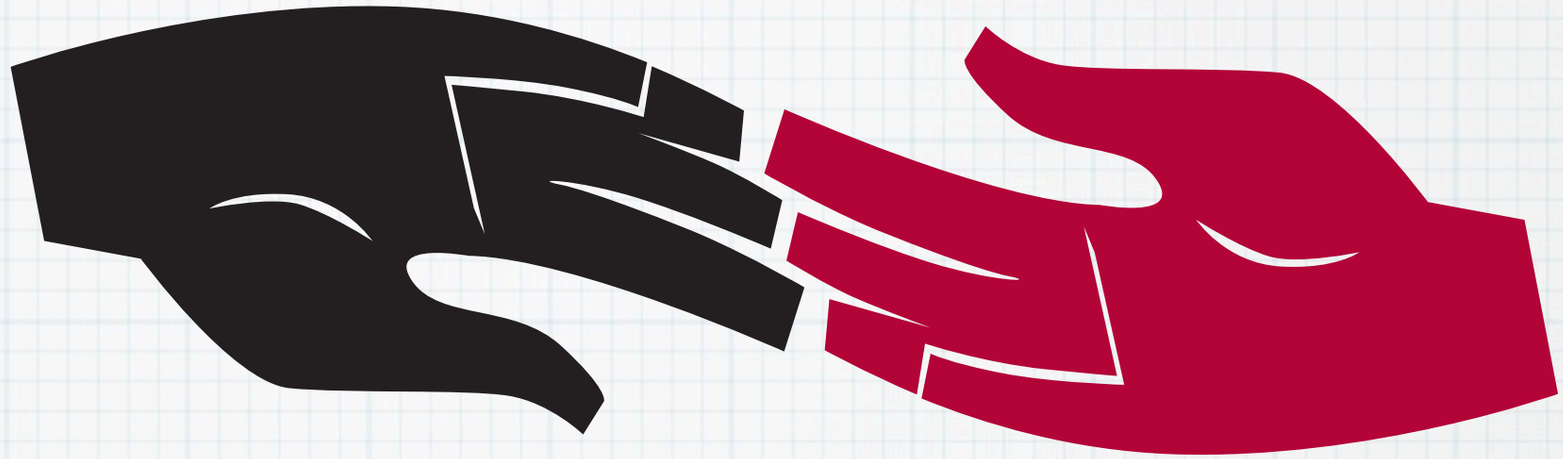


Akara - Spicy Bean Fritters and a Data Services Platform

Uche Ogbuji

26 March 2011, XML Prague

http://x.ogbuji.net/akara-xmlprague

What is Akara?

* Black eyed pea fritter, very popular in West Africa
* Integration platform for data services on the Web
* http://akara.info

Akara recipe

INGREDIENTS

- 2 cups Black eyed peas drained
- 1 Onion finely chopped
- 1/2 teaspoon Salt as required
- 1 chile pepper chopped
- 1/2 teaspoon Ginger roots minced
- “Nigerian pepper”, or Cayenne or Sichan pepper as substitute, to taste
- 1 cup Peanut oil as required

PREPARATION

Clean the black-eyed peas in running water. Soak them in water for at least a few hours or overnight. After soaking them, rub them together between your hands to remove the skins. Rinse to wash away the skins and any other debris. Drain them in a colander.

Crush, grind, or mash the black-eyed peas into a thick paste. Add enough water to form a smooth, thick paste of a batter that will cling to a spoon. Add all other ingredients (except oil). Some people allow the batter to stand for a few hours (overnight in the refrigerator); doing so improves the flavor.

Heat oil in a deep skillet . Beat the batter with a wire whisk or wooden spoon for a few minutes. Make fritters by scooping up a spoon full of batter and using another spoon to quickly push it into the hot oil. Deep fry the fritters until they are golden brown. Turn them frequently while frying. (If the fritters fall apart in the oil, stir in a beaten egg, some cornmeal or crushed breadcrumbs.)

