

9 — Web API Design

From Code to Product
gidgreen.com/course

Lecture 9

- **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

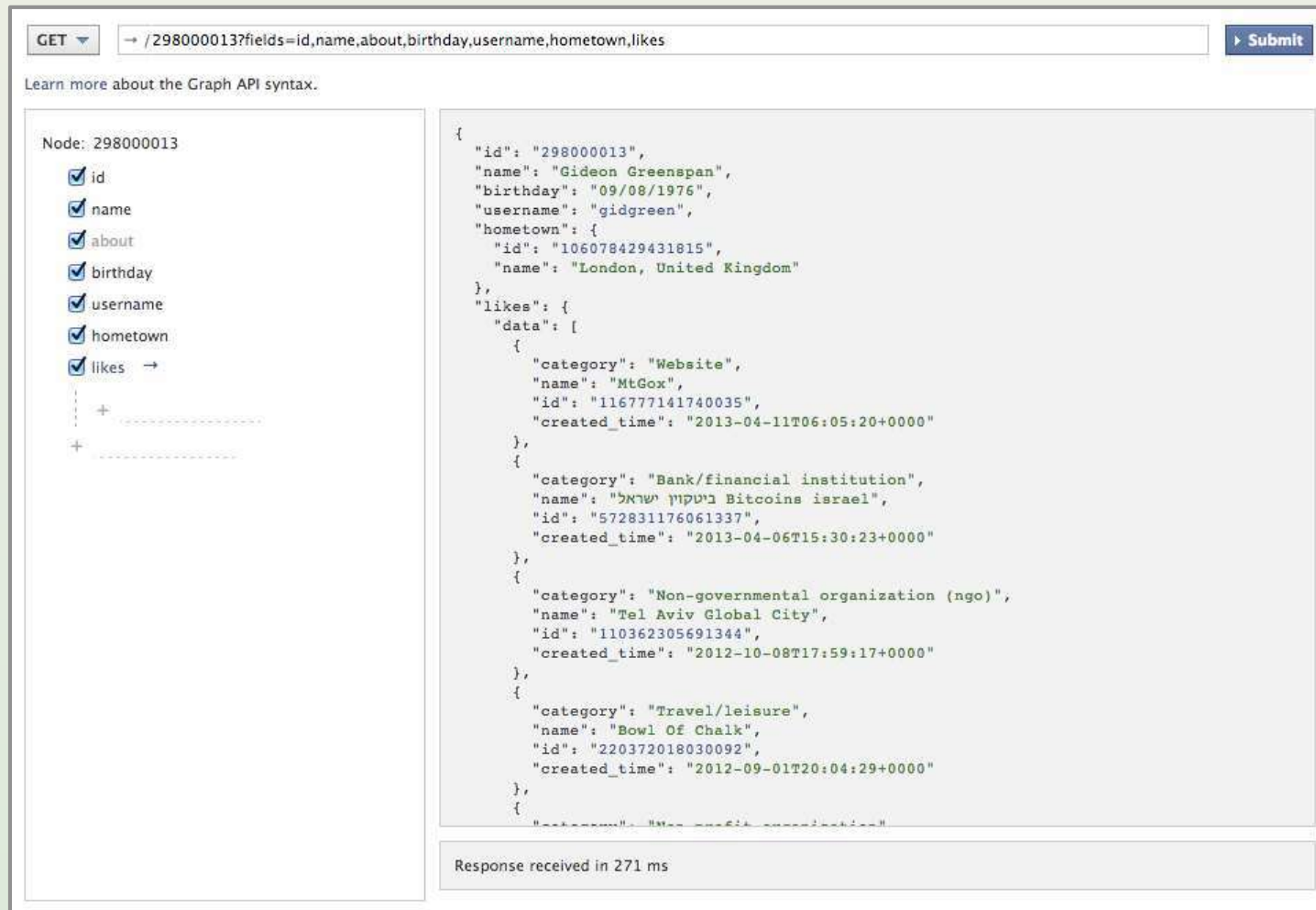
Types of API

- **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**
 - `NPErrror NP_Initialize(...)`
- **Web APIs, e.g. Yahoo! BOSS**
 - `http://yboss.yahooapis.com/ysearch/web?q=API`

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



GET → /29800013?fields=id,name,about,birthday,username,hometown,likes Submit

Learn more about the Graph API syntax.

