponding results.
- The code required for all the files mentioned is presented in Appendix – A.

Appendix – A

```
"""
adapter.py
      The adapter is helpful in making a gereric file object searchable
TODO:
      Thougn the adapter adapts a text file object to be searchable but
the codecs fail to decode objects like PDF or Word documents.
Author:
      Sreeram Nudurupati
      05/10/2006
"""
from zope.index.text.interfaces import ISearchableText
from zope.component import adapts
from zope.interface import implements
from zope.app.file.interfaces import IFile
import tempfile
import os
from string import split, join, lstrip

class SearchableTextAdapter:
      """
            Adapts the IFile interface to the ISearchableText interface
      """
      implements(ISearchableText)
      adapts(IFile)

      def __init__(self, context):
            self.context = context

      def getSearchableText(self):
            open('\root\out.txt', 'w').write(self.context.contentType)
            tmp_name = tempfile.mktemp()
            open(tmp_name, 'w').write(self.context.data)
            text = os.popen('pdftotext %s -' % tmp_name).read()
            return text
```

```
##############################################################################
#######
#
# This software is subject to the provisions of the Zope Public
License,
# Version 2.0 (ZPL).  A copy of the ZPL should accompany this
distribution.
# THIS SOFTWARE IS PROVIDED "AS IS" AND ANY AND ALL EXPRESS OR IMPLIED
# WARRANTIES ARE DISCLAIMED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
# WARRANTIES OF TITLE, MERCHANTABILITY, AGAINST INFRINGEMENT, AND
FITNESS
# FOR A PARTICULAR PURPOSE.
#
# AUTHOR: Sreeram Nudurupati
#
##############################################################################
#######
"""Search Interface

$Id: interfaces.py 18 May,2006 10:07 AM $
The purpose of the module is to provide a uniform search interface for
Catalog in zope3.
"""
from zope.interface import Interface, implements
from zope.schema import TextLine
from zope.schema import getFieldNames
from zope.app.catalog.interfaces import ICatalog
from zope.app import zapi
from zope.formlib import form
from persistent import Persistent

class ISearchInterface(Interface):
      """Interface for SearchInterface"""
      txt = TextLine(
            title=u"Text",
            description=u"Name of keyword",
            required= True)


class SearchInterface(Persistent):
      """Class template for the menu item Search Interface in ZMI"""
      implements(ISearchInterface)
      txt = u"a*"



class SearchResults:
    """Class template for displaying queried results -
Search.html/Search.pt"""

    __used_for__ = ICatalog

    def __init__(self, context, request):
        self.context = context
        self.request = request

    def Results(self):
```

```
          """Actual method which gets keyword from Index.html and then
queries the caalog and returns results as a list to be collected bu
Search.pt"""
          self.context.txt = self.request['keyword']
          sm = zapi.traverse(self.context, '/++etc++site')
          catalog = sm['default']['Catalog']
          if(catalog):
                results =
catalog.searchResults(TextIndex=self.context.txt)
                return [result.__name__ for result in results]
          else:
                return u'Results not found'

class EditView:
                """Class template for getting user search string -
Index.html/Edit.pt"""
      zombie = u"Zope3"
```

“**Configure.zcml**”

```
<configure
      xmlns="http://namespaces.zope.org/zope"
      xmlns:browser="http://namespaces.zope.org/browser"
      i18n_domain="zope"
      >
  <browser:addMenuItem
      class=".search.SearchInterface"
      title="Search Interface"
      permission="zope.ManageContent"
      />
  <browser:page
      for=".search.SearchInterface"
      name="search.html"
      class=".search.SearchResults"
      permission="zope.Public"
      template="search.pt"
      />
  <browser:page
      for=".search.SearchInterface"
      name="index.html"
      class=".search.EditView"
      permission="zope.Public"
      template="edit.pt"
      />

  <content class=".search.SearchInterface">
      <require
            permission="zope.Public"
            interface=".search.ISearchInterface"
            />
      <require
            permission="zope.ManageContent"
            set_schema=".search.ISearchInterface"
            />
  </content>

  </configure>
```

“Form.pt”

```
<html metal:use-macro="context/@@standard_macros/view">
<body>
<div metal:fill-slot="body" tal:content="view">
</div>
</body>
</html>
```

“Edit.pt”

```
<html metal:use-macro="context/@@standard_macros/view">
<body>
<div metal:fill-slot="body">
<form method="post" action="search.html">
<input type="text" value="" name="keyword"
           i18n:attributes="value reindex-button"/>
    <input type="submit" value="Search"
           i18n:attributes="value reindex-button"/>
</form>
</div>
<body>
</html>
```

“Search.pt”

```
<html metal:use-macro="context/@@standard_macros/view"
    i18n:domain="zope">
<script language=javascript>

</script>
<body>
<div metal:fill-slot="body">
<h2 i18n:translate="">Search Interface</h2>

<table class="listing" summary="Indexes">
   <tr><th i18n:translate="">Search Results</th>
   </tr>
<tr tal:repeat="result view/Results">
<td tal:content="result">foo</td>
</tr>
</table>

<div class="bug" i18n:translate="">
  These are all the contants present in the various indexes present in
the Catalog.
</div>
<form method="post" action="index.html">
    <input type="text" value="" name="txt">
    <input type="submit" value="Index"
           i18n:attributes="value reindex-button"/>
</div>
</body>
</html>
```