Building the Open Library

Aaron Swartz
i build
web applications
1999: get.info
2005: reddit.com
2007: Open Library
what would an online library look like?
a wiki with one page per book
quick facts

- six people (distributed)
- Internet Archive in the Presidio
- not-for-profit
- foundation + government funding
- starting with monographs
1) infrastructure
<table>
<thead>
<tr>
<th>Date</th>
<th>Customer</th>
<th>No._Cases</th>
<th>P_Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Oct-01</td>
<td>Qaffer Inc</td>
<td>1</td>
<td>75</td>
</tr>
<tr>
<td>4-Oct-01</td>
<td>Fiesta Time</td>
<td>19</td>
<td>45</td>
</tr>
<tr>
<td>10-Oct-01</td>
<td>Grogger Ent</td>
<td>2</td>
<td>87</td>
</tr>
<tr>
<td>16-Oct-01</td>
<td>Fine Dining</td>
<td>11</td>
<td>99</td>
</tr>
<tr>
<td>26-Oct-01</td>
<td>Fine Dining</td>
<td>18</td>
<td>87</td>
</tr>
<tr>
<td>31-Oct-01</td>
<td>Qaffer Inc</td>
<td>22</td>
<td>99</td>
</tr>
<tr>
<td>1-Nov-01</td>
<td>Grogger Ent</td>
<td>13</td>
<td>87</td>
</tr>
<tr>
<td>14-Nov-01</td>
<td>Fine Dining</td>
<td>12</td>
<td>99</td>
</tr>
<tr>
<td>29-Nov-01</td>
<td>Fine Dining</td>
<td>15</td>
<td>87</td>
</tr>
<tr>
<td>3-Dec-01</td>
<td>Qaffer Inc</td>
<td>14</td>
<td>45</td>
</tr>
<tr>
<td>10-Dec-01</td>
<td>Fine Dining</td>
<td>11</td>
<td>87</td>
</tr>
<tr>
<td>18-Dec-01</td>
<td>Fiesta Time</td>
<td>13</td>
<td>45</td>
</tr>
<tr>
<td>28-Dec-01</td>
<td>Grogger Ent</td>
<td>2</td>
<td>87</td>
</tr>
<tr>
<td>23-Nov-01</td>
<td>Qaffer Inc</td>
<td>5</td>
<td>45</td>
</tr>
<tr>
<td>10-Dec-01</td>
<td>Fiesta Time</td>
<td>3</td>
<td>87</td>
</tr>
<tr>
<td>11-Dec-01</td>
<td>Grogger Ent</td>
<td>12</td>
<td>75</td>
</tr>
<tr>
<td>12-Dec-01</td>
<td>Qaffer Inc</td>
<td>2</td>
<td>45</td>
</tr>
<tr>
<td>14-Dec-01</td>
<td>Fine Dining</td>
<td>24</td>
<td>87</td>
</tr>
<tr>
<td>Date</td>
<td>Customer</td>
<td>No_Cases</td>
<td>P_Code</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>1-Oct-01</td>
<td>Qaffer Inc</td>
<td>1</td>
<td>75</td>
</tr>
<tr>
<td>4-Oct-01</td>
<td>Fiesta Time</td>
<td>19</td>
<td>45</td>
</tr>
<tr>
<td>10-Oct-01</td>
<td>Grogger Ent</td>
<td>2</td>
<td>87</td>
</tr>
<tr>
<td>16-Oct-01</td>
<td>Fine Dining</td>
<td>11</td>
<td>99</td>
</tr>
<tr>
<td>26-Oct-01</td>
<td>Fine Dining</td>
<td>18</td>
<td>87</td>
</tr>
<tr>
<td>31-Oct-01</td>
<td>Qaffer Inc</td>
<td>22</td>
<td>99</td>
</tr>
<tr>
<td>1-Nov-01</td>
<td>Grogger Ent</td>
<td>13</td>
<td>87</td>
</tr>
<tr>
<td>14-Nov-01</td>
<td>Fine Dining</td>
<td>12</td>
<td>99</td>
</tr>
<tr>
<td>29-Nov-01</td>
<td>Fine Dining</td>
<td>15</td>
<td>87</td>
</tr>
<tr>
<td>3-Dec-01</td>
<td>Qaffer Inc</td>
<td>14</td>
<td>45</td>
</tr>
<tr>
<td>10-Dec-01</td>
<td>Fine Dining</td>
<td>11</td>
<td>87</td>
</tr>
<tr>
<td>18-Dec-01</td>
<td>Fiesta Time</td>
<td>13</td>
<td>45</td>
</tr>
<tr>
<td>28-Dec-01</td>
<td>Grogger Ent</td>
<td>2</td>
<td>87</td>
</tr>
<tr>
<td>23-Nov-01</td>
<td>Qaffer Inc</td>
<td>5</td>
<td>45</td>
</tr>
<tr>
<td>10-Dec-01</td>
<td>Fiesta Time</td>
<td>3</td>
<td>87</td>
</tr>
<tr>
<td>11-Dec-01</td>
<td>Grogger Ent</td>
<td>12</td>
<td>75</td>
</tr>
<tr>
<td>12-Dec-01</td>
<td>Qaffer Inc</td>
<td>2</td>
<td>45</td>
</tr>
<tr>
<td>14-Dec-01</td>
<td>Fine Dining</td>
<td>24</td>
<td>87</td>
</tr>
</tbody>
</table>
SQL