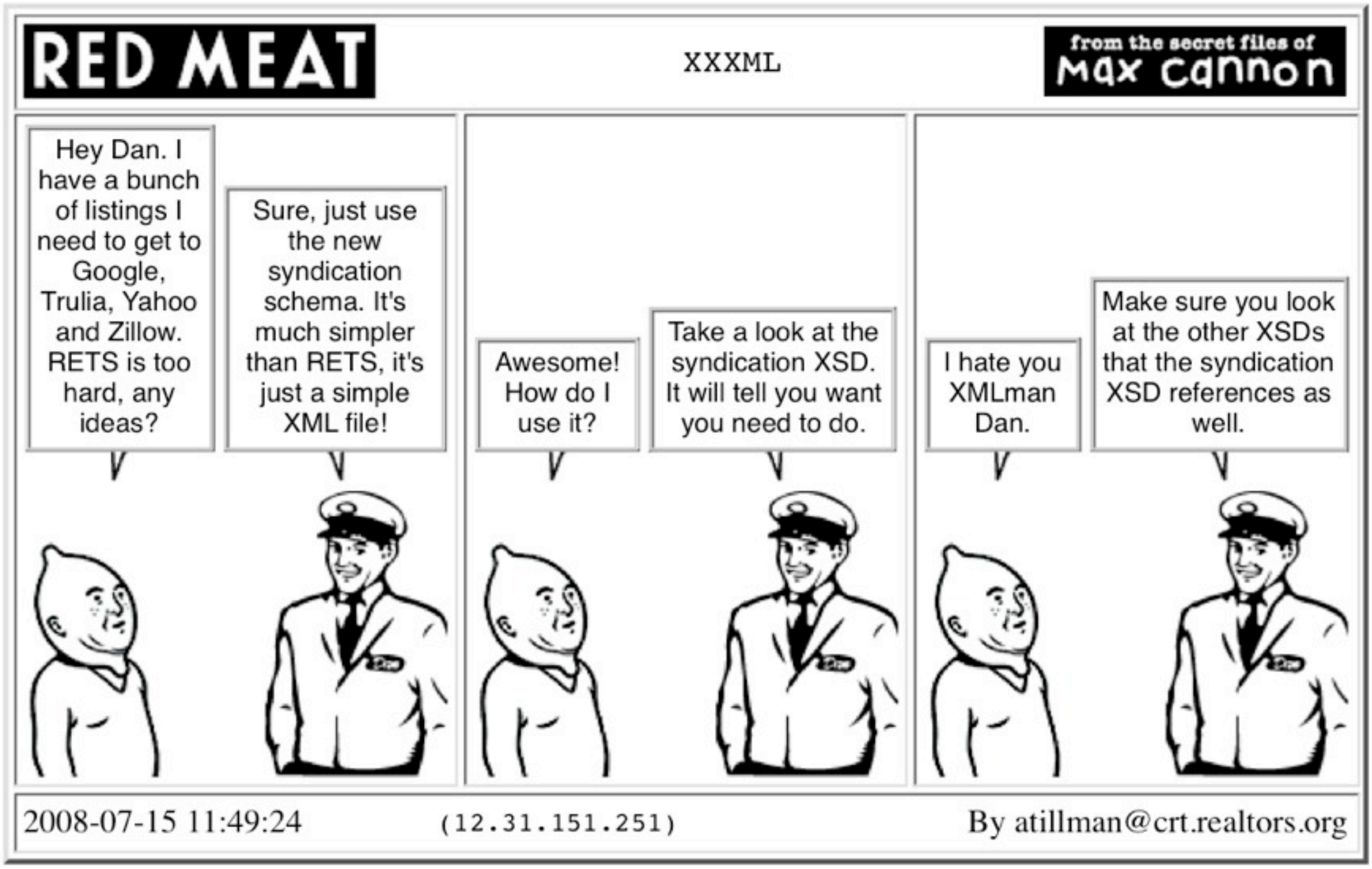
Serve with an African Hot Sauce or salt, as a snack, an appetizer, or a side dish.

Variation: Add a half cup of finely chopped leftover cooked meat to the batter before frying; or add a similar amount dried shrimp or prawns.

Ancestors

* Fourthought: born in 1997
* 4Suite: born in 1999 (from components emerging through '98)
* Amara XML Toolkit: born in 2003
* Akara: born of all the above in 2008

Look, developers hate XML



So why XML Still?

JSON?