Node: 29800013

- id
- name
- about
- birthday
- username
- hometown
- likes →
- ⋮ +
- + ⋮

```
{
  "id": "29800013",
  "name": "Gideon Greenspan",
  "birthday": "09/08/1976",
  "username": "gidgreen",
  "hometown": {
    "id": "106078429431815",
    "name": "London, United Kingdom"
  },
  "likes": {
    "data": [
      {
        "category": "Website",
        "name": "MtGox",
        "id": "116777141740035",
        "created_time": "2013-04-11T06:05:20+0000"
      },
      {
        "category": "Bank/financial institution",
        "name": "ישראל ביטקוין Bitcoin Israel",
        "id": "572831176061337",
        "created_time": "2013-04-06T15:30:23+0000"
      },
      {
        "category": "Non-governmental organization (ngo)",
        "name": "Tel Aviv Global City",
        "id": "110362305691344",
        "created_time": "2012-10-08T17:59:17+0000"
      },
      {
        "category": "Travel/leisure",
        "name": "Bowl Of Chalk",
        "id": "220372018030092",
        "created_time": "2012-09-01T20:04:29+0000"
      },
      {
        "category": "Non-profit organization",
        "name": "The Green Foundation",
        "id": "106078429431815",
        "created_time": "2012-09-01T20:04:29+0000"
      }
    ]
  }
}
```

Response received in 271 ms

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
    %3D165953%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
```

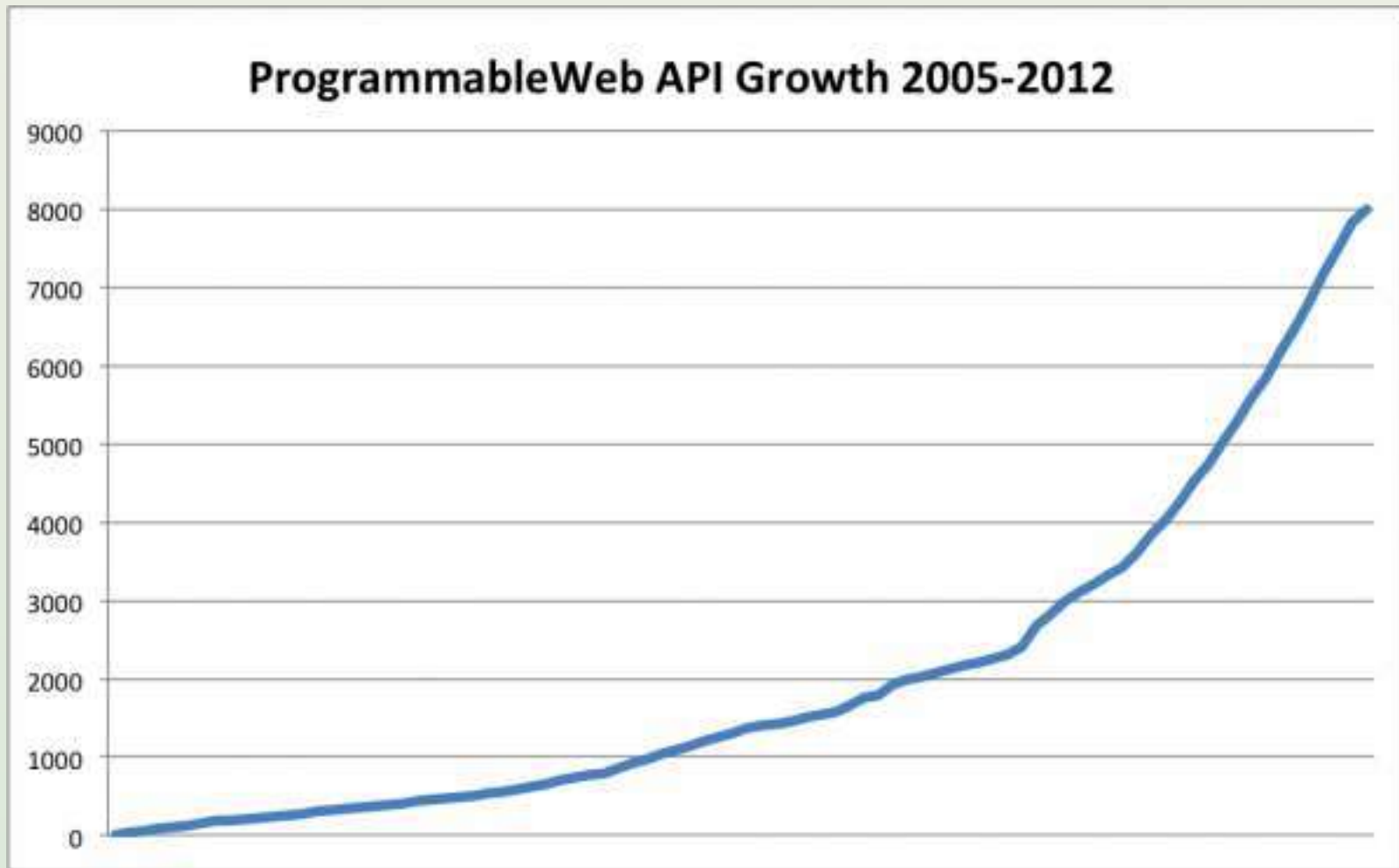
Twitter REST API

POST `https://api.twitter.com/1.1/statuses/update.json`

POST Data `status=Maybe%20he%2711%20finally%20find%20his%20keys.%20%23peterfalk`

