CM0133 Internet Computing

Syndication and MashUps

Objectives

- Syndication
  - RSS
  - Atom
- Advanced Web Applications
- Mashups
  - How to build a Mashup?
  - Demonstration
  - References and Resources
- Yahoo Pipes

Syndication

For many information sources on the Web, it is useful to have some standardized way of subscribing to information updates. Syndication formats such as **RSS** and **Atom** can be used by these information sources to publish a feed of updated information items. Feeds can be read directly in a browser, but in most cases they are read by specialized software; either a feed reader that allows users to subscribe to more than one feed and manage the information received through all these feeds, or some software module that reads feeds and embeds them for example in a Web page. This latter example is the classical usage of feeds; news feeds published by news agencies, and them embedded as news tickers into Web pages as a constantly updated source of information.

http://dret.net/lectures/web-spring10/syndication

Syndication

- syndicate, verb. to publish simultaneously in a number of periodicals
- A syndicate (noun) is a group of people who create or acquire content to be sold in bulk to magazine and newspapers
- Examples: comic strips [http://www.kingfeatures.com/], editorials

http://dret.net/lectures/web-spring10/syndication

Content Feeds

- Early Web content was static or updated very infrequently
  - there was not yet the requirement to reuse content in different contexts
- Frequently updated Web content quickly became a very common scenario
  - as commercial interests took over the Web, users should have a reason to re-visit a site
  - presenting a steady stream of new content creates the image of a live Web site
- There are two major use cases where HTML is not sufficient

  1. users want an efficient way to get the updated content from a site
  2. sites want to aggregate updated content from other sites and re-publish it

- Syndication Formats are designed to support these two use cases
  - container formats for updated items
  - a small amount of metadata about these items for automated processing

http://dret.net/lectures/web-spring10/syndication

RSS History

- "The Myth of RSS Compatibility" provides a good overview http://diveintomark.org/archives/2004/02/04/incompatible-rss
- RSS is a schoolbook example for "why standards are a good thing"
  - RSS 0.9 was created for the My Netscape portal in March 1999
  - RSS 0.91 (a simplification) was introduced in July 1999 (as an interim solution)
  - the AOL/Netscape merger removed the format from the company's portal
  - RSS was without an owner, and different parties claimed/denied ownership
  - RSS 1.0 was created by an informal developer group
  - RSS 0.92 (and 0.93 and 0.94) were published without acknowledging RSS 1.0
  - finally, RSS 2.0 was released as a follow-up to the RSS 0.9x versions
  - Using RSS has become an exercise in managing a menagerie of versions

RSS

- "Really Simple Syndication"
- Family of web feed formats
- Used for contents that are often updated such as newspapers, online magazines and blogs, .e.g. http://news.bbc.co.uk/1/hi/help/3223484.stm
- RSS Formats are specified using XML
- http://en.wikipedia.org/wiki/RSS
- http://cyber.law.harvard.edu/rss/rss.html

The Case For Content Management

- RSS is very rarely produced by hand
  - by definition, RSS contains redundant information for a specific purpose
- If a Content Management System (CMS) is used, RSS can be generated
  - basic metadata can be generated by the CMS (title, author, date)
  - better tagging of content results in better tagging of feeds
  - well-tagged feeds are better foundations for large-scale reuse of feed items
- Blogging is simply a specialized case of a CMS
  - Web-based interface for controlling everything
  - strictly time-ordered sequenced of published items
  - navigation features primarily based on the time-specific facets of the blog (maybe tags)
- all blogging tools include feed support

http://dret.net/lectures/web-spring10/syndication

Consuming RSS

- RSS feeds often have quality problems
  - surprisingly often feeds do not even deliver well-formed XML
  - the use of embedded markup in RSS is not well-defined
- Writing an RSS reader from scratch is not a good idea
- There are three major tasks which RSS readers must do
  1. accept non-XML RSS feeds and fix them to be XML
  2. look at the feed contents and bring them into a unified form
  3. produce a unified view of feeds regardless of the RSS version

Atom History

- RSS's shortcomings were very apparent and could not be fixed
- In mid-2003, discussions started about an improved format
- It also became apparent that the format should have a protocol
- Atom 0.3 was released in December 2003 but had no formal home
- IETF was chosen as the new home with a working group in June 2004
- RFC 4287 [http://tools.ietf.org/html/rfc4287] was published in December 2005
- AtomPub has been published as RFC 5023 [http://tools.ietf.org/html/rfc5023] in October 2007

