

Google APIs

Pamela Fox

What We'll Talk About

- Google APIs
- Google Data APIs Protocol
 - JSON
 - Google Calendar API
 - Google Spreadsheets API
- Google Gears

Google APIs

Our mission: “Organize the world’s information and make it universally accessible and useful”

Most of our APIs allow you programmatic access to a already existent Google products, and represent a subset of what that product can do

30 Google APIs on <http://code.google.com...> and counting!

- Google Account Authentication
- AdSense API
- AdWords API
- Google AJAX Search API
- Google Base Data API
- Blogger Data API
- Google Calendar Data API
- Google Code Search Data API
- Google Data APIs
- Google Desktop SDK
- Google Earth KML
- Google Gadgets API
- Gmail Atom Feeds
- Google Apps APIs
- Google Checkout API
- Google Web Toolkit
- Google Groups Feeds
- Google Maps API
- Google News Feeds
- Google Notebook Data API
- **Picasa Web Albums Data API**
- Google Related Links
- Google Search Appliance APIs
- Google Search History Feeds
- Google Sitemaps
- Google Spreadsheets Data API
- Google Talk XMPP
- Google Toolbar API
- Google SOAP Search API
- **YouTube API**

Google APIs: Various Types

We offer a variety of APIs, here's a loose categorization of some:

- SOAP
 - AdSense
 - AdWords
 - Search ('deprecated' – no more developer keys given out)
- JavaScript
 - Google Maps API
 - Google Ajax Search
 - Google Gadgets
- XML Standards:
 - KML (Google Earth Markup Language)
 - Sitemap protocol (inform search engines about URLs on your websites that are available for crawling)
 - ATOM (Google Data APIs Protocol) –
 - Google Calendar, Google Base, Google Spreadsheets, GMail, Google Groups, Picasa Web Albums
 - Client Libraries available for Python, PHP, Java, Objective C, JavaScript
 -

The REST/SOAP APIs can be accessed by many languages, using official Google client libraries (jJava, Python, PHP, Objective C, JavaScript), or by writing your own wrapper in your favorite language.

The REST APIs provide write access through server-side scripts, except in the case of the JavaScript client library which uses an iframe hack to allow cross-domain writes.

Google Data APIs Protocol: Motivations

Google's mission is to organize the world's information and make it universally accessible and useful. Sometimes making information accessible requires making it available in contexts other than a web browser. Thus, Google provides APIs to let client software request **information outside of a browser context.**

The Google Data APIs protocol provides a general model for **feeds, queries, and results.** You can use it to send queries and updates to any service that has a Google Data API interface.

Syndication is an effective and popular method for providing and aggregating content. The Google Data APIs protocol provides a way to **expand the types of content that Google can make available through syndication**; in particular, it lets you use the syndication mechanism to send queries and receive query results.

Google Data APIs Protocol: Google Products that Support It

Google Base

Blogger

Google Calendar

Google Code Search

Google Notebook

Doc List/Spreadsheets

Picasa Web Albums

YouTube

...and more!

Note: Some of these are read-only.

Google Data APIs Protocol: Use Cases

Base

- Inventory systems
- classifieds
- personals
- public listing service

Calendar

- Migration
- Synchronization
 - Other calendar systems
 - Mobile devices
- Public calendars
- Internal applications
- Desktop applications