```
1. {
2.   "coordinates": null,
3.   "favorited": false,
4.   "created_at": "Wed Sep 05 00:37:15 +0000 2012",
5.   "truncated": false,
6.   "id_str": "243145735212777472",
7.   "text": "Maybe he'll finally find his keys. #peterfalk",
8.   "contributors": null,
9.   "retweet_count": 0,
10.  "id": 243145735212777472,
11.  "in_reply_to_status_id_str": null,
12.  "geo": null,
13.  "retweeted": false,
14.  "in_reply_to_user_id": null,
15.  "place": null,
16.  "source": "<a href=\"http://jason-costa.blogspot.com\" rel=\"nofollow\">My Shiny
App</a>",
17.  "in_reply_to_screen_name": null,
18.  "in_reply_to_status_id": null
19. }
```


Growth in Web APIs



API Billionaires' Club

	13 billion API calls / day <i>(May 2011)</i>
	5 billion API calls / day <i>(April 2010)</i>
	5 billion API calls / day <i>(October 2009)</i>
	1.4 billion API calls / day <i>(May 2012)</i>
	1.1 billion API calls / day <i>(April 2011)</i>
	1 billion API calls / day <i>(May 2012)</i>
	1 billion API calls / day <i>(Q1 2012)</i>
	1 billion API calls / day <i>(January 2012)</i>

[http://blog.programmableweb.com/2012/05/23/
which-apis-are-handling-billions-of-requests-per-day/](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”
- New distribution channels
- 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 forms 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_mkGsLqEW6SLn2a487HYfJVLf: \
-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



The screenshot shows the SendGrid website homepage. At the top left is the SendGrid logo. To the right are links for Demo, Support, Sign In, and Sign Up. Below the logo is a navigation menu with links for PRICING, SOLUTIONS, DEVELOPERS, RESOURCES, PARTNERS, BLOG, and CONTACT SALES. A dark blue banner at the top right displays '107,955,226,689 emails delivered by SendGrid'. The main content area features the headline 'Email Delivery. Simplified.' followed by a paragraph describing SendGrid's cloud-based email infrastructure. Below this is an orange 'Get Started' button and a 'Request a Demo' link. On the right side, a computer monitor displays a video player with the SendGrid logo and a play button, and the URL 'sendgrid.com' below it.

SendGrid

Demo Support Sign In Sign Up

PRICING SOLUTIONS DEVELOPERS RESOURCES PARTNERS BLOG **CONTACT SALES**

107,955,226,689 emails delivered by SendGrid

Email Delivery. Simplified.

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.

[Get Started](#)

[Request a Demo](#)

sendgrid.com

API-only companies: 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?)

Lecture 9

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

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`

```
{  
  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

HTTP command	Database operation	/dogs	/dogs/3
GET	<u>R</u> ead	List all dogs	Get dog details
POST	<u>C</u> reate	Create new dog	—
PUT	<u>U</u> ppdate	—	Update detail/s
DELETE	<u>D</u> elete	Delete all dogs	Delete this dog