Atom vs. RSS

- standardized by the IETF (well-defined process)
- Classification of entries (user-defined categories)
- More XML-like markup design (more nesting)
- Namespaces are used and supported as standard mechanism
- Atom feeds must be well-formed XML (there even is a schema [http://atompub.org/2005/08/17/atom.rnc])
- Interpretation of content is well-defined (various content types)
- Support for xml:lang and xml:base

ATOM Example

```
<feed xmlns="http://www.w3.org/2005/Atom" xml:lang="en-us">
 <title>ongoing</title>
 <id>http://www.tbray.org/ongoing/</id>
 <link rel='self' href="http://www.tbray.org/ongoing/ongoing.atom"/>
 <updated>2007-04-11T12:55:09-07:00</updated>
 <author>
  <name>Tim Bray</name>
 </author>
 <subtitle>ongoing fragmented essay by Tim Bray</subtitle>
 <entry xml:base="When/200x/2007/04/02/">
  <title>Atom Publishing Protocol Interop!</title>
  <id>http://www.tbray.org/ongoing/When/200x/2007/04/02/APP-Interop</id>
  <published>2007-04-02T13:00:00-07:00</published>
  <updated>2007-04-10T14:24:00-07:00</updated>
  <category scheme="http://www.tbray.org/ongoing/What/" term="Technology/Atom"/>
  <category scheme="http://www.tbray.org/ongoing/What/" term="Technology"/>
  <category scheme="http://www.tbray.org/ongoing/What/" term="Atom"/>
  <content type="xhtml">
   <div xmlns="http://www.w3.org/1999/xhtml">
    <p>Mark your calendar: <a href="http://www.intertwingly.net/wiki/pie/April2007Interop">April 16-17 at Google</a>. <em>Everybody</em> is invited, provided they
bring along an APP implementation, client or server. This was just announced a couple of days ago, and as I write this there are already <s>six</s> twelve client and
<s>seven</s> fourteen server implementations signed up to be there and try to <a href="http://www.intertwingly.net/wiki/pie/InteropGrid">fill in the grid</a>. Let’s drop
some names, in alphabetical order: AOL, Flock, Google, IBM, Lotus, Microsoft, Oracle, O’Reilly, Six Apart, Sun, WordPress. Um, have I mentioned that the APP is
going to be huge?</p>
   </div>
  </content>
 </entry>
</feed>
```

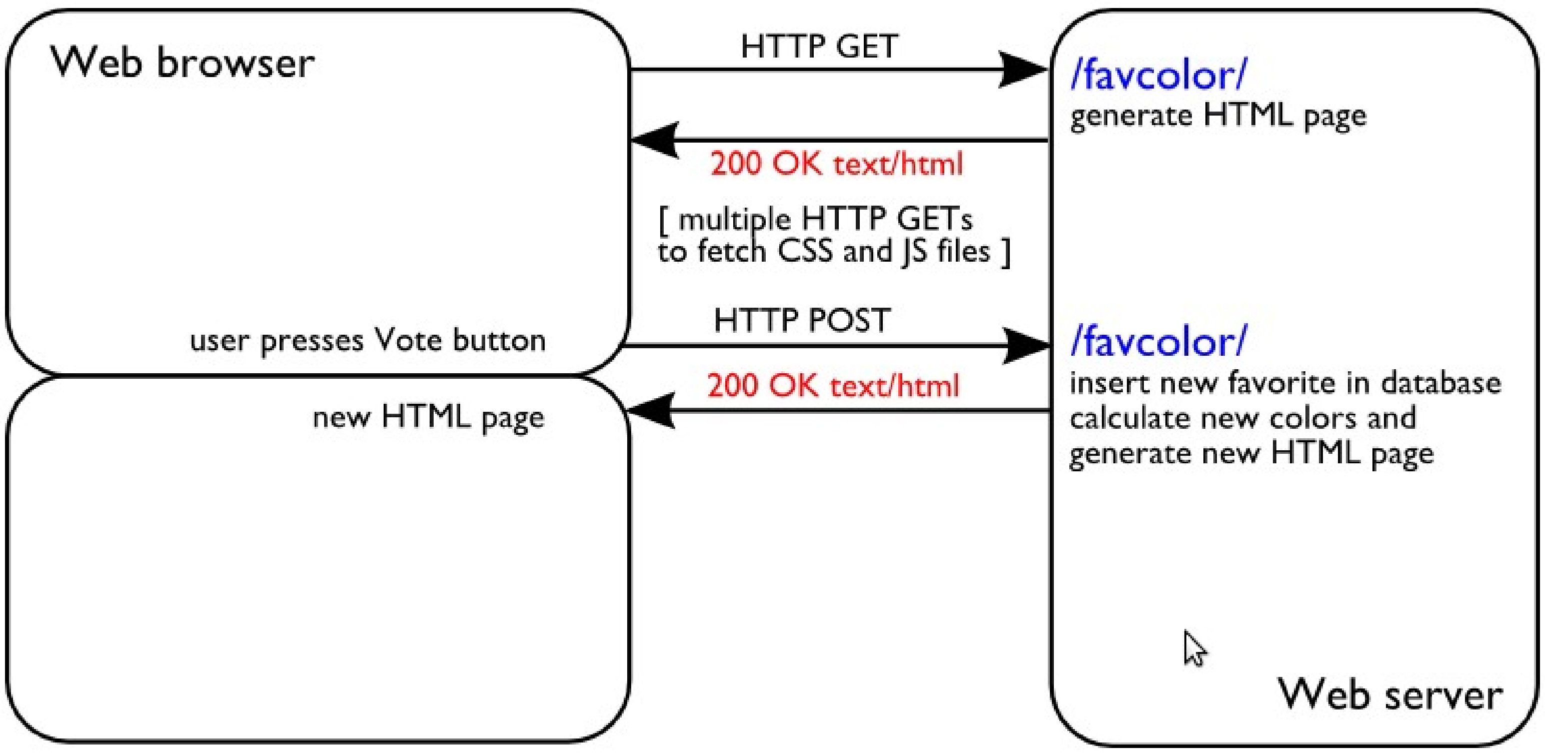
http://dret.net/lectures/web-spring10/syndication