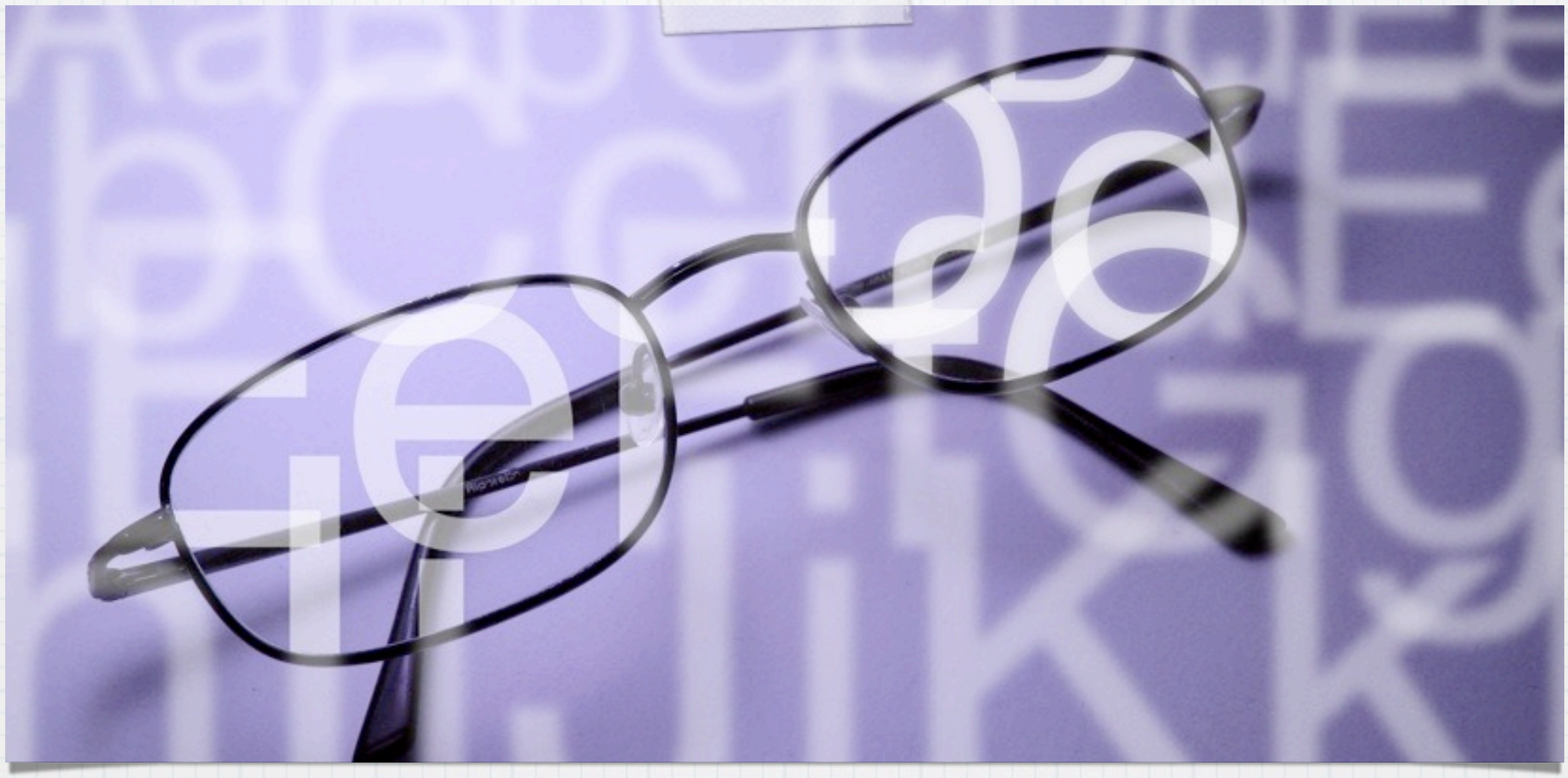
YAML?

Protocol Buffers ?

Coz that's where the data is!

http://www.flickr.com/photos/littlefishyjes/4338571577/

Akara encapsulates 10 years of wisdom in XML and other data transforms, Web services/SOA and Semantic technology



Make the data smart, so the code doesn't have to be

Core concepts

Driving goals

* Combine the practical expressiveness of Python with XML potential for declarative purity
  * Very difficult to balance such divergent moments
* Make it easy to do the right thing in terms of standards and practices
  * Encourage sound design and modeling, using "less code" as an incentive

Amara: Sampler from the core data library

* Expat-based core + hooks for dispatch operations in streaming mode (i.e. SAX without the brainmelt)
* Base tree API (Uses Python conventions, then XPath, then InfoSet, and almost never anything from DOM)
* XPath (& also a dynamic object binding inspired by XPath)
* XSLT (& also a node dispatch mechanism to Python functions)
* Schematron-based assertions, including validation on a manipulated tree (& also a mechanism for expressing model rules in Python)

