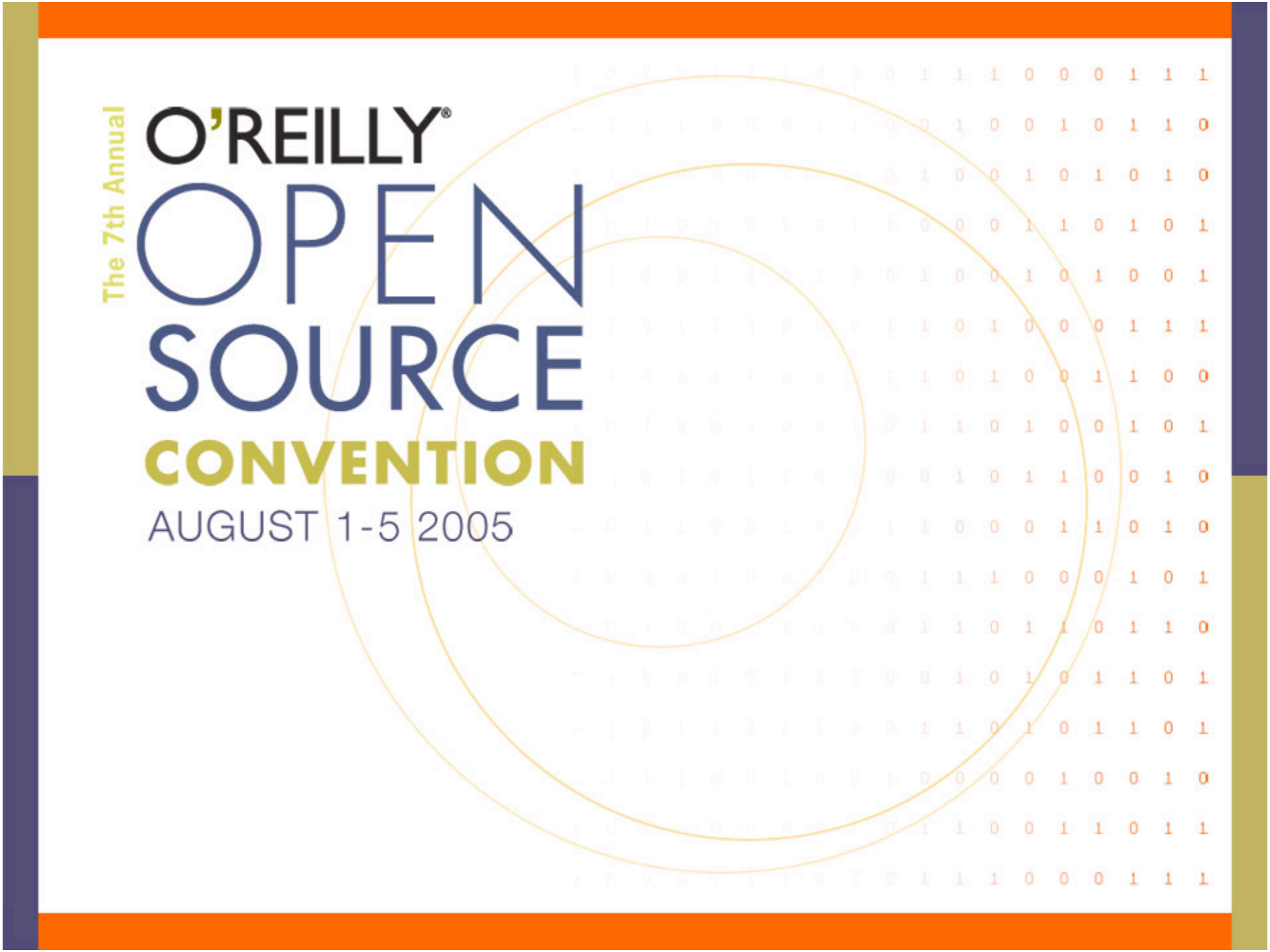

The 7th Annual
O'REILLY®
OPEN
SOURCE
CONVENTION
AUGUST 1-5 2005

Building Your Own Chandler Parcel

Ted Leung
Open Source Applications Foundation

O'Reilly Open Source Convention
August 1 - 5, 2005

Chandler

- Personal Information Manager
- Next release in Fall 2005
  - Focus on Calendar

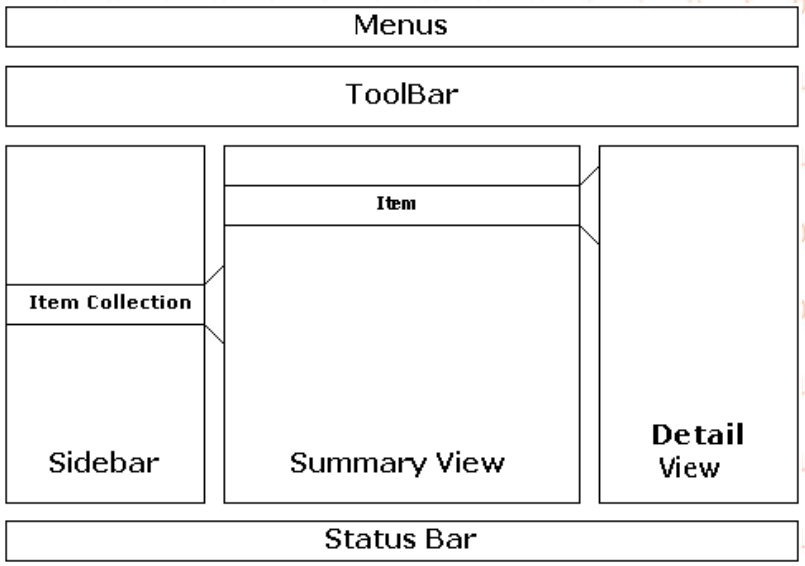
Outline

- Brief Demo
  - A little calendar functionality
- Three sample extensions:
  - ZaoBao, an RSS reader
  - An Amazon wishlist parcel
  - A Flickr parcel

Demo

Chandler Elements

- Model
  - Items (Content Items)
    - calendar events
    - mail, contacts, tasks
  - Item Collections
- View
  - Sidebar
  - Summary View
  - Detail View
  - Menus
  - Toolbar, Status Bar



Parcel Extension Points

- Extend the schema
  - FlickrPhoto
  - Tag
  - PhotoCollection
- UI
  - Menu handler
  - Summary View
  - Detail View
- Create background task
  - Load data into repository

Extending Chandler Schema

- Create a new “Kind”
  - Actually, 2 kinds
  - FlickrPhoto, FlickrPhotoMixin
- Define “Attributes” for this “Kind”
  - owner
  - imageURL
  - tags
  - datePosted

Chandler's built in Photo Kind

```
import osaf.contentmodel.ContentModel as ContentModel

class PhotoMixin(ContentModel.ContentItem):

  schema.kindInfo(displayName="Photo Mixin Kind",
                  displayAttribute="caption")

  caption = schema.One(schema.String, displayName="Caption")
  dateTaken = schema.One(schema.DateTime,
                         displayName="Date Taken")