- schema changes are painful
- no revision history
- type unification is tricky
- single-master scaling wall
ThingDB
(infobase?)
ThingDB

b/0L1827T
Book

a/Ben_Fry
Author

s/Fiction
Subject

/dev/docs
Page
<table>
<thead>
<tr>
<th>title</th>
<th>“The Adventures of Tom Sawyer”</th>
</tr>
</thead>
<tbody>
<tr>
<td>author</td>
<td>a/Mark_Twain</td>
</tr>
<tr>
<td>subject</td>
<td>s/Fiction</td>
</tr>
<tr>
<td>subject</td>
<td>s/Americana</td>
</tr>
<tr>
<td>published</td>
<td>1876</td>
</tr>
<tr>
<td>language</td>
<td>“en”</td>
</tr>
<tr>
<td>title</td>
<td>“The Adventures of Tom Sawyer”</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>author</td>
<td>a/Mark_Twain</td>
</tr>
<tr>
<td>subject</td>
<td>s/Fiction</td>
</tr>
<tr>
<td>subject</td>
<td>s/Americana</td>
</tr>
<tr>
<td>published</td>
<td>1876</td>
</tr>
<tr>
<td>language</td>
<td>“en”</td>
</tr>
<tr>
<td>pages</td>
<td>275</td>
</tr>
<tr>
<td>title</td>
<td>“The Adventures of Tom Sawyer”</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>author</td>
<td>a/Mark_Twain</td>
</tr>
<tr>
<td>subject</td>
<td>s/Fiction</td>
</tr>
<tr>
<td>subject</td>
<td>s/Amerciana</td>
</tr>
<tr>
<td>published</td>
<td>1876</td>
</tr>
<tr>
<td>language</td>
<td>“en”</td>
</tr>
<tr>
<td>pages</td>
<td>275</td>
</tr>
<tr>
<td>folk_novel_id</td>
<td>F92982</td>
</tr>
</tbody>
</table>
"The Adventures of Tom Sawyer" a/Mark_Twain s/Fiction s/Americana 1876 "en" 275
• everything’s a thing
• schema updates are trivial
• all changes can be reverted
• time travel
class edit:

    def GET(self, name):
        if name.startswith('_') and not db.theycan('admin', db.cur.site):
            return requirelogin()

        format.correctname(name, '/edit/')

        if name.endswith('_add'):
            if name == '_add': prefix = ''
            else: prefix = web.rstrip(name, '/_add')
            # add?
            if not db.theycan('edit', db.byPage(prefix)): return requirelogin()

            title = ''
            body = ''
            rev = None
        else:
            rev = web.input().get('r')
            page = db.byVersion(name, rev)
            if page: check = page.thing
            else: check = db.cur.site
            if not db.theycan('edit', check): return requirelogin()

            if page is False:
                return web.notfound()
            if page is None or page.deleted():
                rev = False
            if page and hasattr(page, 'body'):
                body = page.body
            else:
                body = ''
            if page and hasattr(page, 'title'):
                title = page.title
            else:
                title = format.pathdecode(name)
            prefix = None

