7 — Web API Design

From Code to Product
gidgreen.com/course
Lecture 7

• Introduction
• REST
• Data formats
• Security
• Maintenance
• Documentation
• Resources
Application Programming Interface

“a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.” — Oxford English Dictionary

“An interface or go-between that enables a software program to interact with other software.” — Investopedia
Types of API

- Programming language libraries, e.g. C
  - `malloc()`, `printf()`, `strcpy()`
- Operating systems, e.g. Android
  - `findViewById(R.id.search).setText(""');`
- Plug-in APIs, e.g. NPAPI for browsers
  - `NPError NP_Initialize(...)`
- Web APIs, e.g. Yahoo! BOSS
Web APIs

• Same infrastructure as websites
  – Request—Response over HTTP
  – Open and exposed to the world

• Textual request/response
  – URLs in, JSON/XML out (generally)

• Many simply wrap web requests...
  – e.g. search APIs, Twitter posting

• ...but many go far beyond
Example: Facebook Graph API
Amazon Product Advertising API

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&Version=2011-08-01
&Operation=ItemSearch
&SearchIndex=Books
&Keywords=harry+potter
&AssociateTag=YourAssociateTagHere
```

```
<TotalResults>2427</TotalResults>
<TotalPages>243</TotalPages>
<Item>
  <ASIN>0545139708</ASIN>
  <DetailPageURL>http://www.amazon.com/Harry-Potter-Deathly-Hallows-Rowling/dp/0545139708%3FSubscriptionId
%3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative
%3D16953%26creativeASIN%3D0545139708</DetailPageURL>
  <ItemLinks>
    <ItemLink>
      <Description>Technical Details</Description>
      <URL>http://www.amazon.com/Harry-Potter-Deathly-Hallows-Rowling/dp/tech-data/0545139708%3FSubscriptionId
%3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative
%3D386001%26creativeASIN%3D0545139708</URL>
    </ItemLink>
    <ItemLink>
      <Description>Add To Baby Registry</Description>
      <URL>http://www.amazon.com/gp/registry/baby/add-item.html%3Fasin.0%3D0545139708%26SubscriptionId</URL>
    </ItemLink>
  </ItemLinks>
</Item>
```
POST  
https://api.twitter.com/1.1/statuses/update.json

POST Data  
status=Maybe%20he%27ll%20finally%20find%20his%20keys.%20%23peterfalk