  data = schema.One(schema.Lob)
  file = schema.One(schema.String)
  exif = schema.Mapping(schema.String, initialValue={})

class Photo(PhotoMixin, Notes.Note):
  schema.kindInfo(displayName = "Photo")
```

Adding a FlickrPhotoMixin Item

```
class FlickrPhotoMixin(Photos.PhotoMixin):

  schema.kindInfo(displayName="Flickr Photo Mixin",
                  displayAttribute="caption")

  flickrID = schema.One(schema.String,
                        displayName="Flickr ID")
  imageURL = schema.One(schema.URL,
                        displayName="imageURL")
  datePosted = schema.One(schema.DateTime,
                          displayName="Upload Date")
  tags = schema.Sequence(displayName="Tag")
  owner = schema.One(schema.String,
                     displayName="Owner")
```

Adding a FlickrPhoto Item

```
class FlickrPhoto(FlickrPhotoMixin, Notes.Note):
  schema.kindInfo(displayName = "Flickr Photo")
```

Adding Tags

```
class Tag(ContentItem):

  itemsWithTag =
    schema.Sequence(FlickrPhoto,
                    inverse=FlickrPhoto.tags,
                    displayName="Tag")
```

Creating a Collection

```
class FlickrPhotoCollection(ContentModel.ContentItem):

  schema.kindInfo(displayName="Collection of Flickr Photos")

  photos = schema.Sequence(FlickrPhotoMixin,
                           displayName="Flickr Photos")

  username = schema.One(
    schema.String, displayName="Username", initialValue=''
  )

  tag = schema.One(
    Tag, otherName="itemsWithTag", displayName="Tag",
    initialValue=None
  )
```