        #blogs = db.Things(kind=db.Blog, parent=db.cur.site).limit(None).list()

        url = ''

        if web.input().get('raw') == 'body':
            print body
        else:
            print render.edit(title, body, prefix, url, rev)

    def POST(self, name):
        if name.startswith('_') and not db.theycan('admin', db.cur.site):
            return requirelogin()

        try: i = web.input('body', 'title')
scripting
scripting

- mismatch with SQL
- specialized query language
- ORM war
- caching is difficult
scripting

• lots of repeated code
• objects
  • map URLs to objects
  • locate the objects
  • display the objects
  • edit the objects
scripting

• lots of repeated code
• accounts
  • create user accounts
  • log in users
• forgot password?
class edit:
    def GET(self, name):
        if name.startswith('_') and not db.theycan('admin', db.cur.site):
            return requirelogin()

        format.correctname(name, '/edit/')

        if name.endswith('_add'):
            if name == '_add': prefix = ''
            else: prefix = web.rstrps(name, '/_add')
            # add?
            if not db.theycan('edit', db.byPage(prefix)): return requirelogin()

            title = ''
            body = ''
            rev = None
        else:
            rev = web.input().get('r')
            page = db.byVersion(name, rev)
            if page: check = page
            else: check = db.cur.site
            if not db.theycan('edit', check): return requirelogin()

            if page is False:
                return web.notfound()
            if page is None or page.deleted():
                rev = False
            if page and hasattr(page, 'body'):
                body = page.body
            else:
                body = ''
            if page and hasattr(page, 'title'):
                title = page.title
            else:
                title = format.pathdecode(name)
            prefix = None

    #blogs = db.Things(kind=db.Blog, parent=db.cur.site).limit(None).list()

    url = ''

    if web.input().get('raw') == 'body':
        print body
    else:
        print render.edit(title, body, prefix, url, rev)

    def POST(self, name):
        if name.startswith('_') and not db.theycan('admin', db.cur.site):
            return requirelogin()
        try: i = web.input('body', 'title')
# open
book = tdb.Thing.get('b/OL1827T')

# view
print '<a href="$book.url">$book.title</a>'

# edit
book.pages = 275
book.save('add pages')

# query
long_books = tdb.Things(
    tdb.k.pages > 200,
    type='book')
caching for free

- store items, queries in RAM
- on save
  - invalidate item cache
  - invalidate matching queries
- broadcast invalidations
scaffolding for free

- **URL mapping**: URL == name
- **view/edit**: types
- **schemas → default templates**
- **custom templates**
- **user templates**
- **history/diff**: free
accounts for free

- infogami handles it all
- your app code exists in plugins
- lots of hooks
HTML templates
HTML templates

- skinning is difficult
- web designers can’t touch SQL
- web designers need SVN
put the templates in the wiki
templates in wiki

- web designers can work away
- skins come for free
- *users* can create skins
- tied to data, not code (time travel!)
2) data
publishers: ONIX

- HaperCollins
- Simon and Schuster
- Random House
- Cambridge University Press
- Penguin
- Wlley
- Thomas Nelson
- O’Reilly
- Elseveier
libraries: MARC

- Library of Congress
- Montana State
- UNC
- Drexel
- University of Toronto
- MBL/WHOI
libraries: MARC

• Talis!
book reviewers

• New York Times
• Harper’s
• New York Review of Books
• Reader’s Catalog
LOVE FOR AN HOUR IS LOVE FOREVER.

CHAPTER I.

“WHO RIDES BY WITH ROYAL AIR?”

“Who rides by with royal air?”

“Though fate may frown, and death may sever,
Love for an hour is love forever.”

BETWEEN the leaves of an old romance I found
one day the shadow of a lily and a song. The lily
grew forty years ago, the song was sung as it was gath-
ered. The flower is nearly dust, the words have nearly
faded away, but the story they keep is unforgotten.
For in becoming “Life” it made itself eternal.