```
{
  "coordinates": null,
  "favorited": false,
  "created_at": "Wed Sep 05 00:37:15 +0000 2012",
  "truncated": false,
  "id_str": "243145735212777472",
  "text": "Maybe he'll finally find his keys. #peterfalk",
  "contributors": null,
  "retweet_count": 0,
  "id": 243145735212777472,
  "in_reply_to_status_id_str": null,
  "geo": null,
  "retweeted": false,
  "in_reply_to_user_id": null,
  "place": null,
  "source": "<a href=\"http://jason-costa.blogspot.com\" rel=\"nofollow\">My Shiny App</a>",
  "in_reply_to_screen_name": null,
  "in_reply_to_status_id": null
}
```
Growth in Web APIs

ProgrammableWeb API Growth 2005 - 2013
API Billionaires’ Club

- Twitter: 13 billion API calls / day (May 2011)
- Google: 5 billion API calls / day (April 2010)
- Facebook: 5 billion API calls / day (October 2009)
- Netflix: 1.4 billion API calls / day (May 2012)
- AccuWeather: 1.1 billion API calls / day (April 2011)
- Klout: 1 billion API calls / day (May 2012)
- eBay: 1 billion API calls / day (Q1 2012)
- Sabre: 1 billion API calls / day (January 2012)

http://blog.programmableweb.com/2012/05/23/which-apis-are-handling-billions-of-requests-per-day/
Why offer an API?

• Avoid (control) scraping
• Develop partnerships
  – “Business development 2.0”
• Increase revenue (if paid)
• Externalize innovation
  – Copy the best?
• Customer lock-in through integration
Business questions

• What is our goal for the API?
  – How does it contribute to business?
• Free vs paid?
  – Revenue generation vs marketing
• Who will use it?
  – Aim at those developers’ success
• What do they want to do with it?
  – Can our competitors make use of it?
API-focused companies: Stripe

Why you'll love using Stripe

Full-stack payments
You don't need a merchant account or gateway. Stripe handles everything, including storing cards, subscriptions, and direct payouts to your bank account.

Stripe.js lets you build your own payment form while still avoiding PCI requirements.

An API that gets out of your way
It's so easy, we've embedded a bunch of examples right here. Copy some of these requests into your terminal and check out what happens.

With wrappers in Ruby, PHP, Python and more, you can get started in minutes. Learn more ›

$ curl https://api.stripe.com/v1/charges
   -u sk_test_mkGg1qEYW66lnZa4B7HY3V1L
   -d amount=400
   -d currency=usd
   -d "description=Charge for test@example.com"
   -d "card[number]=4242424242424242"
   -d "card[exp_month]=12"
   -d "card[exp_year]=2014"
   -d "card[cvc]=123"

Create a new charge using curl
API-focused companies: Zencoder
API-only companies: SendGrid

SendGrid’s cloud-based email infrastructure relieves businesses of the cost and complexity of maintaining custom email systems. SendGrid provides reliable delivery, scalability and real-time analytics along with flexible APIs that make custom integration a breeze.
API-only companies: Twilio

Your phones just got smarter.

Build intelligent phone systems with **twilio**
API vs licensing code

• Better business model
  – Recurring revenue (by usage)
  – Suits small and large clients
• Easier to maintain
  – No need for “releases”
  – Controlled environment
• Keep control over IP
• But it’s a serious operation
  – Risk of downtime (SLAs?)
REST

• **Representational State Transfer**
  – Most popular design model for Web APIs
• Entities ("resources") = URLs
• Actions = HTTP commands
  – GET, POST, PUT, DELETE
• Resources are self-descriptive
• No hidden server-side state
• (UI Principles applied to developers!)
HTTP request example

PUT /api/dogs/3 HTTP/1.1
Host: dog-db.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

Request data...

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 94

Response data...
REST GET Example 1

GET http://dog-db.com/api/dogs

[
  { id:1, name:"Fido" },,
  { id:2, name:"Rover" },,
  { id:3, name:"Spot" },,
  { id:4, name:"Fluffy" },
]
REST GET Example 2

GET http://dog-db.com/api/dogs/3

```json
{
    id: 3,
    name: "Spot",
    dob: "2009-05-21",
    type: "spaniel",
    photo: "http://dog-db/images/…"
}
```
Expressing relationships

{
  id: 3,
  name: "Spot",
  dob: "2009-05-21",
  owner: {
    id: 16,
    name: "Sam",
    url: "http://dog-db.com/api/owners/16"
  }
...
## REST as CRUD

<table>
<thead>
<tr>
<th>HTTP command</th>
<th>Database operation</th>
<th>/dogs</th>
<th>/dogs/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td><em>Read</em></td>
<td>List all dogs</td>
<td>Get dog details</td>
</tr>
<tr>
<td>POST</td>
<td><em>Create</em></td>
<td>Create new dog</td>
<td>—</td>
</tr>
<tr>
<td>PUT</td>
<td><em>Update</em></td>
<td>—</td>
<td>Update detail/s</td>
</tr>
<tr>
<td>DELETE</td>
<td><em>Delete</em></td>
<td>Delete all dogs</td>
<td>Delete this dog</td>
</tr>
</tbody>
</table>
REST PUT Example

PUT http://dog-db/api/dogs/3
name=Fifi&type=poodle

{
  id:3,
  name: "Fifi",
  dob: "2009-05-21",
  type: "poodle",
}
Rules for REST actions

• GET does not change server state
  – Allows caching, prefetching
  – Like requesting web page

• PUT and DELETE are “idempotent”
  – Repeated calls don’t matter

• POST can change server state each time
  – Classic example: transfer money
  – Like submitting web form
Choosing REST URLs

• Stick to plural forms
  – /dogs → /dogs/3 not /dog/3

• Avoid abstractions
  – /dogs/3 better than /entities/3

• If multiple return types:
  – /dogs/3?type=json
  – /dogs/3.json

• Consistency is king!
More URL best practices

• Pagination of results
  – ?start=20&count=10

• Subset of fields
  – ?fields=id,name,owner,type

• API calls not on resources
  – GET /api/search?q=...
  – GET /api/convert?
    from=km&to=inch&value=0.63
Other protocols

• **Simple Object Access Protocol (SOAP)**
  – XML-based + lots of extra cruft
  – Hard to read and write manually
  – Formalization and discovery via WSDL

• **XML-Remote Procedure Call (XML-RPC)**
  – Simpler precursor to SOAP
  – Based on functions, e.g. `getDogName()`

• Neither uses URLs for entities
Lecture 7

• Introduction
• REST
• Data formats
• Security
• Maintenance
• Documentation
• Resources
Important data types

- String
- Number
- Boolean
- Date/time
- Null/nil
- Binary large objects (BLOBs)
- Array = unlabeled ordered list
- Object = labeled (ordered) list
Extensible Markup Language (XML)