Akara: RESTful data service framework

* Functions can be written using the core library features, perhaps as unit transforms
* Apply simple wrappers to turn functions into RESTful services
* Akara runs as a repository of services, and allows you to discover these using a simple GET
* Service classes have IDs, independent from locations of individual service end-points

e.g. REST wrappers

* Take any utility function, remote Web service, or data scraper, and add a couple of lines to turn it into a RESTful end-point

* Wikis
* Google search
* Geocoders
* Calais

- RDF
- Feeds (all RSS/Atom flavors)
- Arbitrary Web pages & XML
- SPSS

Web triggers

* AKA "Web hooks"
* Like DBMS triggers: declaration that one event actuates another
* In this case the events are RESTful HTTP messages
* Allows assembly of transform pipelines with minimal effort (i.e. less time and code)

Pipes

* Sorta like Yahoo "Pipes", without the UI
  * Freemix is an example of a GUI front-end, and Akara works purely at the level of data
* Results can be refined using declared patterns
* Very easy to reuse and rearrange pipe components
  * In general you only need to solve a problem once

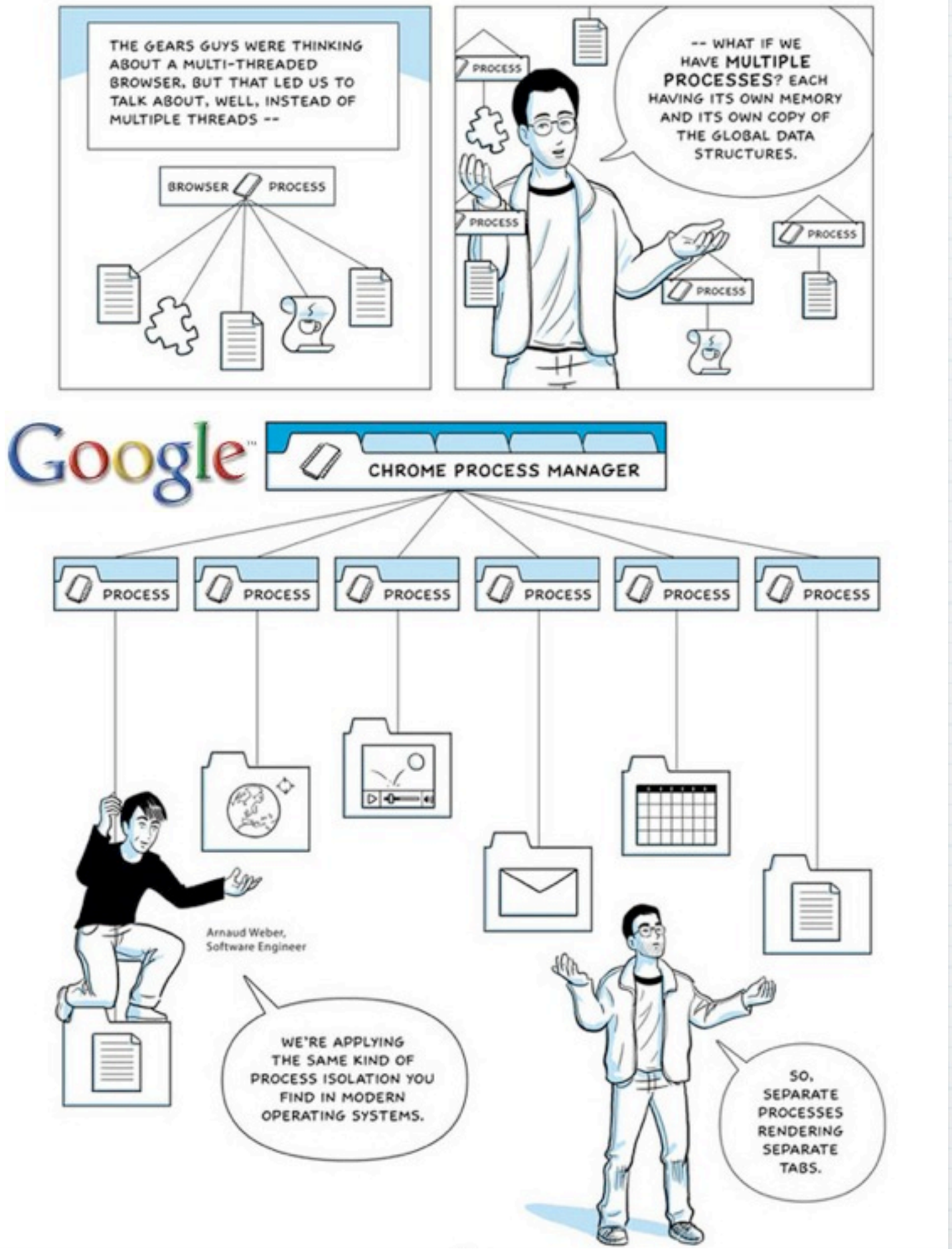
Very modern server

* Server framework largely ported from the very mature/stable 4Suite server core
  * We've been writing network server frameworks for a long time, and we're very good at it
* Designed for sensible cross-platform deployment, stability and scalability (yes multi-core too!)

