Web Services Mash-up : Flickr

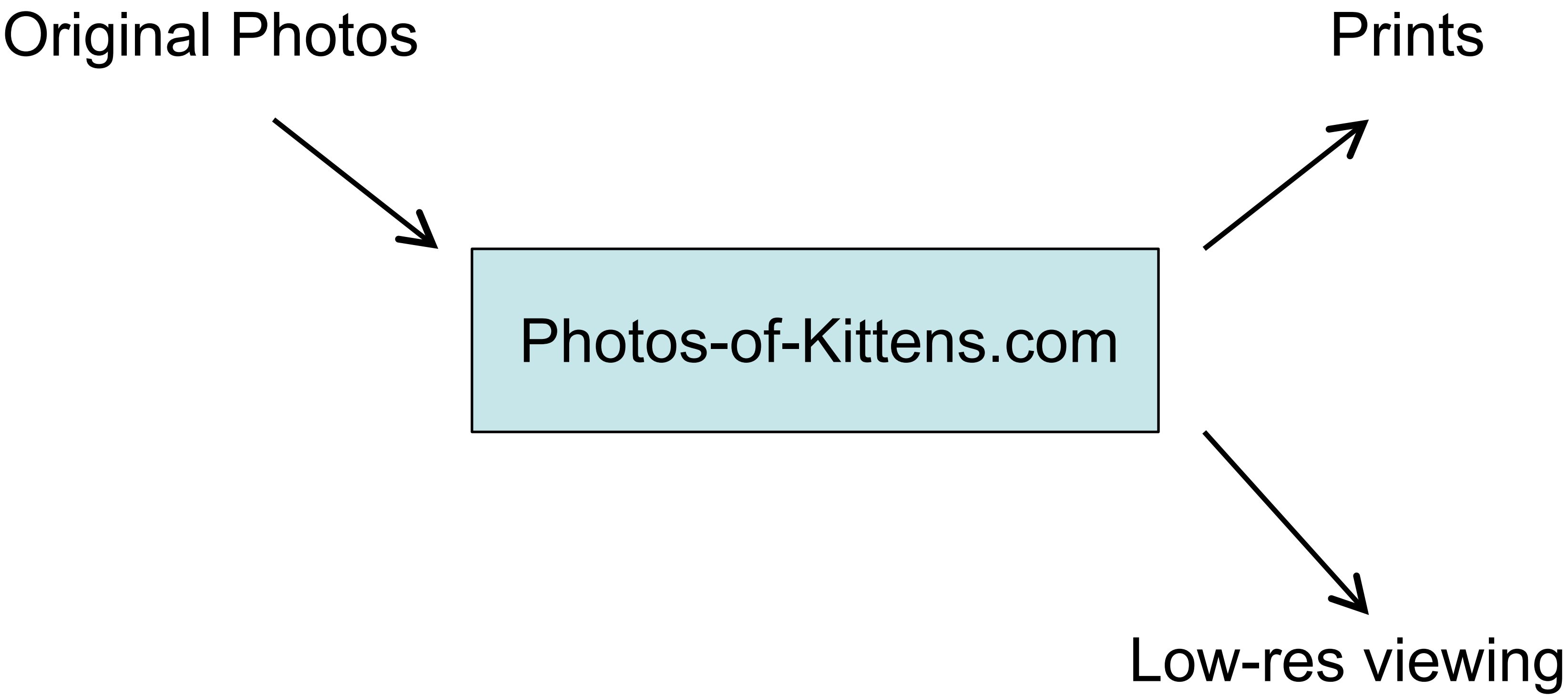
Cal Henderson <cal@flickr.com>

O'Reilly Emerging Technology Conference
March 14-17, 2005

What’s Flickr?