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`

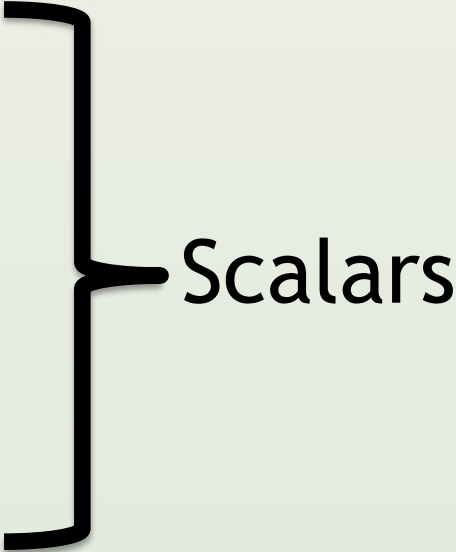
Other protocols

- 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 9

- 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
- 
- Scalars

Extensible Markup Language (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">
    ...
```

- ✓ 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)

```
[
  {
    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)

!	#	\$	&	'	()	*	+
%21	%23	%24	%26	%27	%28	%29	%2A	%2B
,	/	:	;	=	?	@	[]
%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

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

HTTP code	Meaning
200	OK
4xx	Bad request (client's fault)
5xx	Failed request (server's fault)
401	Unauthorized request
404	Resource not found
500	Internal error (bug)
503	Server overloaded