Threads SUCK!

Akara was born knowing that much (and more)

Transforms run ↗ rather like those tabs



Developer notes

* Implemented in C and Python
* Primary support for transforms in Python
* Supports other RESTful transforms and services (regardless of location, platform, language, etc.)
* Apache-style license


Talk is cheap, show me
the code
LINUS TORVALDS

Core tree API

```
import amara
from amara import tree

MONTY_XML = """<monty>
  <python spam="eggs">What do you mean "bleh"</python>
  <python ministry="abuse">But I was looking for argument</python>
</monty>"""

doc = amara.parse(MONTY_XML)
assert doc.xml_type == tree.entity.xml_type
m = doc.xml_children[0] #xml_children is a sequence of child nodes
assert m.xml_local == u'monty' #local name, i.e. without any prefix
assert m.xml_qname == u'monty' #qualified name, e.g. includes prefix
assert m.xml_prefix == None
assert m.xml_namespace == None
assert m.xml_name == (None, u'monty') #"universal" or "expanded" name
assert m.xml_parent == doc

p1 = m.xml_children[1]
p1.xml_attributes[(None, u'spam')] = u"greeneggs"
p1.xml_children[0].xml_value = u"Close to the edit"
```

"Bindery" API

```
from amara import bindery
MONTY_XML = """<quotes>
  <quote skit="1">This parrot is dead</quote>
  <quote skit="2">What do you mean "bleh"</quote>
  <quote skit="2">I don't like spam</quote>
  <quote skit="3">But I was looking for argument</quote>
</quotes>"""

doc = bindery.parse(MONTY_XML)
q1 = doc.quotes.quote # or doc.quotes.quote[0]
print q1.skit
print q1.xml_attributes[(None, u'skit')] # XPath works too:
q1.xml_select(u'@skit')

for q in doc.quotes.quote: # The loop will pick up both q elements
    print unicode(q) # Just the child char data

from itertools import groupby #Python stdlib
from operator import attrgetter #Python stdlib
skit_key = attrgetter('skit')
for skit, quotegroup in groupby(doc.quotes.quote, skit_key):
    print skit, [ unicode(q) for q in quotegroup ]
```

"Bindery" API 2

```
from amara import bindery

MONTY_XML = """<quotes>
  <quote skit="1">This parrot is dead</quote>
  <quote skit="2">What do you mean "bleh"</quote>
  <quote skit="2">I don't like spam</quote>
  <quote skit="3">But I was looking for argument</quote>
</quotes>"""

from itertools import groupby #Python stdlib
from operator import attrgetter #Python stdlib

skit_key = attrgetter('skit')
for skit, quotegroup in groupby(doc.quotes.quote, skit_key):
    print skit, [ unicode(q) for q in quotegroup ]
```

```
1 [u'This parrot is dead']
2 [u'What do you mean "bleh"', u"I don't like spam"]
3 [u'But I was looking for argument']
```

Incremental parsing

```
from amara.pushtree import pushtree
from amara.lib import U  #U() = "Unicode, dammit!"

def receive_nodes(node):
    print U(node)
    return

XML="""<doc>
  <one>
    <a>0</a> <a>1</a>
  </one>
  <two>
    <a>10</a> <a>11</a>
  </two>
</doc>
"""

pushtree(XML, u'a', receive_nodes)
```

```
0

1

10

11
```

Basic XML modeling

```
from amara import bindery
from amara.bindery.model import *

MONTY_XML = """<monty>
  <python spam="eggs">What do you mean "bleh"</python>
  <python ministry="abuse">But I was looking for argument</python>
</monty>"""

doc = bindery.parse(MONTY_XML)

#Add constraint that `python` elements must have `ministry` attribute
c = constraint(u'@ministry')
try:
    doc.monty.python.xml_model.add_constraint(c, validate=True)
except bindery.BinderyError, e:
    #Exception raised b/c the doc doesn't meet the constraint we
added
    pass #ignore and move on

#Update the doc to meet the desired constraint
doc.monty.python.xml_attributes[None, u'ministry'] = u'argument'
doc.monty.python.xml_model.add_constraint(c, validate=True)
```