```xml
<dogs>
  <dog id="3">
    <name>Spot</name>
    <age>7</age>
    <type></type>
    <owner id="16">
      <name>Sam</name>
    </owner>
    <collar>true</collar>
  </dog>
  <dog id="4">
    ...  
  </dog>
</dogs>
```

- ✔ User friendly
- ✔ Looks like HTML
- ✗ Wordy
- ✗ Elements vs attributes
- ✗ Implicit typing
- ✗ "123"
- ✗ Array of one
RSS 2.0 (see also: Atom)

<rss version="2.0">
  <channel>
    <title>Dog Tales</title>
    <description>Stories about dogs</description>
    <link>http://dog-tales.com/</link>
    <item>
      <title>Cat chasing</title>
      <description>A dog ran after a cat</description>
      <link>http://dog-tales.com/</link>
      <pubDate>Thu, 09 May 2013 16:45:00 +0000</pubDate>
    </item>
    <item>
      ...
    </item>
  </channel>
</rss>
**Javascript Object Notation (JSON)**

```json
[
    {
        id:3,
        name:"Spot",
        age:7,
        type:null,
        owner:{id:16,name:"Sam"},
        collar:true,
    },
    {
        id:4,
        ...
    }
]
```

- ✅ Compact
- ✅ Explicit types
- ✅ `[]` vs `{}`
- ✅ Javascript-ish
- ✅ JSONP for web access

❌ Feels like programming
Urlencoding

- URL parameters
- Multifield forms (PUT/POST)
BLOBs (rich media)

- Raw delivery
  - Can’t be combined with other data
  - For HTTP use MIME to identify
- Provide URL (string)
  - Separate request to retrieve
- Base64 encoding
  - Inflates size by 33%
  - Standard method for web forms
Error reporting

- Use HTTP response code
  - Allow suppression, e.g. for Flash
- Error in response:

  
  ```
  {
    http-code:401,
    error-code:-329,
    error-message:"Invalid API key",
    error-help:"http://dog-db.com/docs errors/-329.html"
  }
  ```
# HTTP response codes

<table>
<thead>
<tr>
<th>HTTP code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>4xx</td>
<td>Bad request (client’s fault)</td>
</tr>
<tr>
<td>5xx</td>
<td>Failed request (server’s fault)</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized request</td>
</tr>
<tr>
<td>404</td>
<td>Resource not found</td>
</tr>
<tr>
<td>500</td>
<td>Internal error (bug)</td>
</tr>
<tr>
<td>503</td>
<td>Server overloaded</td>
</tr>
</tbody>
</table>
Lecture 7

- Introduction
- REST
- Data formats
- Security
- Maintenance
- Documentation
- Resources
Simple HTTP Authentication

GET /api/dogs/?appID=29838&key=k234nb3bf89
Host: dog-db.com

GET /api/dogs/
Host: dog-db.com
Authorization: Basic QWxhZGRpbjpvcGc2FtZQ==

 ✓ Trivial for developers
 ✗ Visible to intermediaries
 ✓ https can solve this
Signing API calls

• Client and server share secret key
• Signature is hash (one-way function) of:
  – Request
  – Parameters (alphabetical order)
  – Secret key
• Best practice: multiple keys per user
  – Users can disable some applications
• Problem: replay attacks
OAuth 1.0

• Standard for digitally signing API calls
• Permits delegation
  – User grants temporary access to API for them
• Prevents replay attacks
  – Via ‘nonce’ = number used used once
• Popular industry standard
  – Dropbox, Evernote, Flickr, Twitter
• See also: OAuth 2.0
Rate limiting

• Per IP address, but...
  – Proxy networks e.g. Tor
  – Temporary cloud instances
• Per API key, but...
  – Multiple key signups
• Per queried entity
• Based on (API) server load
• Charging solves everything...
Final comments on security

• Do not trust clients
  – All input must be sanitized

• Clients must store key
  – So desktop/mobile apps hackable

• You can’t take back data
  – Limit scope of responses

• Don’t reinvent the wheel
  – Save developers time
Lecture 7

- Introduction
- REST
- Data formats
- Security
- Maintenance
- Documentation
- Resources
Maintenance issues

- Downtime
- Versioning
- Scaling
- Monitoring
- Logging
Downtime

• Developers test then deploy
  – When you go down, they go down
• So avoid at all costs by:
  – Monitoring
  – Versioning
• If unavoidable then:
  – Do it on the weekend
  – Give advanced notice
API status

### Performance and Availability History

<table>
<thead>
<tr>
<th>Service / Website</th>
<th>May 21</th>
<th>May 20</th>
<th>May 19</th>
<th>May 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>/statuses/home_timeline</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>favorites</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>friends/de</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>search</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>stream</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>users/show</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>userstream</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

- ✔️ Service is operating normally
- 🔴 Performance issues
- 🔵 Service is down or have performance issues

---

**Public Website Health Status for Twitter API**

Twitter API Status - Real-time API availability and performance status. This site shows if Twitter APIs are down or have performance issues.

**Follow**

Twitter API @twitterapi

The Real Twitter API. I tweet about API changes, service issues and happily answer questions about Twitter and our API. Don't get an answer? It's on my website.

San Francisco, CA - [http://dev.twitter.com](http://dev.twitter.com)

- 3,423 TWEETS
- 34 FOLLOWING
- 1,658,192 FOLLOWERS

**Tweets**

- **Twitter API @twitterapi**
  - 17 May
  - We'll be performing additional API v1 Blackout Testing on May 22, 2013 at 20:00 UTC. [dev.twitter.com/blog/api-black](http://dev.twitter.com/blog/api-black)
  - View summary

- **Twitter API @twitterapi**
  - 13 May
  - We have deprecated HTTP 1.0 support for the Streaming API: [dev.twitter.com/blog/deprecating-http-1-0-support](http://dev.twitter.com/blog/deprecating-http-1-0-support)
  - View summary
Versioning

GET http://dog-db.com/api/v1/dogs/

• Version at start of URL
• v1 then v2 — no v1.1
  – Makes compatibility clear
• Maintain one version back
• It’s still a failure
  – Add URLs/parameters instead
Scaling

• Usage volumes can surprise you
  – You’re serving software, not people
  – Small number of heavy users
  – Very peaky traffic
• Caching is your friend
• Drop expensive requests under load
• Slow response better than none
• Separate domain: api.dog-db.com
Monitoring

• Volume of API calls
• Popular calls
• Response time
• Error rates
• Active developers
  – Hyperactive developers
• Revenue (+indirect) vs costs
Monitoring made public

Platform Status

Current Status

- Facebook Platform is Healthy
  Last updated: today at 1:02am

- Push: Complete
  As of yesterday at 10:54pm

API Health

Average API Response Time

Error Count
Logging

• Log everything
  – Incoming requests
  – Outgoing response
  – Response time

• To enable...
  – Bug resolution
  – Abuse forensics
  – Deeper (offline) analytics
Lecture 7

• Introduction
• REST
• Data formats
• Security
• Maintenance
• Documentation
• Conclusion
Documentation

• Reference
• Examples
• API explorer
• Language libraries
• Example apps
• Discussion forum
• (and support)
Using OAuth with BOSS API

OAuth allows you, and visitors to your web page, to securely access the Yahoo! Web, Image, and News content. As a publisher, OAuth provides secure access to this content is using your BOSS API application ID and the Yahoo! API key to verify your authorized access privileges and allow for correct billing from Yahoo!

AUTHENTICATION

Authentication for BOSS API queries requires OAuth information in the HTTP header OR through parameters in the GET request. There are six elements that are required for authorization:

1. oauth_version=1.0 – The standard of OAuth supported by BOSS API.