Picasa

- Migration from Flickr!
- Publishing to your website

Spreadsheets

- Pretty much anything that could be done with a database

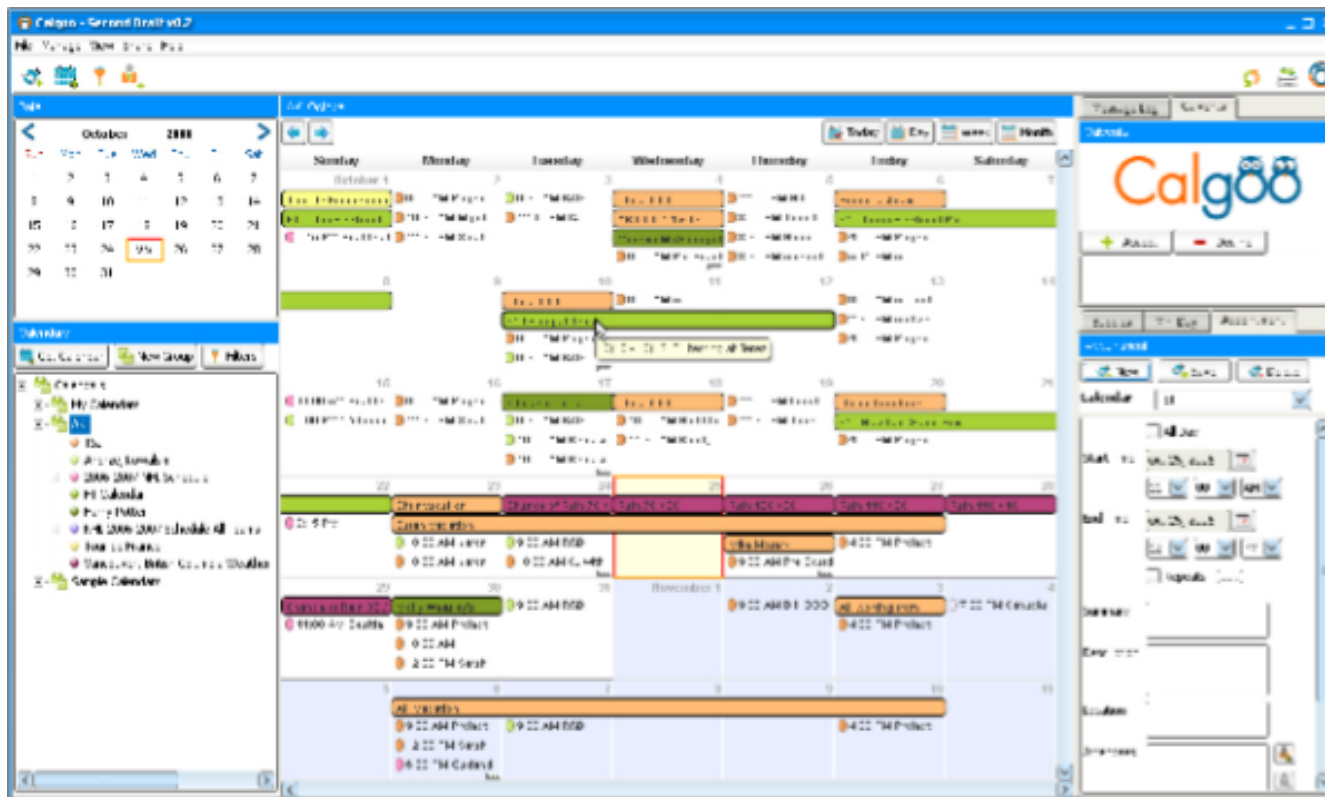
Google Data APIs Protocol: Use Cases

Often, a developer will create an app that compensates for something that the Google product is lacking, at least from their perspective.

Google creates the application with the core functionality, and through the APIs, allows developers to specialize it for their needs.

i.e. Google does the hard stuff, developers do the fun stuff ☺

Case in point – Calgoo: “Calgoo is free online/offline software that allows you to: Bring together iCal, Outlook and Google calendars in one view, **Use your calendars offline** and sync them when you go online”



Google Data APIs Protocol: Introduction

The Google data APIs protocol provides a simple standard protocol for reading and writing data on the web.

The Google Data APIs protocol is a standards-based protocol that uses Atom and RSS as XML-based formats for feeds and data interchange

Atom 1.0

- Allows for extensions
- Allows different type attribute values (html/text/media/etc) for elements
- Is associated with a schema (can be validated)
- Allows for elements to be used elsewhere
- Uses xml:lang for localization

Atom Publishing Protocol (AtomPub) –

- Defines PUT/POST/DELETE operations

RSS 2.0

- Commonly accepted syndication format

	Google Data APIs Protocol	Atom / AtomPub	RSS
Syndication Format	*	*	*
Updates	*	*	
Authentication	*		
Queries	*		
Optimistic Concurrency	*		

Google Data APIs Protocol: Comparing RSS 2.0 to ATOM 1.0

- Colored elements are shared across the feeds
- Most elements differ by name, but convey the same/similar content

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
<channel>
<title>Example Feed</title>
<description>Witty remark here</description>
<link>http://example.org/</link>
<lastBuildDate>Sat, 13 Dec 2003 18:30:02
GMT</lastBuildDate>
<managingEditor>johndoe@jd.com (John Doe)
</managingEditor>
<item>
<title>Atom-Powered Robots Run Amok</title>
<link>http://example.
org/2003/12/13/atom03</link>
<guid isPermaLink="false">
urn:uuid:1225c695-cfb8-4ebb-aaaa-
80da344efa6a</guid>
<pubDate>Sat, 13 Dec 2003 18:30:02
GMT</pubDate>
<description>I put up a fierce fight with the
atom-powered robots from Smith County last
Tuesday. Unfortunately, their atoms were more
powerful than mine and I was defeated.
</description>
</item>
</channel>
</rss>
```

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
<title>Example Feed</title>
<subtitle>Insert witty remark here</subtitle>
<link href="http://example.org/" />
<updated>2003-12-13T18:30:02Z</updated>
<author>
<name>John Doe</name><email>johndoe@example.com</email>
</author>
<id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
<entry>
<title>Atom-Powered Robots Run Amok</title>
<link href="http://example.org/2003/12/13/atom03" />
<id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
<updated>2003-12-13T18:30:02Z</updated>
<summary>John describes his run-in with the robots</summary>
<content type="text">I put up a fierce fight with the atom-
powered robots from Smith County last Tuesday.
Unfortunately, their atoms were more powerful than mine and
I was defeated.</content>
</entry>
</feed>
```