Basic XML modeling

```
from amara import bindery
from amara.bindery.model import *

MONTY_XML = """<monty>
  <python spam="eggs">What do you mean "bleh"</python>
  <python ministry="abuse">But I was looking for argument</python>
</monty>"""

doc = bindery.parse(MONTY_XML)

#Add a constraint using a specialized model primitive that supports a
default
c = attribute_constraint(None, u'ministry', u'nonesuch')
doc.monty.python.xml_model.add_constraint(c, validate=True)
```

Modeling by example

```
LABEL_MODEL = '''<?xml version="1.0" encoding="utf-8"?>
<labels>
  <label>
    <name>[Addressee name]</name>
    <address>
      <street>[Address street info]</street>
      <city>[City]</city>
      <state>[State abbreviation]</state>
    </address>
  </label>
</labels>
'''

#Construct a set of constraints and other model info from the example
label_model = examplotron_model(LABEL_MODEL)

#Now use this to validate an instant document VALID_LABEL_XML
doc = bindery.parse(VALID_LABEL_XML, model=label_model)
doc.xml_validate()
```

Model-driven metadata

```
MODEL_A = '''<labels xmlns:eg="http://examplotron.org/0/"
xmlns:ak="http://purl.org/dc/org/xml3k/akara">
  <label id="tse" added="2003-06-10" eg:occurs="*" ak:resource="@id">
    <!-- use ak:resource="" for an anonymous resource -->
    <quote eg:occurs="?">
      <em>Midwinter</em> Spring is its own <strong>season</strong>...
    </quote>
    <name ak:rel="name()">Thomas Eliot</name>
    <address ak:rel="'place'" ak:value="concat(city, ',', province)">
      <street>3 Prufrock Lane</street>
      <city>Stamford</city>
      <province>CT</province>
    </address>
    <opus year="1932" ak:rel="name()" ak:resource="">
      <title ak:rel="name()">The Wasteland</title>
    </opus>
    <tag eg:occurs="*" ak:rel="name()">old possum</tag>
  </label></labels>
'''

labelmodel = examplotron_model(MODEL_A)
```

Model-driven metadata

```
INSTANCE_A_1 = '''<labels>
  <label id="co" added="2004-11-15">
    <name>Christopher Okigbo</name>
    <address>
      <street>7 Heaven's Gate</street>
      <city>Idoto</city>
      <province>Anambra</province>
    </address>
    <opus>
      <title>Heaven's Gate</title>
    </opus>
    <tag>biafra</tag>
    <tag>poet</tag>
  </label>
</labels>
'''

from amara.bindery.model import generate_metadata
doc = bindery.parse(INSTANCE_A_1, model=labelmodel)

for triple in generate_metadata(doc): #Triples but not RDF ;)
    print triple
```

Model-driven metadata

```
INSTANCE_A_1 = '''<labels>
  <label id="co" added="2004-11-15">
    <name>Christopher Okigbo</name>
    <address>
      <street>7 Heaven's Gate</street>
      <city>Idoto</city>
      <province>Anambra</province>
    </address>
    <opus>
      <title>Heaven's Gate</title>
    </opus>
    <tag>biafra</tag>
    <tag>poet</tag>
  </label>
</labels>
'''

from amara.bindery.model import generate_metadata
doc = bindery.parse(INSTANCE_A_1, model=labelmodel)

for triple in generate_metadata(doc):
    print triple
```

```
(u'co', u'name', u'Christopher Okigbo')
(u'co', u'place', u'Idoto,Anambra')
(u'co', u'opus', u'r3e0e1e5')
(u'r3e0e1e5', u'title', u"Heaven's Gate")
(u'co', u'tag', u'biafra')
(u'co', u'tag', u'poet')
```

“Hello Prague”

```
from akara.services import simple_service

HELLO_SERVICE_ID = 'http://example.org/my-services/hello'

@simple_service('GET', HELLO_SERVICE_ID, 'hello')
def helloworld(friend=None):
    return u'Ahoj, ' + friend.decode('utf-8') #Returns Unicode object
```

```
curl http://localhost:8880/hello?friend=Jirka

-> “Ahoj, Jirka”
```