2. oauth_timestamp= – The timestamp is expressed in the number of seconds since January 1, 1970 00:00:00 GMT. The timestamp value MUST be a positive integer and MUST be equal to or greater than the timestamp used in previous requests. The timestamp can be reused for up to 5 minutes. Important: After 5 minutes a fresh timestamp must be supplied.
3. oauth_nonce – is a random string, uniquely generated for all requests for a specific timestamp. This helps verify that a request has never been made before and helps prevent replay attacks when requests are made over a non-secure channel (such as HTTP).
4. oauth_consumer_key= – obtained from YDN during the BOSS API project registration process. This is unique to the developer. Please follow the directions on the displayed key page and copy the entire key from YDN. If you do not copy the entire key, this results in a "Consumer Key rejected" error.
5. oauth_signature_method=HMAC-SHA1 – (specific algorithm used for BOSS API OAuth calls).
6. oauth_signature – can be generated by an OAuth library. A list of supported OAuth libraries is available here: http://oauth.net/code. Over a dozen languages are supported.
Reference: URLs

Syntax for Data Reads (GET)

1. http://yboss.yahooapis.com/ysearch/{service,*}?q={keywords}
2. http://yboss.yahooapis.com/ysearch/{service,*}?service1.q={keywords}&service2.q={keywords}

SERVICE

Below is a list of data sources that can be used with the Query specifications described above:

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>web</td>
<td>Yahoo! web search index results with basic url, title, and abstract data.</td>
</tr>
<tr>
<td>limitedweb</td>
<td>Limited web results. Index refresh rate ~ 3days.</td>
</tr>
<tr>
<td>images</td>
<td>Image search. Image Search includes images from the Yahoo! Image Search index and Flickr.</td>
</tr>
<tr>
<td>news</td>
<td>News search. News Search includes late breaking news articles from the past 30 days.</td>
</tr>
<tr>
<td>blogs</td>
<td>Blogs search [Beta]</td>
</tr>
<tr>
<td>ads</td>
<td>Advertising. If publisher has qualified for Yahoo! Search Advertising.</td>
</tr>
</tbody>
</table>
# Universal Yahoo! BOSS API Arguments

The following arguments apply to these BOSS API services: Web, Limitedweb, Image, and News.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>Web; Limitedweb; Images; News; Blogs services</td>
</tr>
<tr>
<td>start</td>
<td>Ordinal position of first result. First position is 0. Default sets start to 0.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Start parameter cannot be greater than 1000. All services return a maximum of 1000 results only.</td>
</tr>
<tr>
<td>count</td>
<td>Total number of results to return. Maximum default value is 50.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>• The maximum default value for images is 35.</td>
</tr>
<tr>
<td></td>
<td>• Blogs maximum default value is 20.</td>
</tr>
<tr>
<td>format</td>
<td>The data format of the response. Value can be set to either &quot;xml&quot; or &quot;json&quot;. Default sets format to &quot;json&quot;.</td>
</tr>
</tbody>
</table>
For each input parameter

- Name of parameter
- Explanation/meaning
- Possible values/range
- Example values
- Optional or required?
  - Default value if optional
Response Fields

This section describes the response fields returned by web search.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract</td>
<td>Abstract with keywords highlighted with HTML tags. <code>abstract=long</code> will retrieve and display an abstract of a web document up to 300 characters. This expanded abstract provides the requestor with a larger piece of information to work from in a web search query.</td>
</tr>
<tr>
<td>title</td>
<td>Returns the result’s title, with keywords highlighted with HTML <code>&lt;b&gt;</code> and <code>&lt;/b&gt;</code> tags.</td>
</tr>
<tr>
<td>url</td>