Creating a Menu Item

```
<MenuItem itsName="NewFlickrCollectionByTag">
  <blockName>
    NewFlickrCollectionByTagItem
  </blockName>
  <title>New Flickr Collection by Tag</title>
  <event itemref="doc:NewFlickrCollectionByTagEvent"/>
  <parentBlock itemref="main:NewItemMenu"/>
</MenuItem>
```

Creating an Event

```
<BlockEvent itsName="NewFlickrCollectionByTagEvent">
  <blockName>NewFlickrCollectionByTag</blockName>

  <dispatchEnum>SendToBlockByReference</dispatchEnum>
  <destinationBlockReference
    itemref="doc:FlickrCollectionControllerItem"/>
  <commitAfterDispatch>True</commitAfterDispatch>
</BlockEvent>
```

Creating an Event Handler

```
class FlickrCollectionController(Block):

  def onNewFlickrCollectionByTagEvent(self, event):
    CreateCollectionFromTag(self.itsView,
                            Globals.views[0])
```

Creating an Event Handler

```
from application.dialogs.Util import promptUser

def CreateCollectionFromTag(repView, cpiaView):
  myPhotoCollection =
    FlickrPhotoCollection(view = repView)
  tagstring =
    promptUser(wx.GetApp().mainFrame, "Tag",
               "Enter a Flickr Tag", "")
  myPhotoCollection.tag = Tag.getTag(repView,
                                     tagstring)
  myPhotoCollection.getCollectionFromFlickr(repView)

  # Add the channel to the sidebar
  cpiaView.postEventByName(
    'AddToSidebarWithoutCopying',
    {'items' :
     myPhotoCollection.sidebarCollection]})
```

Creating an Event Handler

```
def getCollectionFromFlickr(self, repView):
  coll = ItemCollection.ItemCollection(view = repView)
  if self.username:
    …
  elif self.tag:
   flickrPhotos =
      flickr.photos_search(tags=self.tag,per_page=10)
   coll.displayName = self.tag.displayName

   self.sidebarCollection = coll

   for i in flickrPhotos:
     photoItem = getPhotoByFlickrID(repView, i.id)
     if photoItem is None:
       photoItem = FlickrPhoto(photo=i, view=repView,
                               parent=coll)
       coll.add(photoItem)
```

Behind the scenes: Item Collections

- Explicit list of items (Photo.tags)
- Query against the repository

Summary View

```
class PhotoMixin(ContentModel.ContentItem):
  …
  about = schema.One(redirectTo = 'caption')
  date = schema.One(redirectTo = 'dateTaken')
  who = schema.One(redirectTo = 'creator')
  displayName = schema.Role(redirectTo="caption")


class FlickrPhotoMixin(Photos.PhotoMixin):
  …
  who = schema.One(redirectTo="owner")
```

Detail View

From Photo's parcel.xml

```
<detail:DetailTrunkSubtree itsName="PhotoSubtree">

  <!-- this DetailTrunkSubtree is for Photos -->
  <key itemref="photos:PhotoMixin"/>

  <!-- define UI Elements -->
  <rootBlocks itemref="photos:PhotoDetailsSpacer"/>
  <rootBlocks itemref="photos:PhotoDetails"/>
</detail:DetailTrunkSubtree>
```

Detail View

From FlickrPhoto's parcel.xml

```
<detail:DetailTrunkSubtree>
  <key itemref="flickr:FlickrPhoto"/>
  <rootBlocks itemref="doc:AuthorArea"/>
</detail:DetailTrunkSubtree>
```

Behind the scenes: CPIA

- CPIA: Chandler Presentation Interaction Architecture
- Blocks
  - Sidebar, DetailView, Summary View
  - Menus, Toolbar, StatusBar
- Model/View
  - Block ==> View (Summary View)
  - ContentItem, ItemCollection ==> Mode
- Blocks are also Items
  - Persistent repository Item
  - wxWidgets peer