- A website – flickr.com
- A photo-sharing application
- The centre of a big distributed system
- An open set of APIs
  - – flickr.com/services/

‘Traditional’ Photo Sites



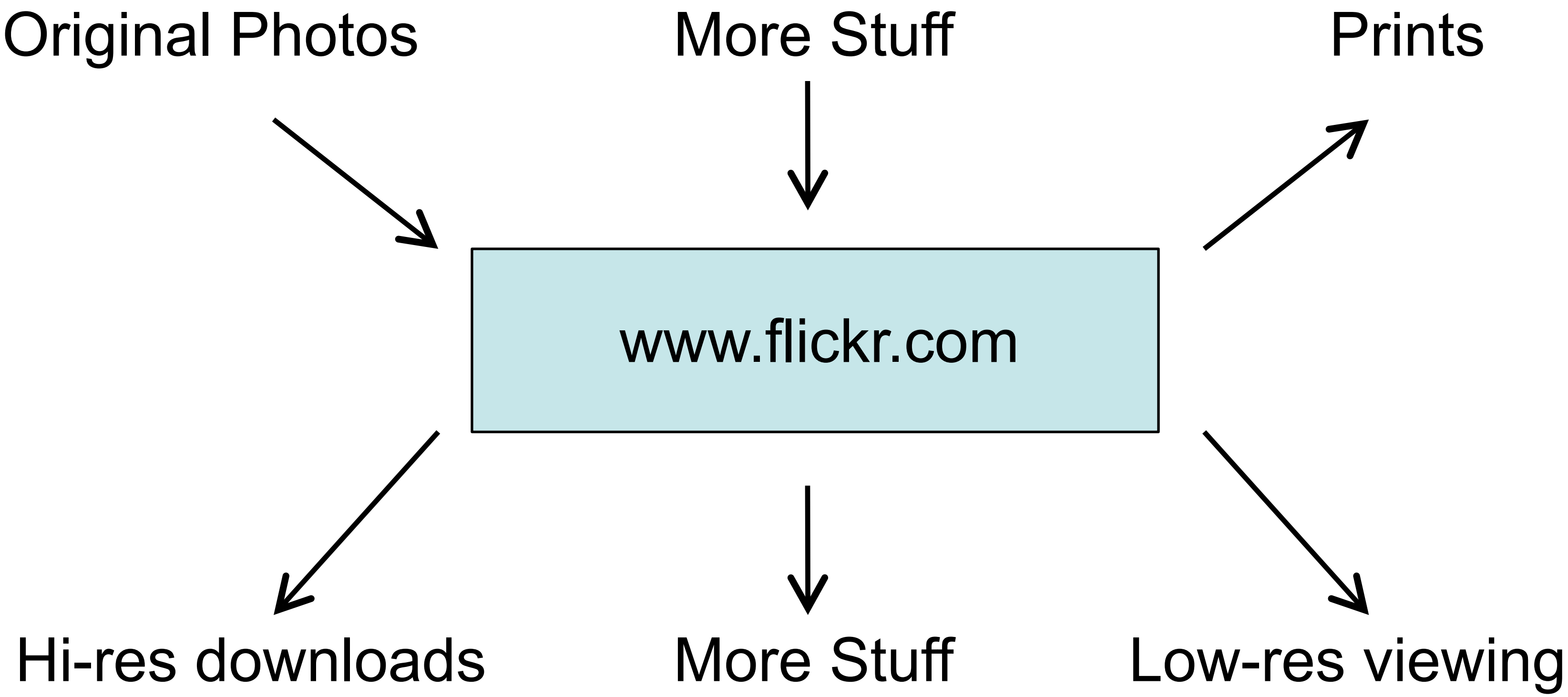
Original Photos

Photos-of-Kittens.com

Prints

Low-res viewing

Flickr



Original Photos

More Stuff

Prints

www.flickr.com

Hi-res downloads

More Stuff

Low-res viewing

More *stuff* ?

- But how do we get more stuff into the system
- And how do we get it out?
- And what ‘stuff’ do we want to be able to get in and out?

