Comment about Programming ................... 3
Containers ........................................... 7
Events ............................................... 13

Copyright ©, All rights reserved. 2005 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.
References

Web Component Development with Zope 3, von Weiter-shausen
Comment about Programming
Coding - No Libraries

Write all code from scratch
You are in complete control
Only need to know the language
Slow as you write all code from scratch

```python
primeCandidate = someInteger
possibleFactor = primeCandidate /2
while possibleFactor > 1:
    if primeCandidate % possibleFactor == 0:
        print primeCandidate, 'has factor', possibleFactor
        break
    possibleFactor = possibleFactor -1
else:
    print primeCandidate, 'is prime'
```
Direct Coding with Libraries

Use language and libraries
Faster development using libraries
Still mainly in control
Need to know about libraries (How?)
What happens if library does not
  • Perform action you need
  • Does something you don't want (Perl die)

```python
import urllib2

webPage = urllib2.urlopen('http://www.eli.sdsu.edu')

print webPage.info() #print http headers
for line in webPage:
    print line,
```
Coding with a Framework

Code extends framework

Framework determines

- What can be done
- How things are done

Framework can save you a lot of time

Need to know what Framework allows before starting

```java
import java.awt.*;
class HelloApplication extends Frame {
    public void paint( Graphics display ) {
        int startX = 30;
        int startY = 40;
        display.drawString( "Hello World", startX, startY );
    }
}
```
Coding with a Complex Framework
Zope

• Interfaces
• ZCML
• TAL
• METAL
• Events
• Security
• Adapters
• Services
• i18n
• Metadata

One starts working in a complex system
Takes long time to understand the system
Steep learning curve

Issues above are common in professional programming jobs
Containers

Zope containers

• Contain things
• Have a name
• Have a reference to their container
• Can restrict what they contain

Example - Container that only Holds Books
from zope.schema import Field
from zope.app.container.interfaces import IContainer, IContained
from zope.app.container.constraints import ContainerTypesConstraint
from zope.app.container.constraints import ItemTypePrecondition

class IBookContainer(IContainer):
    def __setitem__(name, object):
        """Add a Book""

        __setitem__.precondition = ItemTypePrecondition(IBook)

class IBookContained(IContained):
    __parent__ = Field(
        constraint = ContainerTypesConstraint(IBookContainer) )
from zope.interface import implements
from zope.app.container.btree import BTreeContainer
from zope.app.container.contained import NameChooser
from zope.app.exception.interfaces import UserError
from bookstore.interfaces import IBookContainer

class BookFolder(BTreeContainer):
    implements(IBookContainer)

class BookNameChooser(NameChooser):
    def checkName(self, name, object):
        if name != object.title:
            raise UserError(u"Given name & Book title do not match!")
        return \
        super(BookNameChooser, self).checkName(name, object)

    def chooseName(self, self, name, object):
        name = object.title
        print object
        print object.title
        self.checkName(name, object)
        return name
from persistent import Persistent
from zope.interface import implements
from zope.schema.fieldproperty import FieldProperty
from bookstore.interfaces import IBook, IBookContained
from datetime import date

class Book(Persistent):
    implements(IBook, IBookContained)

    __name__ = __parent__ = None

title = FieldProperty(IBook['title'])
author = FieldProperty(IBook['author'])
date = FieldProperty(IBook['date'])

def asString(self):
    return self.title + ';' + self.author + ';' + str(self.date)

def __fromString(main, string):
    book = Book()
    attributes = string.split(';')
    book.title = unicode(attributes[0])
    book.author = unicode(attributes[1])
    dateStringParts = attributes[2].split('-')
    dateParts = map(int, dateStringParts)
    book.date = date(*dateParts)
    return book

.fromString = classmethod(__fromString)
bookstore/configure.zcml

<configure xmlns="http://namespaces.zope.org/zope">

  <interface
      interface=".interfaces.IBook"
      type="zope.app.content.interfaces.IContentType"
  />

  <!-- Folder -->
  <content class=".folder.BookFolder">
    <implements
        interface="zope.app.annotation.interfaces.IAttributeAnnotatable"
        zope.app.container.interfaces.IContainerNamesContainer"
    />
    <require
        permission="zope.View"
        interface="zope.app.container.interfaces.IReadContainer"
    />
    <require
        permission="zope.ManageContent"
        interface="zope.app.container.interfaces.IWriteContainer"
    />
  </content>