API

The widespread adoption of client-side scripting and AJAX techniques has resulted in web applications becoming easier use but harder to understand. No longer is it the case that HTML is used simply to present a document to be read. Now HTML, Javascript and CSS are used together to build dynamic applications that run in the browser. These applications often depend on APIs, resources intended for use by programs rather than people.

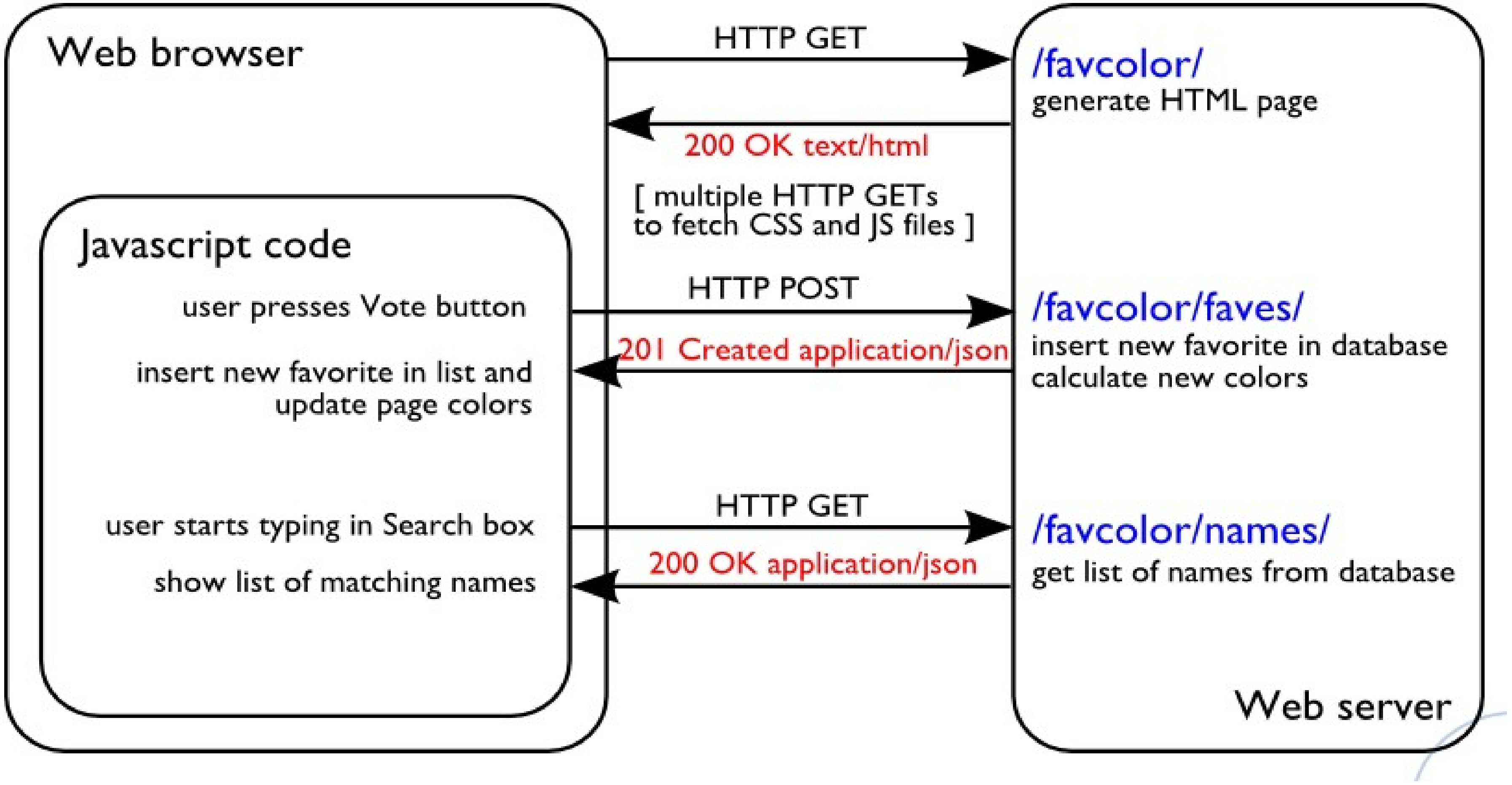
http://dret.net/lectures/web-spring10/webapps-advanced