<td>Returns the result’s URL.</td>
</tr>
<tr>
<td>clickurl</td>
<td>Returns a navigation URL that leads to the target URL for each result. A clickurl might lead through a redirect server, which provides Yahoo! with important usage data from search result sets. See coding requirement (URL vs clickurl) in overview.</td>
</tr>
<tr>
<td>dispurl</td>
<td>Returns the URLs of documents matching the query result. Use this field only for display purposes on result pages. To direct search users to the target document, use the clickurl value.</td>
</tr>
<tr>
<td>date</td>
<td>Returns the date the URL was last crawled in <code>YYYY/MM/DD</code> format.</td>
</tr>
<tr>
<td>language</td>
<td>Displays the language of the search results document. See <code>view=language</code> for more information. Other language results may appear in results.</td>
</tr>
</tbody>
</table>
## Reference: response codes

<table>
<thead>
<tr>
<th>HTTP status codes</th>
<th>Problem Description &amp; Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 bad request</td>
<td>Un-escaped reserved characters. Something else wrong with request (API). Check your BOSS API parameters.</td>
</tr>
<tr>
<td>401</td>
<td>Invalid consumer key in a request</td>
</tr>
<tr>
<td>403 invalid AppID error, access denied, Access is forbidden</td>
<td>Possible production code errors and not client-side errors. Please retry your API query.</td>
</tr>
<tr>
<td>406 not acceptable</td>
<td>Accept-encoding setting must be set to 'gzip'. Keep-alive must be turned off (cannot reuse port connections).</td>
</tr>
<tr>
<td>407</td>
<td>Proxy authentication required. Yahoo! BOSS API does not use proxy authentication. Please check your systems.</td>
</tr>
<tr>
<td>408</td>
<td>Your request has timed out. Please try your request again. If this problem persists, please send a message to the <a href="http://www.bossyahoo.com/groups">BOSS Yahoo! Groups forum</a></td>
</tr>
<tr>
<td>414</td>
<td>Request-URL Too Large. URL too long for the system to process.</td>
</tr>
<tr>
<td>500 Internal server error</td>
<td>Received when server problems occur. Please retry your query. Single letter query may also cause this error to be displayed.</td>
</tr>
<tr>
<td>503</td>
<td>• Service Unavailable. Key has exceeded its configured rate limit. • If it’s a new key, there may be a delay between approval and turn-on. Wait and retry the key again. • If it’s an existing key that worked earlier, please wait and try again. • If the above does not help you resolve your issue and it repeats for long time or happens often, report the issue on the <a href="http://www.bossyahoo.com/groups">BOSS Yahoo! Groups forum</a></td>
</tr>
<tr>
<td>504</td>
<td>Caused by server problems in Yahoo! services. Please retry your query.</td>
</tr>
</tbody>
</table>
Examples

**SIMPLE QUERY EXAMPLE**


**SIMPLE XML RESPONSE**

1. <bossresponse responsecode="200">
2.   <web start="0" count="50" totalresults="32"/>
3.     <results>
4.       <result>
5.         <date>2011/02/01</date>
6.         <clickurl>http://www.barackobama.com/</clickurl>
7.         <url>http://www.barackobama.com/</url>
8.         <dispurl type="default">www.barackobama.com</dispurl>
9.         <title type="default">Organizing for America | BarackObama.com</title>
10.        <abstract type="default">Organizing for America is the grassroots</abstract>

These should match!
API explorer

Graph API Explorer

Access Token: CAACEe0e0cBACcIuO1eHw4Jte0qql3sSCeBc0nDDHMYrIpc6ik5EdxXcKT261Cr5nRchATUzA4ajG/o8gPBlu77pVY3t

Graph API

FQL Query

POST

https://graph.facebook.com/298000137?fields=id,name

name: Naedla Naasneaga

Learn more about the Graph API syntax.

{ "error": { "message": "(#10) Application does not have permission for this action", "type": "OAuthException", "code": 10 } }
Language libraries