Getting Items Periodically

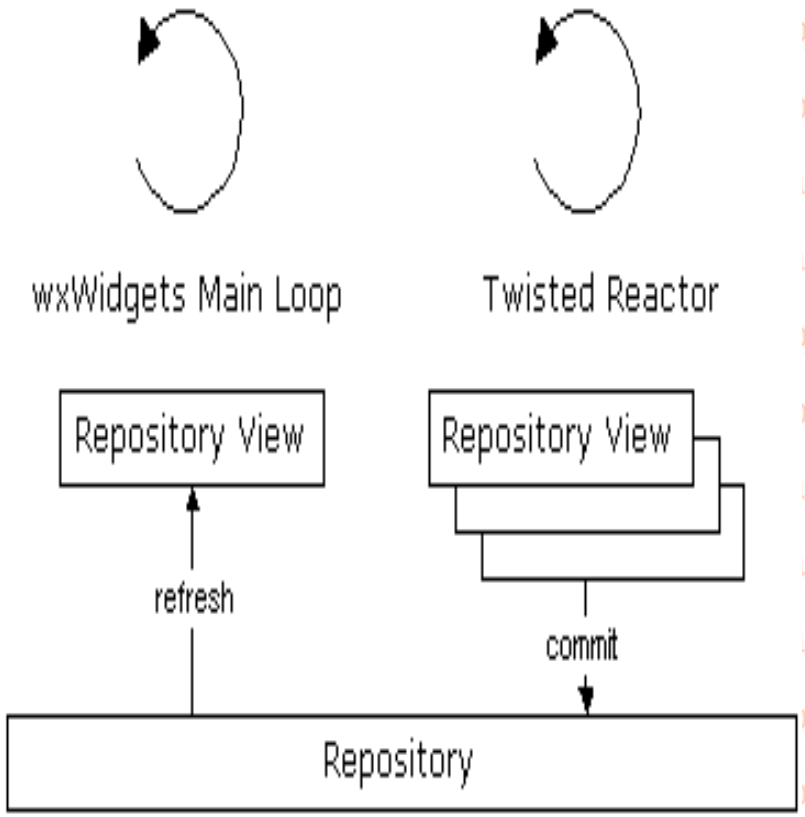
```
<startup:PeriodicTask itsName="FlickrUpdateTask">
  <invoke>osaf.examples.flickr.UpdateTask</invoke>
  <run_at_startup>True</run_at_startup>
  <interval>00:05:00</interval>
</startup:PeriodicTask>
```

Getting new Items

```
class UpdateTask:
  def __init__(self, item):
    self.view = item.itsView

  def run(self):
    # We need the view for most repository operations
    self.view.refresh()

    # We need the Kind object for PhotoCollection
    for myPhotoCollection in
      PhotoCollection.iterItems(self.view):
      myPhotoCollection.update(self.view)

     # commit the changes to the repository
     self.view.commit()
     return True
```

Behind the scenes: Twisted

- wxWidgets thread runs GUI event loop
- Twisted reactor schedules work outside Widgets thread
- PeriodicTasks run via Twisted reactor
- Communication through the repository



O'REILLY

Behind the scenes: Parcels

- parcel.xml
  - xml files that sit alongside python modules
- Items are loaded into repository
  - Discovered by queries
  - Linked to well known items
- Repository is primary data store
  - changes not serialized back to parcel.xml

Summary

- Chandler has a model driven architecture
- Extension points
  - added new data type (Kinds)
  - added menu
  - added view customizations
  - added background task (PeriodicTask)
- Data discovery from repository
  - Queries, references to well known Items

Parcel Futures

- Documentation
  - Tutorial
- Refactoring, modularization
  - Flatten parcel hierarchy
- PythonEggs
- Python for instances

Python Instances

```
controller = FlickrCollectionController.update(parcel,
  'FlickrCollectionControllerItem')

ownerEvent = BlockEvent.update(parcel,
  'NewFlickrCollectionByOwnerEvent',
  blockName = 'NewFlickrCollectionByOwner',
  dispatchEnum = 'SendToBlockByReference',
  destinationBlockReference = controller,
  commitAfterDispatch = True)

tagEvent = BlockEvent.update(parcel,
  'NewFlickrCollectionByTagEvent',
  blockName = 'NewFlickrCollectionByTag',
  dispatchEnum = 'SendToBlockByReference',
  destinationBlockReference = controller,
  commitAfterDispatch = True)
```

Get Involved

- PyCon Sprint
  - 2 days, 2 parcels
    - del.icio.us
    - flickr
- dev@osafoundation.org
- http://wiki.osafoundation.org/Projects/ChandlerHome
- irc://irc.osafoundation.org#chandler


The 7th Annual
O'REILLY®
OPEN
SOURCE
CONVENTION
AUGUST 1-5 2005