Favourite Color - Basic



http://waim.aeshin.org/favcolor/
http://dret.net/lectures/web-spring10/webapps-advanced

Favourite Colour - Advanced


Web browser
Javascript code
user presses Vote button
insert new favorite in list and update page colors
user starts typing in Search box
show list of matching names
HTTP GET
200 OK text/html
[multiple HTTP GETs to fetch CSS and JS files]
HTTP POST
201 Created application/json
HTTP GET
200 OK application/json
/favcolor/
generate HTML page
/favcolor/faves/
insert new favorite in database calculate new colors
/favcolor/names/
get list of names from database
Web server

http://dret.net/lectures/web-spring10/webapps-advanced

New Resources

- We've added two new resources, names and faves
  - names only responds to GETs (read-only)
  - faves only responds to POSTs (write-only)
- The representations returned for these resources are JSON (Javascript Object Notation), not HTML
- These resources are intended to be used by programs, not people

Web Service APIs

- These two resources compose a very simple Application Programming Interface [http://en.wikipedia.org/wiki/Application_programming_interface]
- An API is an interface intended for use by programs rather than people
  - people use UIs (User Interfaces)
- There are thousands of Web APIs - http://www.programmableweb.com/apis/directory
- Web APIs are built using URIs and HTTP just like Web UIs are
  - Representations are different: XML or JSON [Javascript Object Notation (JSON) (1)] rather than HTML
- Mashups [http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)] combine data from multiple APIs to create a new application
  - example: Craigslist and Google Maps [http://joesmap.org]