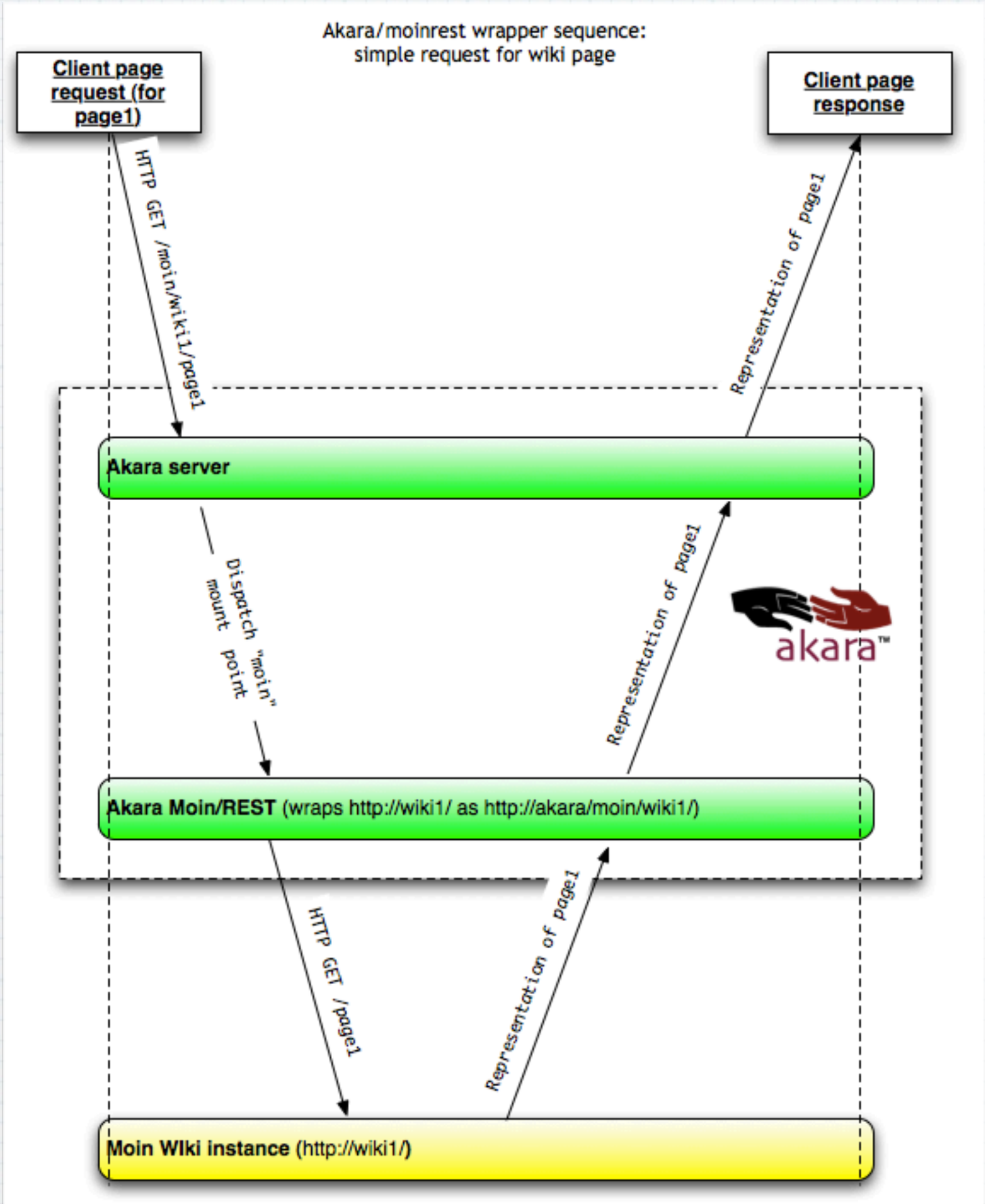
Another Akara module

```
import amara
from akara.services import simple_service, response

ECOUNTER_SERVICE_ID = 'http://purl.org/akara/services/demo/
element_counter'

@simple_service('GET', ECOUNTER_SERVICE_ID, 'ecounter', 'text/plain')
def ecounter(uri):
    doc = amara.parse(uri[0])
    ecount = doc.xml_select(u'count(//*)')
    return str(ecount)
```