Yes, More *Stuff*

- We want everything!
- We don’t yet know what’s great...
  - People can use data in all sorts of cool ways that you never thought of
  - And people can send you data in cool ways too

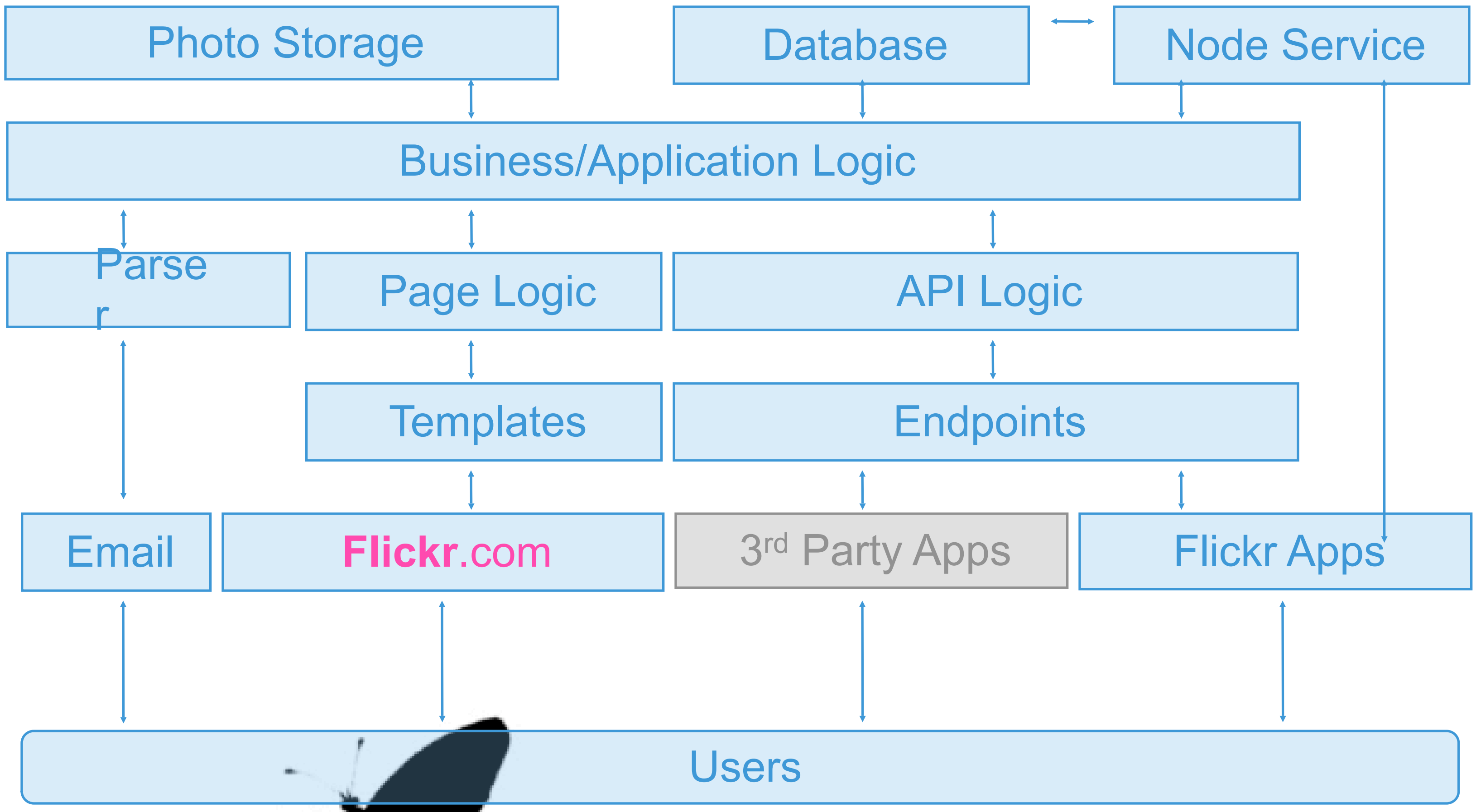
Web services to the rescue

- But what use are web services?
- The future of the Interwebnet!!!1
- A technology which enables geeks to interface with your software