Lecture 9

- 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 QWxhZGRpbjpvGc2FtZQ==
```

- ✓ 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 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 9

- 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

Service / Website	May 21	May 20	May 19	May 18
/statuses/home_timeline (OAuth 1.0a)	✓	⊖	⊖	⊖
favorites	✓	✓	✓	✓
friends/ids	✓	✓	✓	✓
search	✓	✓	✓	✓
stream	✓	✓	✓	✓
users/show	✓	✓	✓	✓
userstream	✓	✓	✓	✓

✓ Service is operating normally ⊖ Performance issues ! Service down

Twitter API Status - Real-time API availability and performance status. This site shows if Twitter APIs are down or have performance issues. [View history.](#)

Public Website Health Status for Twitter API Powered by [Pingdom](#)



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>

3,423 TWEETS 34 FOLLOWING 1,658,192 FOLLOWERS [Follow](#)

Tweets All / No replies

 **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-blackout)
[View summary](#)

 **Twitter API** @twitterapi 13 May
We have deprecated HTTP 1.0 support for the Streaming API: dev.twitter.com/blog/deprecati...
[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

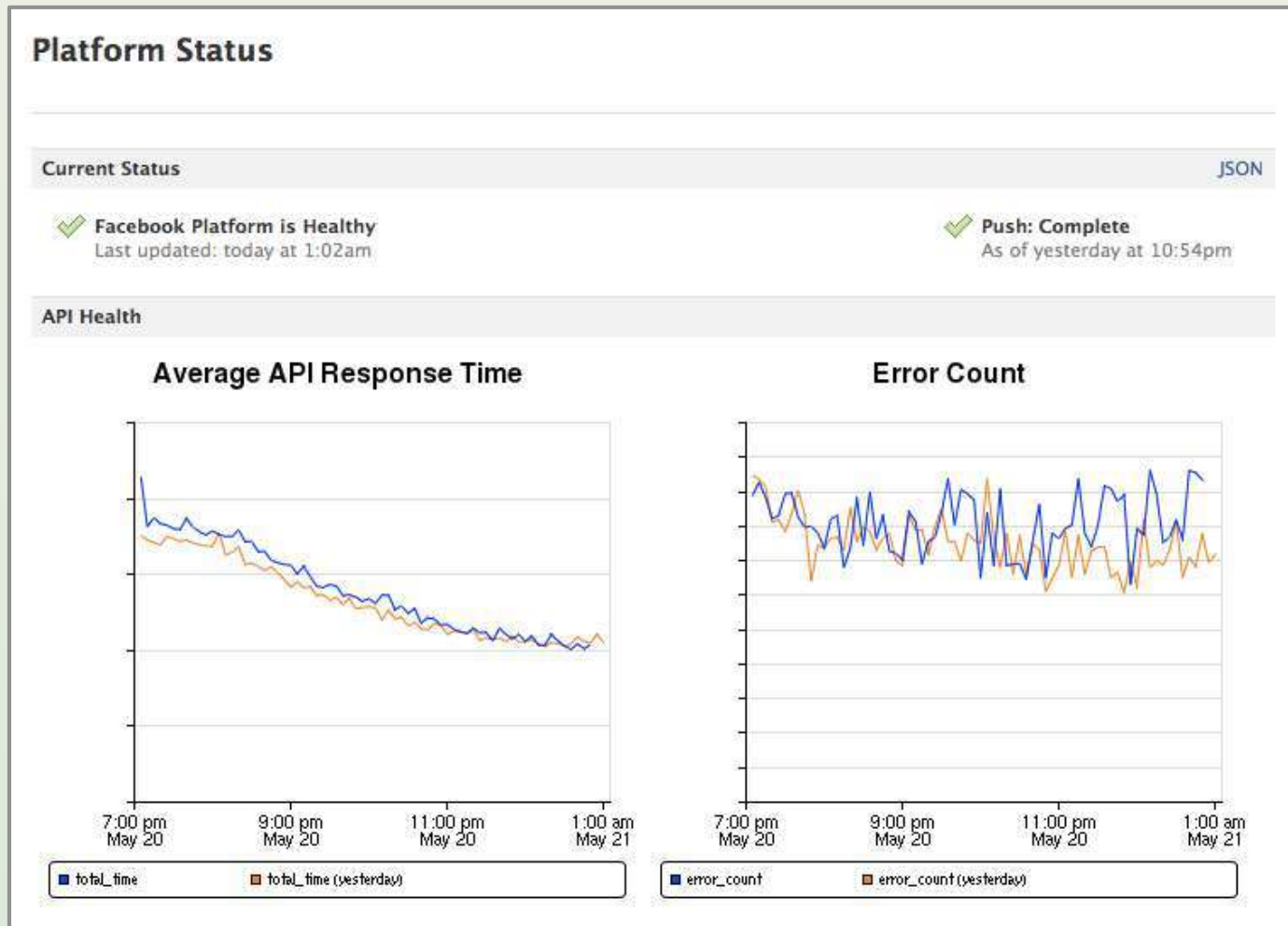
Scaling

- 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



Logging

- Log everything
 - Incoming requests
 - Outgoing response
 - Response time
- To enable...
 - Bug resolution
 - Abuse forensics
 - Deeper (offline) analytics

Lecture 9

- Introduction
- REST
- Data formats
- Security
- Maintenance
- **Documentation**
- Conclusion

Documentation

- Reference
- Examples
- API explorer
- Language libraries
- Example apps
- Discussion forum
- (and support)

Reference: security

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:

Service	Description
web	Yahoo! web search index results with basic url, title, and abstract data.
limitedweb	Limited web results. Index refresh rate ~ 3days.
images	Image search. Image Search includes images from the Yahoo! Image Search index and Flickr.
news	News search. News Search includes late breaking news articles from the past 30 days.
blogs	Blogs search [Beta]
ads	Advertising. If publisher has qualified for Yahoo! Search Advertising.

Reference: input parameters

Universal Yahoo! BOSS API Arguments

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

Argument	Description
service	Web; Limitedweb; Images; News; Blogs services
start	Ordinal position of first result. First position is 0. Default sets start to 0. Note: Start parameter cannot be greater than 1000. All services return a maximum of 1000 results only.
count	Total number of results to return. Maximum default value is 50. Note: <ul style="list-style-type: none">• The maximum default value for images is 35.• Blogs maximum default value is 20.
format	The data format of the response. Value can be set to either "xml" or "json". Default sets format to "json".

For each input parameter

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

Reference: output fields

Response Fields

This section describes the response fields returned by web search.

XML header fields

Field name	Description
abstract	Abstract with keywords highlighted with html tags. abstract=long 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.
title	Returns the result's title, with keywords highlighted with html <code></code> and <code></code> tags.
url	URL of resultReturns the result's URL.
clickurl	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.
dispurl	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
date	Returns the date the URL was last crawled in YYYY/MM/DD format.
language	Displays the language of the search results document. See view=language for more information. Other language results may appear in results.

Reference: response codes

HTTP status codes	Problem Description & Resolution
400 bad request	Un-escaped reserved characters. Something else wrong with request (API). Check your BOSS API parameters.
401	Invalid consumer key in a request
403 invalid AppID error, access denied, Access is forbidden	Possible production code errors and not client-side errors. Please retry your API query.
406 not acceptable	Accept-encoding setting must be set to 'gzip'. Keep-alive must be turned off (cannot reuse port connections).
407	Proxy authentication required. Yahoo! BOSS API does not use proxy authentication. Please check your systems.
408	Your request has timed out. Please try your request again. If this problem persists, please send a message to the BOSS Yahoo! Groups forum .
414	Request-URL Too Large. URL too long for the system to process.
500 Internal server error	Received when server problems occur. Please retry your query. Single letter query may also cause this error to be displayed.
503	<ul style="list-style-type: none">• 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 BOSS Yahoo! Groups forum.
504	Caused by server problems in Yahoo! services. Please retry your query.

Examples

SIMPLE QUERY EXAMPLE

view plain print ?

1. `http://yboss.yahooapis.com/ysearch/web?q=yahoo for xm`

SIMPLE XML RESPONSE

view plain print ?