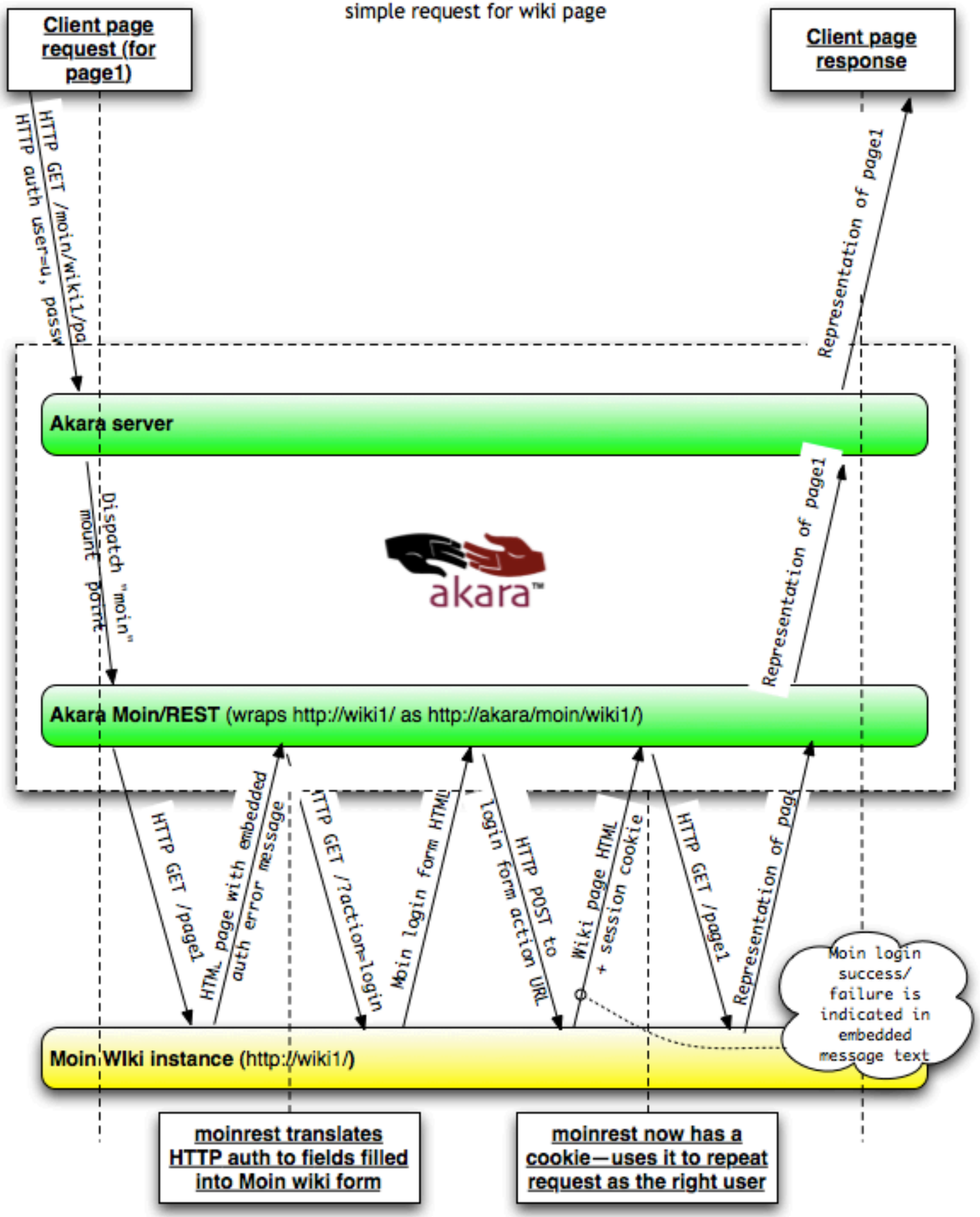
```
curl http://localhost:8880/ecounter?uri=http://example.org/test.xml
```

e.g. Session auth->HTTP auth

Wrapping moin wiki for read/write Web service


Akara/moinrest wrapper sequence:
simple request for wiki page
Client page request (for page1)
Client page response
HTTP GET /moin/wiki1/page1
Representation of page1
Akara server
Dispatch "moin" mount point
Representation of page1
akara™
Akara Moin/REST (wraps http://wiki1/ as http://akara/moin/wiki1/)
HTTP GET /page1
Representation of page1
Moin Wiki instance (http://wiki1/)

Akara/moinrest wrapper sequence:
simple request for wiki page



To do...

* Generic query framework across pipeline results
* Support for policy controls (e.g. API keys/ metering and such)
* Support for local transforms implemented in languages other than Python
* XProc, NVDL, Full Schematron (porting Scimitar)

Questions?

* http://akara.info

* uche@zepheira.com