Google Data APIs Protocol: Comparing RSS 2.0 to ATOM 1.0

Feed Schema Item	RSS Representation	Atom Representation
Feed Title	<code>/rss/channel/title</code>	<code>/feed/title</code>
Feed ID	<code>/rss/channel/atom:id</code>	<code>/feed/id</code>
Feed HTML Link	<code>/rss/channel/link</code>	<code>/feed/link[@rel="alternate"] [@type="text/html"]@href</code>
Feed Description	<code>/rss/channel/description</code>	<code>/feed/subtitle</code>
Feed Language	<code>/rss/channel/language</code>	<code>/feed/@xml:lang</code>
Feed Copyright	<code>/rss/channel/copyright</code>	<code>/feed/rights</code>
Feed Author	<code>/rss/channel/managingEditor</code>	<code>/feed/author/name /feed/author/email</code>
Feed Last Update Date	<code>/rss/channel/lastBuildDate</code> (RFC 822 format)	<code>/feed/updated</code> (RFC 3339 format)

Google Data APIs Protocol: Element Collections (“Kinds”)

It's often useful for services that use the Google data APIs to be able to provide a consistent set of information (elements) about a particular kind of item.

Google Data APIs Protocol Kinds:

- Contact - Represents a contact: a person, a venue such as a club or a restaurant, or an organization.
- Message - Represents a message, such as an email, a discussion group posting, or a comment.
- Event - Represents a calendar event. The event location is represented by a Contact kind embedded in (or linked from) a <gd:where> element; the event planners and attendees are represented as Contact kinds embedded in (or linked from) <gd:who> elements.

Some of the elements in a “Kind” are ordinary Atom or RSS elements; others are defined by Google in a namespace called the “Google data namespace.” (<http://schemas.google.com/g/2005>)

- <http://code.google.com/apis/gdata/elements.html>

Google Data APIs Protocol: The Event “Kind”

atom:author	Person who created this event	gd:transparency?	(opaque transparent)
atom:category*	Categories	gd:visibility?	(confidential default private public)
atom:content	Description of event	gd:when*	RFC 3339
atom:link*	Link to alt html rep, 'self' xml for entry,etc	gd:when/ gd:reminder*	(eg 15 minutes)
atom:title	Title of event	gd:where*	Location
gd:comments?	Feed or link to feed of the message “kind”	gd:who*	email, attendee organizer, name
gd:eventStatus	(canceled planned tentative)	gd:who/ gd:attendeeStatus?	(canceled confirmed tentative)
gd:recurrence?	RFC 2445 iCalendar content	gd:who/ gd:attendeeType?	(required optional)

Google Data APIs Protocol: Event “Kind” Example

```
<entry xmlns:gd="http://schemas.google.com/g/2005">
<category scheme="http://schemas.google.com/g/2005#kind" term="http://schemas.google.com/g/2005#event"/>
<id>http://mycal.example.com/feeds/jo/home/full/e1a2af06df8a5f967b</id>
<published>2005-01-18T21:00:00Z</published>
<updated>2006-01-01T00:00:00Z</updated>
<title>Discuss BazMat API</title>
<content>We will discuss integrating this.</content>
<author>
<name>Ryan Boyd</name>
<email>jo@example.com</email>
</author>
<gd:when startTime='2005-01-18T21:00:00Z' endTime='2005-01-18T22:00:00Z'>
<gd:reminder minutes='15' />
</gd:when>
<gd:where valueString='Building 41, Room X' />
<gd:eventStatus value="http://schemas.google.com/g/2005#event.confirmed"/>
<gd:visibility value="http://schemas.google.com/g/2005#event.public"/>
<gd:transparency value="http://schemas.google.com/g/2005#event.transparent"/>
</entry>
```

Google Data APIs Protocol: **Service Specific Elements**

Each Google product can have service-specific elements

- **Calendar**

- **xmlns:gCal='http://schemas.google.com/gCal/2005'**
- **gCal:accesslevel**
 - none|read|freebusy|contributor|owner
- **gCal:color**
 - Hex RGB value
- **gCal:hidden**
 - true|false
- **gCal:selected**
 - true|false
- **gCal:timezone**
 - Time zone ID (eg America/Los_Angeles)

Google Data APIs Protocol: HTTP Verbs / Methods and Usage

The Google Data APIs protocol is RESTful, all updates and queries are done over HTTP by specifying the HTTP protocol, query URI, and content (for POST/PUT).