Before the flower bloomed, before the song had found
a voice, Francesca Atherton had dreamed of love, as
saints dream of heaven—wonderful, mystical, far off—an
object both of fervent desire and of wistful fear and
uncertainty. For her young life had been peopled from
noble books, and it was in their pages she had met her
friends and companions—men, romantically honorable
and loyal; women, faithful in love, even unto death—
3) processing
parsing

- lots of Python scripts
- convert to intermediate dictionaries
merging

- MELVYL algorithm
- impressive success
- dupes even in LC
FRBRizing

• coming soon...
4) display
Open Library

Imagine a library that collected all the world's information about all the world's books and made it available for everyone to view and update. We're building that library. (More...)

against love

(Or take a guided tour of the site...)
“Laura Kipnis is a professor of media studies at Northwestern University. She has received fellowships and grants from the Guggenheim Foundation, the Rockefeller Foundation, and the National Endowment for the Arts. She has published numerous essays and articles on sexual politics and contemporary culture both here and abroad. Chapter One

Love’s Labors
~ Options ~

BUY
- Amazon
- Abe Books
- Barnes and Noble
- Alibris
- Powells

BORROW
- Your local library
- BookMooch
- Title Trader

DOWNLOAD
not available for this book
Laura Kipnis

Website:
http://www.communication.northwestern.edu/rtf/faculty/Laura_Kipnis/

Location: Evanston, IL

Alternate Name:

Books By This Author ~

Against Love
Against Love by Laura Kipnis
(Knopf Publishing Group, 20040914)

Against Love
Against Love by Laura Kipnis
(Knopf Publishing Group, 20030826)

The Female Thing
The Female Thing by Laura Kipnis
(Knopf Publishing Group, 20061017)
Browse by Subject - Please select a topic of interest.

- General Works
  - Philosophy, Psychology, and Religion
    - Philosophy (General)
    - Logic
    - Speculative philosophy
    - Psychology
    - Aesthetics
    - Ethics
    - Religions - Mythology - Rationalism
    - Judaism
    - Islam - Bahaism - Theosophy, etc
    - Buddhism
    - Christianity
    - The Bible
    - Doctrinal Theology
    - Practical Theology
    - Christian Denominations
- Auxiliary Sciences of History
- General and Old World History
- History of America
- Music
- Fine Arts
- Language and Literature
- Science
- Medicine
- Agriculture
- Technology
- Military Science
- Naval Science
- Bibliography, Library Science, and General Information Resources
Currently viewing the page:

<table>
<thead>
<tr>
<th>Name ~</th>
<th>Type ~</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>is Primitive</td>
<td></td>
</tr>
<tr>
<td>properties</td>
<td>title of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>subject[] of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>scan_contribution_of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>DOI of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>source_name_of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>GTIN_14 of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>scan_sponsor_of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>edition of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>language_code_of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>source_record_of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>work_of type type/work</td>
</tr>
<tr>
<td>properties</td>
<td>contributions[] of type type/string</td>
</tr>
<tr>
<td>properties</td>
<td>oca_identifier_of type type/string</td>
</tr>
</tbody>
</table>
Currently editing the template for:
Edition View Template

Page type: type/template
Title: Edition View Template

```html
$def with (page)
Svar title: $page.title

$add_stylesheet('/static/css/results.css')

<a name="top"></a>

<div id="book-art">
  <div id="book">
    Sif page.d.get('ISBN_10'):
  </div>
  <br clear="all" />
  <div id="viewthisbook-oln">
    <a href="/addrview?edition=$page.name"><img src="/static/images/reviewbook.png" alt="Review Book" border="0" width="140" /></a>
  </div>
</div>
```
Thank you for your patronage, brewster.

Please review the following information before proceeding:

Scanning-on-demand system for the Open Library is done in partnership with the University of North Carolina School of Information and Library Science. This book is available from the University of Toronto Libraries.

You have offered to sponsor:

The soul of man under socialism, The socialist ideal--art, and The coming solidarity (by Wilde, Oscar.)

The scanning time and cost estimate is:

\[(48\text{pp.} \times 10\text{¢}) + $5 = $9.80\]
HELP WANTED
we want your data
we want your time
we want you