```
def search(command,bucket="web",count=10,start=0,more={}):
    params = {
        'oauth_version': "1.0",
        'oauth_nonce': oauth.generate_nonce(),
        'oauth_timestamp': int(time.time()),
        'q': quote_plus(command),
        'count': count,
        'start': start,
        'format': 'json',
        'ads.recentSource': SOURCE_TAG
    }
    params.update(more)
    url = SEARCH_API_URL_V2 + bucket
    consumer = oauth.Consumer(CC_KEY,CC_SECRET)
    req = oauth.Request(method="GET", url=url, parameters=params)
    req.signature_method = oauth.SignatureMethod_HMAC_SHA1
    req.sign_request(req.signature_method, consumer, None)
    return rest.load_json(req.to_url())
```
Example apps

BOSS Mashup Framework

The BOSS Mashup Framework is an experimental Python library that provides developers with SQL-like constructs for mashing up the BOSS API with third-party data sources.

bossmashup /

<table>
<thead>
<tr>
<th>File</th>
<th>Author</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templates</td>
<td>jcleblanc</td>
<td>2 years ago</td>
<td>boss mashup files - initialization [jcleblanc]</td>
</tr>
<tr>
<td>util</td>
<td>jcleblanc</td>
<td>2 years ago</td>
<td>boss mashup files - initialization [jcleblanc]</td>
</tr>
<tr>
<td>yos</td>
<td>jcleblanc</td>
<td>2 years ago</td>
<td>boss mashup files - initialization [jcleblanc]</td>
</tr>
<tr>
<td><strong>init</strong>.py</td>
<td>jcleblanc</td>
<td>2 years ago</td>
<td>boss mashup files - initialization [jcleblanc]</td>
</tr>
<tr>
<td>config.json</td>
<td>jcleblanc</td>
<td>2 years ago</td>
<td>boss mashup files - initialization [jcleblanc]</td>
</tr>
<tr>
<td>setup.py</td>
<td>jcleblanc</td>
<td>2 years ago</td>
<td>boss mashup files - initialization [jcleblanc]</td>
</tr>
</tbody>
</table>

latest commit ac9be987ff
Discussion forum

Home

Activity within 7 days: 17 New Messages

Description
Developer group for Yahoo! Search BOSS

Most Recent Messages (View All)

Re: Rate Limit Exceeded ???
Hi Nick, I just emailed ‘you’ asking for some info, please reply to that email. Thanks.

Posted - Mon May 20, 2013 3:53 pm
From: nick c
Lecture 7

• Introduction
• REST
• Data formats
• Security
• Maintenance
• Documentation
• Conclusion
Things to avoid

- Lengthy signup process
- Exposing raw/ugly data
- Complex security model
- Breaking backwards compatibility
- Inaccurate documentation
- Multi-call operations ("chatty APIs")
- Developer frustration
Books

APIs
A Strategy Guide
Daniel Jacobson, Greg Brail & Dan Woods

Web Services for the Real World
Leonard Richardson & Sam Ruby
Forward by David Heineker Hanson
Resources and services

programmableweb

apigee

{"logo":"API Evangelist"}

StatusPage.io

apiphony®

3scale infrastructure for the programmable web