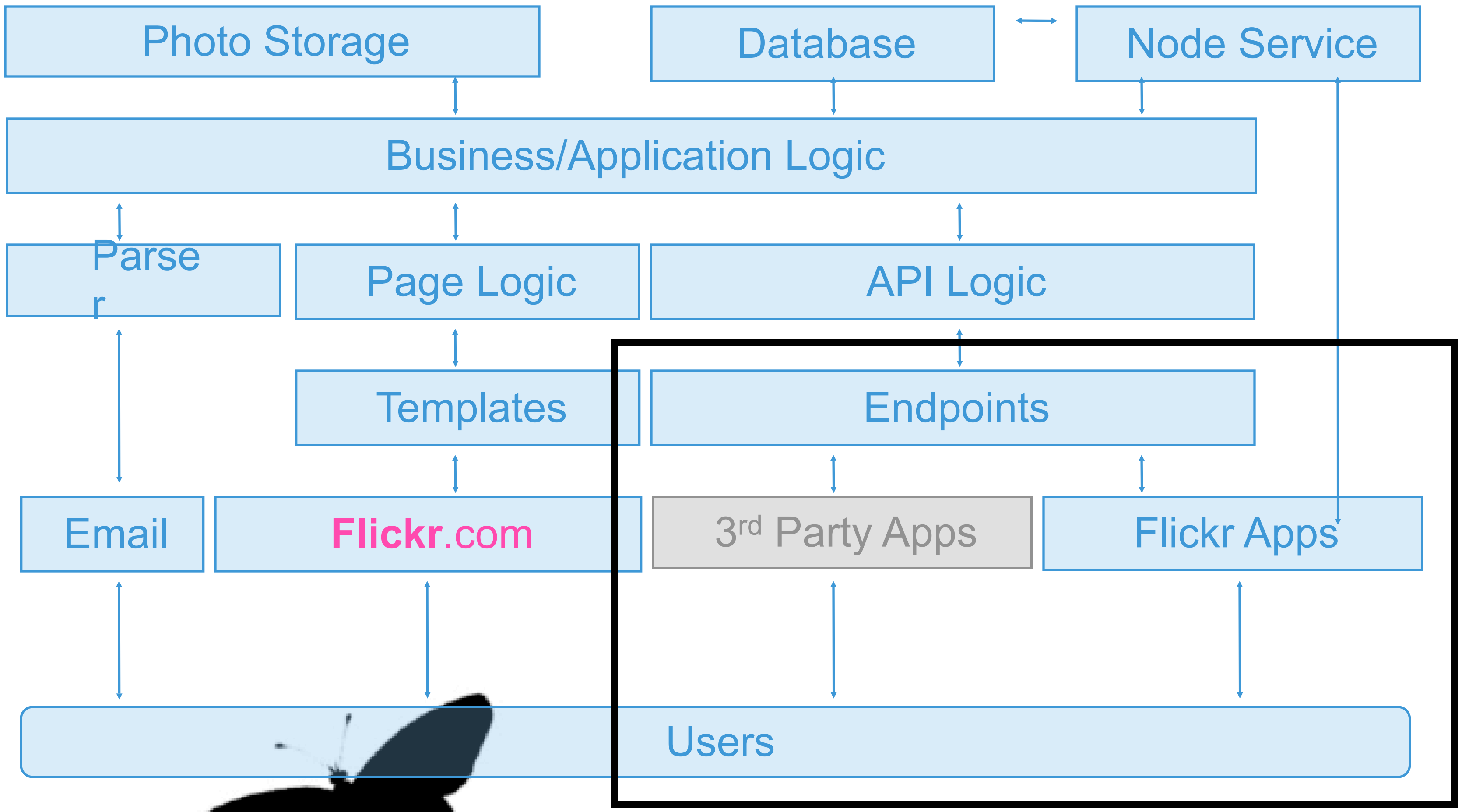
What’s this about a new way?