http://dret.net/lectures/web-spring10/webapps-advanced

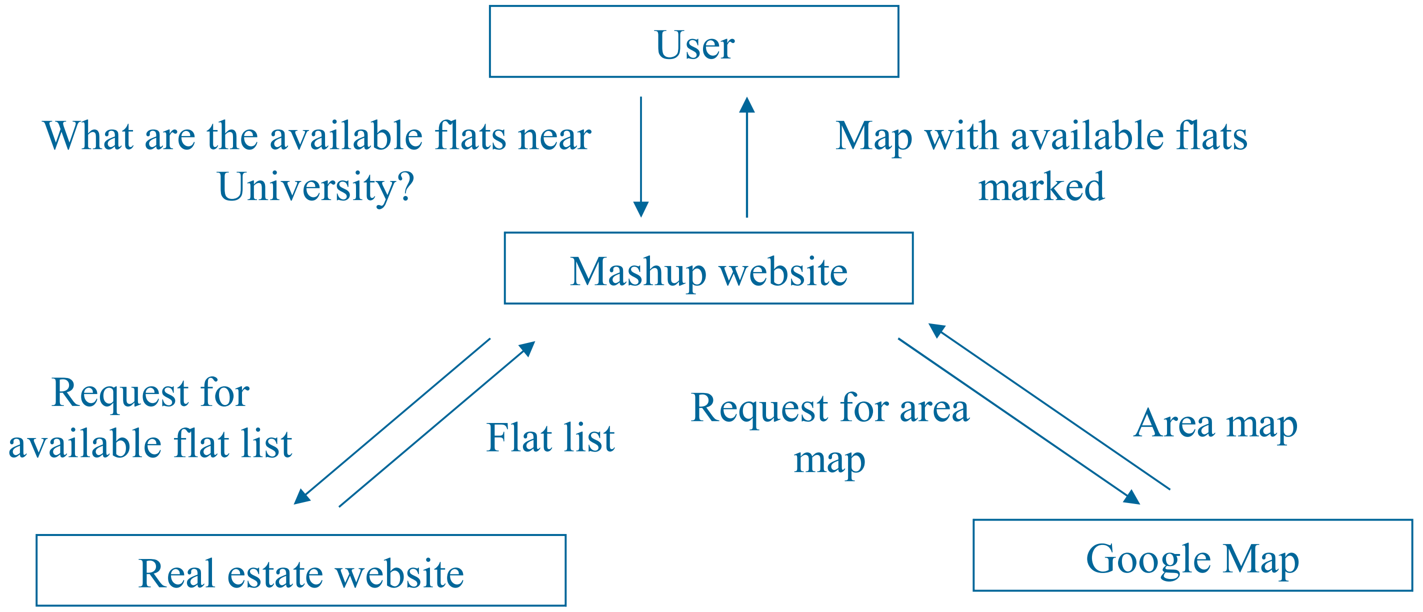
What is a Mashup?

- (Music) A musical genre which combines the layers of music from different songs

- (Computing) A website or web application which combines contents from different websites

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

A Simple Example


User
What are the available flats near University?
Map with available flats marked
Mashup website
Request for available flat list
Flat list
Request for area map
Area map
Real estate website
Google Map

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Pros and Cons

- Pros
  - Information reuse
  - More resources to play with

- Cons
  - Network congestion
  - Speed bottleneck
  - Danger of service failure

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

How to build a Mashup?

- Prerequisites

- Three steps
  - Planning
  - API sign-up
  - Coding

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Prerequisites

- Required
  - Programming Language
  - Web Programming
  - XML Manipulation (XPath, XQuery, etc…)
- Recommended
  - Web authoring tools

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Planning

- Pick a subject
  - A Mashup of What?
    - Map + Real Estate?
    - Bookshop + Library Catalog?
  - More sources of data → More complicated

- Decide your data sources
  - Who is your data provider?
    - Maps: Google maps, Yahoo maps, etc..
    - Online shopping: Amazon, EBay, etc..
  - Usually language agnostic
  - Varying complexity

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Planning

- Other concerns
  - How much time do you have?
  - Do you have a server to run it on?
  - Which programming language?

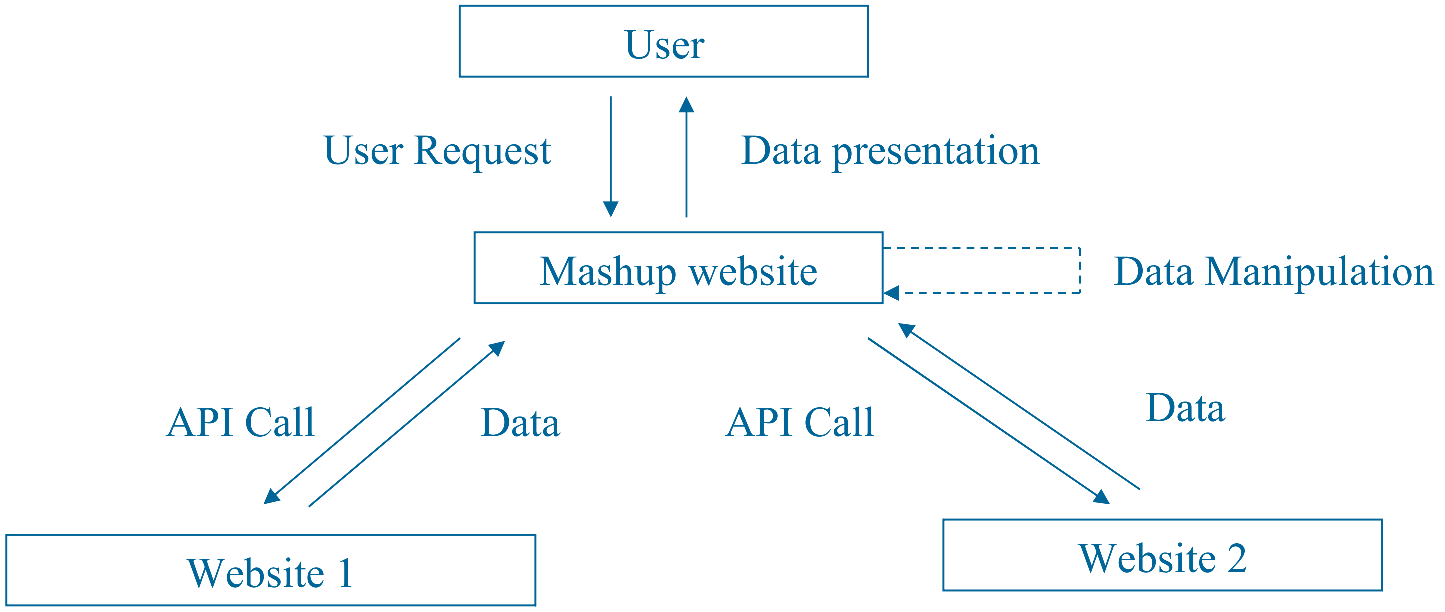
http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