```
1. <bossresponse responsecode="200">
2.   <web start="0" count="50" totalresults="300">
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.
9.         <dispurl type="default">www.barackobama.com/</dispurl>
10.        <title type="default">Organizing for America | BarackObama.com</title>
11.        <abstract type="default">Organizing for America is the grassroots
12.
```

These
should
match!

API explorer

The screenshot shows the Facebook Graph API Explorer interface. At the top, there is a navigation bar with the Facebook Developers logo, a search bar, and links for Docs, Tools, Support, News, and Apps. The user's name, Gideon Greenspan, is visible in the top right corner.

The main content area is titled "Graph API Explorer" and includes a breadcrumb trail: Home > Tools > Graph API Explorer. On the right, there is a dropdown menu for the application, currently set to "Graph API Explorer".

Below the application dropdown, there is a section for the Access Token. The token is displayed as "CAACEdEose0cBACGtIU0nebHw4Jt0qqI3d5CeBc0nDDHMYrlpcSiksEDdXxKT26I0RSnRchATTuZA4ajGjU8FPBluj7pVky3tl". There are "Debug" and "Get Access Token" buttons next to it.

The interface has two tabs: "Graph API" (selected) and "FQL Query". Under the "Graph API" tab, there is a "POST" dropdown menu and a URL input field containing "https://graph.facebook.com/298000013?fields=id,name". A "Submit" button is located to the right of the URL field.

Below the URL field, there is a field for the "name" parameter, which contains the value "Noedig Napsneerg". There is an "Add a field" link below this field.

Below the parameter field, there is a link that says "Learn more about the Graph API syntax."

At the bottom of the interface, there is a code block showing the JSON response of the failed request:

```
{
  "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, params=params)
    signature_method = oauth.SignatureMethod_HMAC_SHA1()
    req.sign_request(signature_method, consumer, None)
    return rest.load_json(req.to_url())
```

- ✓ Developers save time
- ✓ Get fewer bad API calls
- ✗ You must learn many languages
- ✗ Maintenance


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 /  🔒 2 commits

boss mashup files - initialization

 **jcleblanc** authored 2 years ago latest commit `ac9be987ff`

 templates	2 years ago	boss mashup files - initialization [jcleblanc]
 util	2 years ago	boss mashup files - initialization [jcleblanc]
 yos	2 years ago	boss mashup files - initialization [jcleblanc]
 __init__.py	2 years ago	boss mashup files - initialization [jcleblanc]
 config.json	2 years ago	boss mashup files - initialization [jcleblanc]
 setup.py	2 years ago	boss mashup files - initialization [jcleblanc]

Discussion forum

Home

[Join This Group!](#)

Activity within 7 days: **17** New Messages

Description
Developer group for Yahoo! Search BOSS



Messages Topics

Search: [Advanced](#) [Start Topic](#)

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.
From: nick c

Paymon
[paymon_a](#)
 [Send Email](#)

Posted - Mon May 20, 2013 3:53 pm

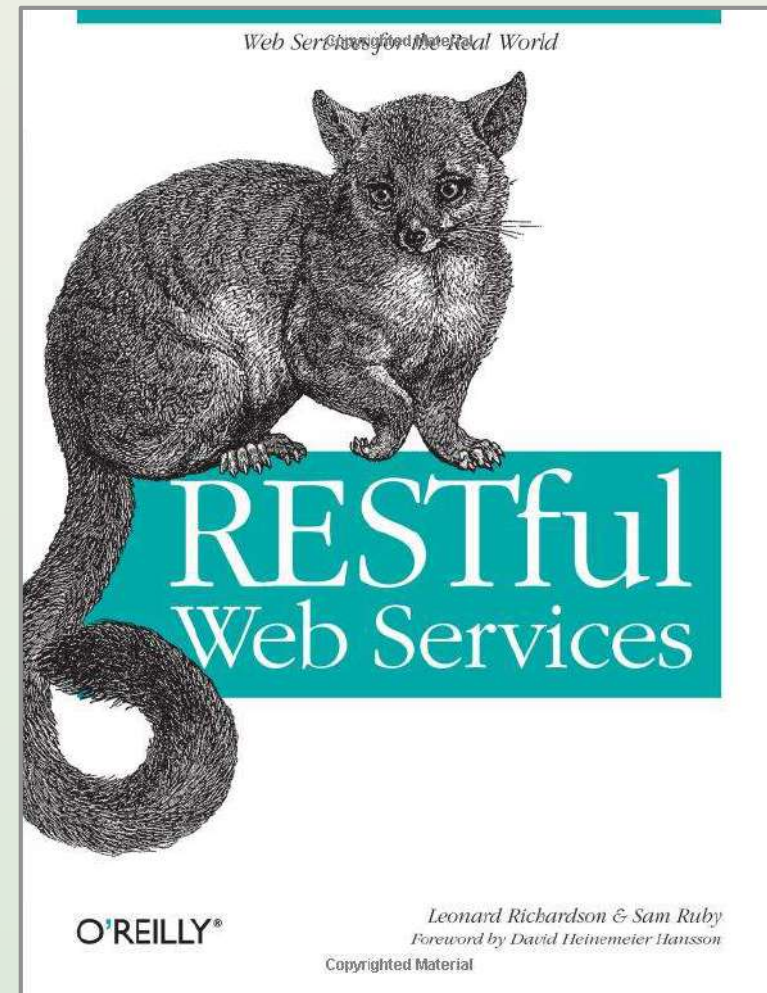
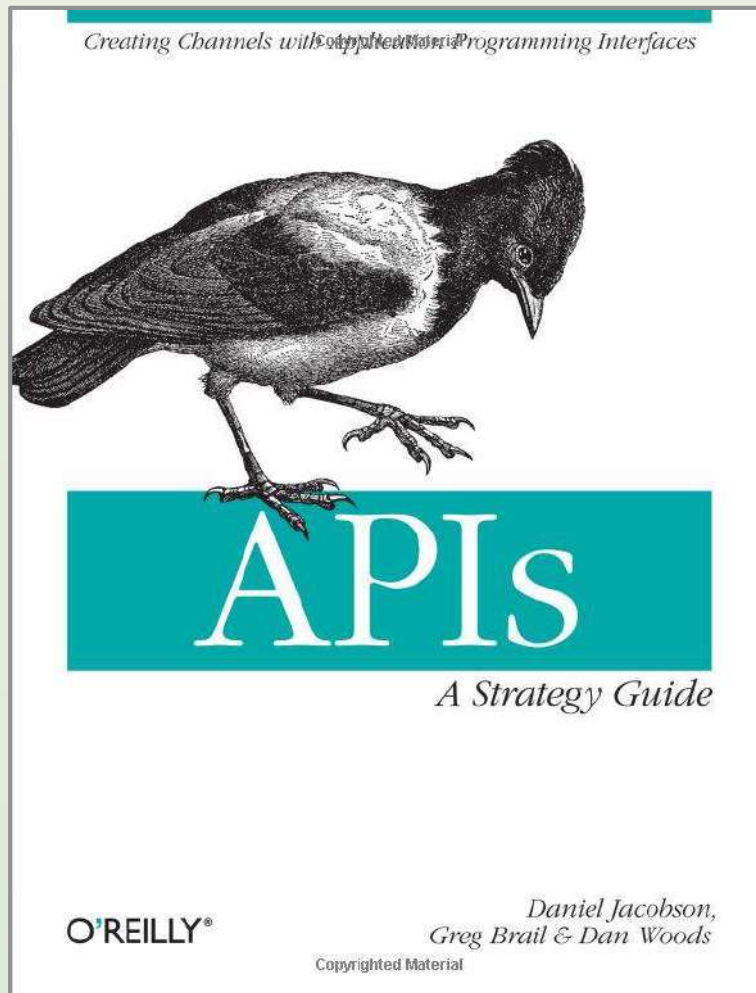
Lecture 9

- 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



Resources and services



```
{"logo": "API Evangelist"}
```



MASHERY