- Flickr has a bunch of ‘web services’
- RSS/Atom/RDF feeds output data in nice reusable ways
- The Flickr API lets people get data in and out of Flickr however they like

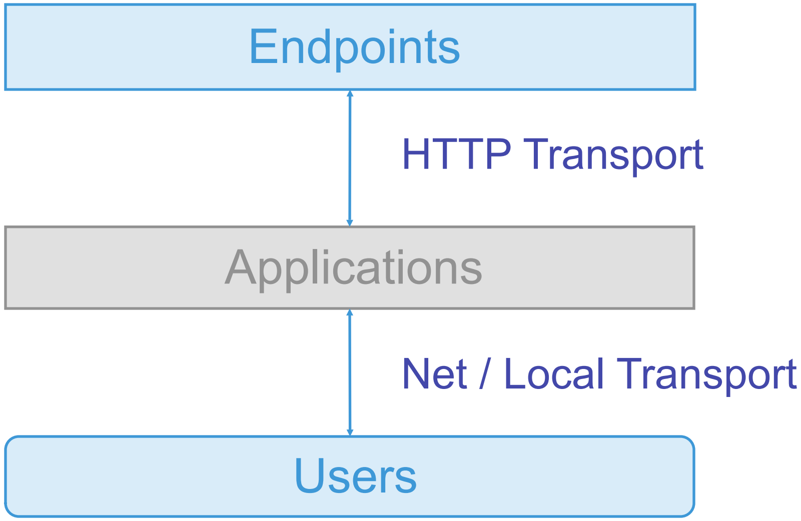
Logical Architecture



Logical Architecture



API Architecture



Protocol Voodoo

- Like any decent Internet ‘standard’, there’s more than one
- A quick guide to the trendy ones flickr supports…

SOAP

- Simple Object Access Protocol
- Now just SOAP
  - (not so simple anymore)
- Google uses it

SOAP Response

```
<s:Envelope
        xmlns:s=“http://www.w3.org/2003/05/soap-envelope”
        xmlns:xsi=“http://www.w3.org/1999/XMLSchema-instance”
        xmlns:xsd=“http://www.w3.org/1999/XMLSchema”
>
        <s:Body>
                <x:FlickrResponse xmlns:x="urn:flickr">
                        [escaped-xml-payload]
                </x:FlickrResponse>
        </s:Body>
</s:Envelope>
```

XML-RPC

- XML Remote Procedure Call
- Used by the blogging APIs
- Created by Dave Winer in 1998
  - Because SOAP was taking a long time

XML-RPC Response

```
<methodResponse>
        <params>
                <param>
                        <value>
                                <string>
                                        [escaped-xml-payload]
                                </string>
                        </value>
                </param>
        </params>
</methodResponse>
```

REST

- Representational State Transfer
  - – Crazy name
    - Thanks Roy Fielding at Apache
- It’s really simple
  - – Just XML over HTTP
  - – (Though purists say it’s *only* HTTP GET)

REST Response

```
<rsp stat="ok">
	[xml-payload]
</rsp>
```

Page Scraping

- Been around for ever
- HTML-over-HTTP
- Volatile interface
- Makes site owners angry
- Other protocols are for sissies (possibly)

Offering Web Services

- Be transport agnostic
  - Some people love SOAP, some love REST
  - Make them all (somewhat) happy

- Beware of ‘shitty coders’

Performance Problems

- People can scrape your site and pull a lot of pages in a short time
- This is bad
- But API abuse (even accidental) can be a lot worse

An example

- Someone writes a trendy screensaver app for Flickr which shows recent photos
- It checks for new photos every 2 seconds
- A bunch of people download it

Danger!

- With 100 users, that’s 50 hits per second (about 4.3 million in a day)
- If it’s making a particularly taxing database call, it’s going to cause problems

Possible solutions

- Incorporate caching into API bindings
- Enforce a policy
  - Through API keys, etc.
- Cache at the host application level
- Monitor things closely

Authentication

- Authentication parameters as request parameters
  - ?username=cal&password=kittens
- HTTP Basic Auth
- HTTPS
- WSSE

What we have learned

- Be open
- Be protocol agnostic
- Be careful of abuse
- Be nice

O'REILLY®

EMERGING TECHNOLOGY

CONFERENCE™

March 14–17, 2005