API Sign-up

- Sign-up for the API
  - Visit the homepage of your data source and sign up
  - Examples
    - http://www.google.com/apis/maps/
    - http://www.flickr.com/services/api/
    - http://www.panoramio.com/api/

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Coding – The big picture



http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Coding – The steps

- API Call
- Data Manipulation
- Web programming

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

API Call

- Types of API – How to call?
  - REST
  - XML-RPC
  - SOAP
  - Javascript
- Functions of API – What to call?
  - API specific

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Types of API - REST

- Request in HTTP and Response in XML

- Example: Google Geocoder
  - http://maps.google.com/maps/geo?
  - q – The address that you want to geocode
  - Key – Your API key
  - Output – The output format

- Sample request:
  - http://maps.google.com/maps/geo?q=1600+amphitheare+mtn+view+ca&key=***&output=xml

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Types of API - REST

- Sample response:

```
<xml>
      <Placemark>
           <address>
            1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA
           </address>
           <Point>
                  <coordinates>-122.083739,37.423021,0
                  </coordinates>
           </Point>
      </Placemark>
</xml>
```

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Types of API – XML-RPC

- Request/Response in XML via HTTP

- Example: MSN Blogging API
  - XmlRpcClient in Apache

- Sample Request:
  -
```
<?xml version="1.0"?>
 <methodCall>
    <methodName>getStateName</methodName>
    <params>
       <param><value><i4>4</i4></value></param>
    </params>
 </methodCall>
```

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Types of API – XML-RPC

- Sample Response

```
– <?xml version="1.0"?>
    <methodResponse>
       <params>
          <param>
             <value>
                <string>South Dakota</string>
             </value>
          </param>
       </params>
    </methodResponse>
```

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Types of APIs – SOAP

- Request/Reply in SOAP format via SMTP/HTTP

- Example: MSN Search API
  - HttpURLConnection in Java
  - Post the XML to the target URL

- Sample request:

```
<soap:Envelope xmlns:soap=schemaURL>
<soap:Body>
   <getProductDetails xmlns=targetURL>
<productID>827635</productID>
   </getProductDetails>
      </soap:Body>
</soap:Envelope>
```

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Types of APIs – SOAP

- Sample Response

```
<soap:Envelope xmlns:soap=SchemaURL>
   <soap:Body>
      <getProductDetailsResponse xmlns=targetURL>
         <getProductDetailsResult>
            <productName>Toptimate</productName>
             <productID>827635</productID>
            <price>96.50</price>
         </getProductDetailsResult>
      </getProductDetailsResponse>
   </soap:Body>
</soap:Envelope>
```

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Types of APIs – JavaScript

- Request/Reply embedded in a stub object
- XML not required unless data needed explicitly
- Example: Gogglemap
  - GMAP2 in Googlemap API
  - Var map = new GMAP2(document.getElementById(“map”))
  - map.setCenter(new GLatLng(37.4419,-122.1419),13));

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Functions of API

- API specific and to be Learned as needed, e.g. Panoramio

```
http://www.panoramio.com/map/get_panoramas.php?
order=popularity&
set=public&
from=0&to=20&
minx=-180&miny=90&
maxx=180&maxy=90&
size=medium
```

- for "order" you can use: popularity, upload_date
- for "set" you can use: public (popular photos), full (all photos), user ID number
- for "size" you can use: original, medium (default value), small, thumbnail, square, mini_square
- the minx, miny, maxx, maxy define the area to show photos from (minimum longitude, latitude, maximum longitude and latitude, respectively).

http://www.panoramio.com/api/

Data Manipulation

- Purpose:
  - To generate API requests
  - To process API responses
  - To represent data internally
- Two components
  - Data schema
  - Tools for manipulation

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Data schema (XML)

- Define the schema for your data
  - What data do you want to store?
    - Entities and attributes
  - Good to be generic

- Learn the schema of the APIs

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Tools for manipulation

- Basic level
  - Parser, Modifier, Writer
  - Available online
- Higher level
  - Filter, Converter, Generator
  - Need to write on your own
- Learn to use XPath, XQuery, etc.