  <adapter
      for=".interfaces.IBookContainer"
      provides="zope.app.container.interfaces.INameChooser"
      factory=".folder.BookNameChooser"
  />

  <include package=".browser" />
  <include package=".text" />

</configure>
bookstore/browser/configure.zcml

<configure
    xmlns="http://namespaces.zope.org/zope"
    xmlns:browser="http://namespaces.zope.org/browser"
>
    <browser:menu
        id="alternate_views"
        title="Menu showing different ways to view an object"
    />

    <browser:addform
        schema="bookstore.interfaces.IBook"
        content_factory="bookstore.book.Book"
        label="Add a Book"
        name="AddBook.html"
        permission="zope.ManageContent"
        set_before_add="title"
    />

    <browser:addMenuItem
        title="Book Folder"
        class="bookstore.folder.BookFolder"
        permission="zope.View"
    />

    <browser:containerViews
        for="bookstore.interfaces.IBookContainer"
        contents="zope.ManageContent"
        index="zope.View"
        add="zope.ManageContent"
    />

    <browser:resource name="bookstore.css" file="bookstore.css" />
</configure>
Events

Zope supports events

Code can be registered to know when events occur
Code is called when the event occurs
Code can trigger an event

Zope Events

<table>
<thead>
<tr>
<th>IObjectCreatedEvent</th>
<th>IObjectMovedEvent</th>
<th>IBeforeTraversalEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>IObjectCopiedEvent</td>
<td>IObjectAddedEvent</td>
<td>IEndRequestEvent</td>
</tr>
<tr>
<td>IObjectModifedEvent</td>
<td>IObjectRemovedEvent</td>
<td>IMailEvent</td>
</tr>
<tr>
<td>IObjectAnnotationsModifiedEvent</td>
<td>IDatabaseOpenedEvent</td>
<td>IMailSentEvent</td>
</tr>
<tr>
<td>IObjectContentModifiedEvent</td>
<td>IProcessStartingEvent</td>
<td>IMailErrorEvent</td>
</tr>
</tbody>
</table>
Sending an Event

```python
def rate(self, rating):
    ratings = self.mapping['ratings']
    ratings.append(float(rating))
    self.mapping['average'] = sum(ratings)/len(ratings)
    notify(ObjectAnnotationsModifiedEvent(self.context))
```
import email.Charset
email.Charset.add_charset('utf-8', email.Charset.SHORTEST, None, None)

from datetime import datetime
from email.MIMEText import MIMEText
from zope.app import zapi
from zope.app.mail.interfaces import IMailDelivery

def notifyAdded(book, event):
    return emailNotifications(book)

def emailNotifications(book):
    subject = "%s' was added" % {'name': book.title}
    message = MIMEText(book.asString().encode('utf-8'), 'plain', 'utf-8')
    message['Subject'] = subject
    message['From'] = 'whitney@rohan.sdsu.edu'
    message['To'] = 'whitney@cs.sdsu.edu'
    message['Date'] = datetime.now().strftime('%a, %d %b %Y %H:%M:%S %z')
    mailer = zapi.getUtility(IMailDelivery, 'bookstore-delivery')
    mailer.send("whitney@rohan.sdsu.edu",
                ["whitney@cs.sdsu.edu"],
                message.as_string())
bookstore/mail/configure.zcml

```xml
<configure
    xmlns="http://namespaces.zope.org/zope"
    xmlns:mail="http://namespaces.zope.org/mail"
>

    <mail:smtpMailer
        name="bookstore-mailer"
        hostname="cs.sdsu.edu"
        port="8025"
    />

    <mail:queuedDelivery
        name="bookstore-delivery"
        permission="zope.SendMail"
        queuePath="mail-queue"
        mailer="bookstore-mailer"
    />

    <subscriber
        for="bookstore.interfaces.IBook
            zope.app.container.interfaces.IObjectAddedEvent"
        factory=".book.notifyAdded"
    />

</configure>
```
bookstore/browser/configure.zcml

```xml
<browser:addform
  schema="bookstore.interfaces.IBook"
  content_factory="bookstore.book.Book"
  label="Add a Book"
  name="AddBook.html"
  permission="zope.ManageContent"
  set_before_add="title author date"
/>
```

Added to bookstore/configure.zcml

```xml
<include package=".mail" />
```