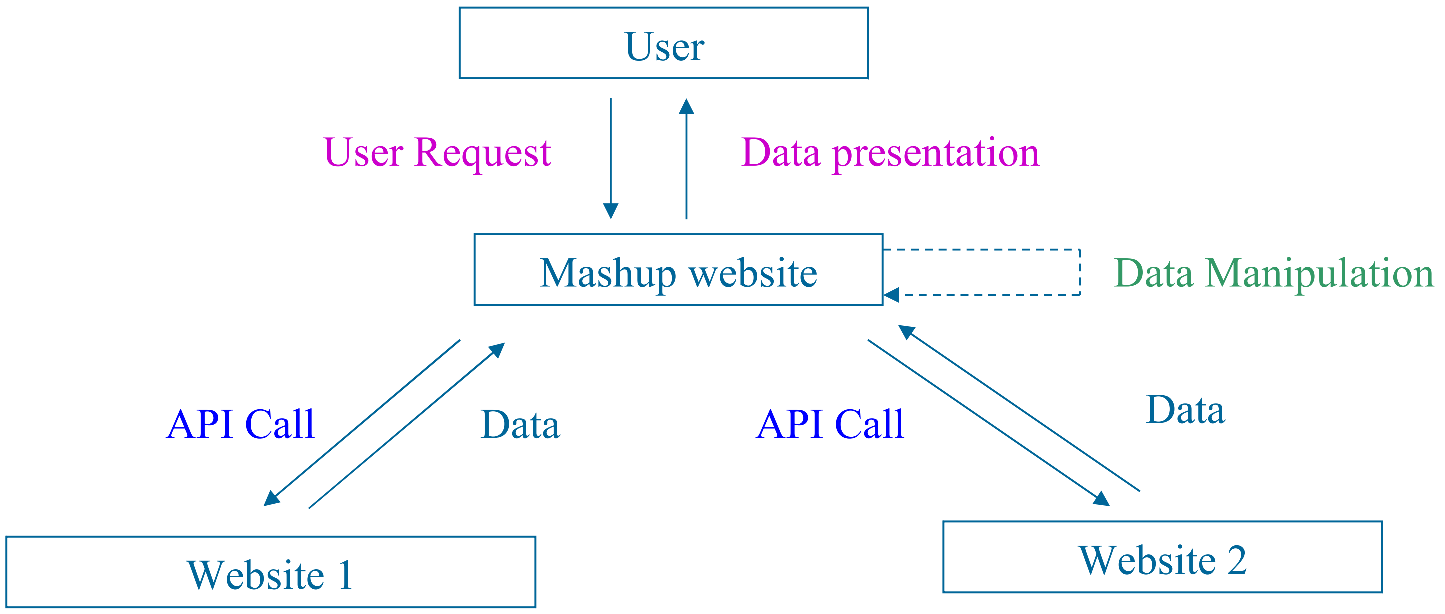
Web Programming

- Design UI for your page

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Putting everything together

- Your Mashup = API calls + Data Manipulation + UI



http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Demonstration

- Googlemap + Craigslist Real Estate
  - http://www.housingmaps.com/
- Amazon + NLB catalog
  - http://www.bookjetty.com/

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

References and Resources

- Reference:
  - Wikipedia: http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)
  - Tutorial from IBM
    http://www-128.ibm.com/developerworks/edu/x-dw-x-ultimashup1.html

- Resources:
  - List of Mashups, APIs:
    http://www.programmableweb.com/
    http://www.webmashup.com/

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

References and Resources

- Resources:
  - APIs and Platforms

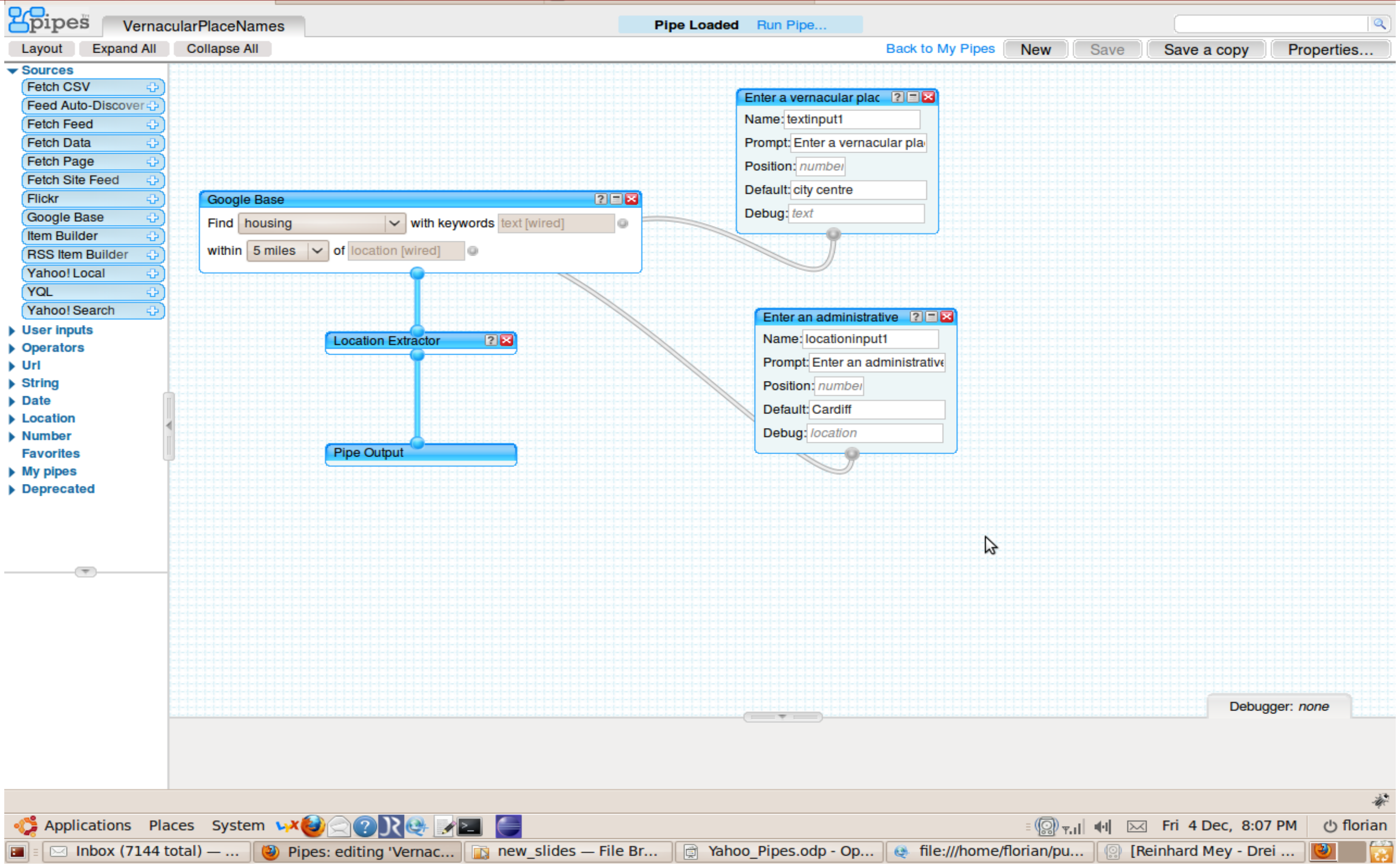
    Amazon:

    http://www.amazon.com/gp/browse.html?node=3435361

    Googlemap: http://www.google.com/apis/maps/
  - XPath, XQuery Tutorial:
    - http://www.w3schools.com/xpath/
    - http://www.w3schools.com/xquery/default.asp

http://aye.comp.nus.edu.sg/meetings/061110/Mashups.htm

Yahoo Pipes



Find: housing with keywords: text [wired]
within: 5 miles of: location [wired]

Name: textinput1
Prompt: Enter a vernacular pla
Position: number
Default: city centre
Debug: text

Name: locationinput1
Prompt: Enter an administrative
Position: number
Default: Cardiff
Debug: location

http://video.yahoo.com/watch/4137374/11144413

http://pipes.yahoo.com/pipes/

Links

- http://www.webreference.com/authoring/languages/xml/rss/custom_feeds/
- http://www.google.com/support/feedburner/bin/answer.py?answer=79408