GET – XML or JSON output!

- Retrieve a <feed> or <entry>

POST

- Create a new <entry>

PUT

- Update an existing <entry>

DELETE

- Delete an existing <entry>

Google Data APIs Protocol: Using GET

- Use GET on feed URI

```
GET /myFeed
```

- Response includes meta-data for feed, even though no entries exist yet

```
200 OK
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
<title>Foo</title>
<updated>2006-01-23T16:25:00-08:00</updated>
<id>http://www.example.com/myFeed</id>
<author> <name>Jo March</name> </author>
<link href="/myFeed" rel="self"/>
</feed>
```

Google Data APIs Protocol: Using POST to Create Entry

- Use POST, include required user-defined elements in body

```
POST /myFeed
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom">
<author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author>
<title type="text">Entry 1</title>
<content type="text">This is my entry</content>
</entry>
```

- Notice response creates an ID, edit URI, and updated element in the entry

```
201 CREATED
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom">
<id>1</id>
<link rel="edit" href="http://example.com/myFeed/1/1/" />
<updated>2006-01-23T16:26:03-08:00</updated>
<author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author>
<title type="text">Entry 1</title>
<content type="text">This is my entry</content>
</entry>
```

Google Data APIs Protocol: Using PUT to Update

- Use PUT with complete entry as body (with desired changes)

```
PUT /myFeed/1/1/  
<?xml version="1.0"?>  
<entry xmlns="http://www.w3.org/2005/Atom">  
<id>1</id>  
<link rel="edit" href="http://example.com/myFeed/1/1/" />  
<updated>2006-01-23T16:26:03-08:00</updated>  
<author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author>  
<title type="text">Entry 1</title>  
<content type="text">This is my first entry</content>  
</entry>
```

- Notice entry's edit URI changes in response

```
200 OK  
<?xml version="1.0"?>  
<entry xmlns="http://www.w3.org/2005/Atom">  
<id>1</id>  
<link rel="edit" href="http://example.com/myFeed/1/2/" />  
<updated>2006-01-23T16:26:03-08:00</updated>  
<author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author>  
<title type="text">Entry 1</title>  
<content type="text">This is my first entry</content>  
</entry>
```

Google Data APIs Protocol: Using DELETE

- Use DELETE on the entry's edit URI

```
DELETE /myFeed/1/2/
```

```
200 OK
```

Google Data APIs Protocol: Using GET with a Query (“Searching!”)

- Basic query. Query types vary per Google product.

```
GET /myFeed?q=This
```

- Returns any matching entries in the feed.

```
200 OK
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Foo</title>
  <updated>2006-01-23T16:26:03-08:00</updated>
  <id>http://www.example.com/myFeed</id>
  <author> <name>Jo March</name> </author>
  <link href="/myFeed" rel="self"/>
  <entry>
    <id>1</id>
    <link rel="edit" href="http://example.com/myFeed/1/1/" />
    <updated>2006-01-23T16:26:03-08:00</updated>
    <author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author>
    <title type="text">Entry 1</title>
    <content type="text">This is my entry</content>
  </entry>
</feed>
```

Google Data APIs Protocol: JSON Output - Alternative to XML

- The GData APIs can provide read-only feed data in JSON format as well as in Atom and RSS formats.
- Specifying alt=json in the query will convert ATOM to JSON output.
- <http://code.google.com/apis/gdata/json.html>

JSON



- Stands for Javascript Object Notation
- Lightweight data interchange format
- Simple: Easy to parse, easy to output
- Simple plain text format, can be read in any language
- Smaller than XML, transmits much more quickly

JSON Simple Example

Hierarchical key/value pairs

- Key can be with/without quotes (means same thing)
- Value can be string, number, boolean, or array

Resembles objects/hash tables/structs of programming languages

```
{ "firstName": "John",  
  "lastName": "Smith",  
  "address": {  
    "streetAddress": "21 2nd Street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": 10021  
  },  
  "phoneNumbers": [  
    "212 732-1234",  
    "646 123-4567" ]  
}
```

Javascript: The J in JSON

The reason JSON is so great for mashups is that it is Javascript code itself

Javascript code can be “inserted” in a page using a script tag (or using `eval(JSON)`)

XML has to be retrieved via a XMLHttpRequest after the page loads, and has to be on the same server

So JSON can be used to grab data from other servers, without the use of server-side proxy

Now anyone can mashup with APIs that provide JSON output, just with a Google Pages account (or similar host)

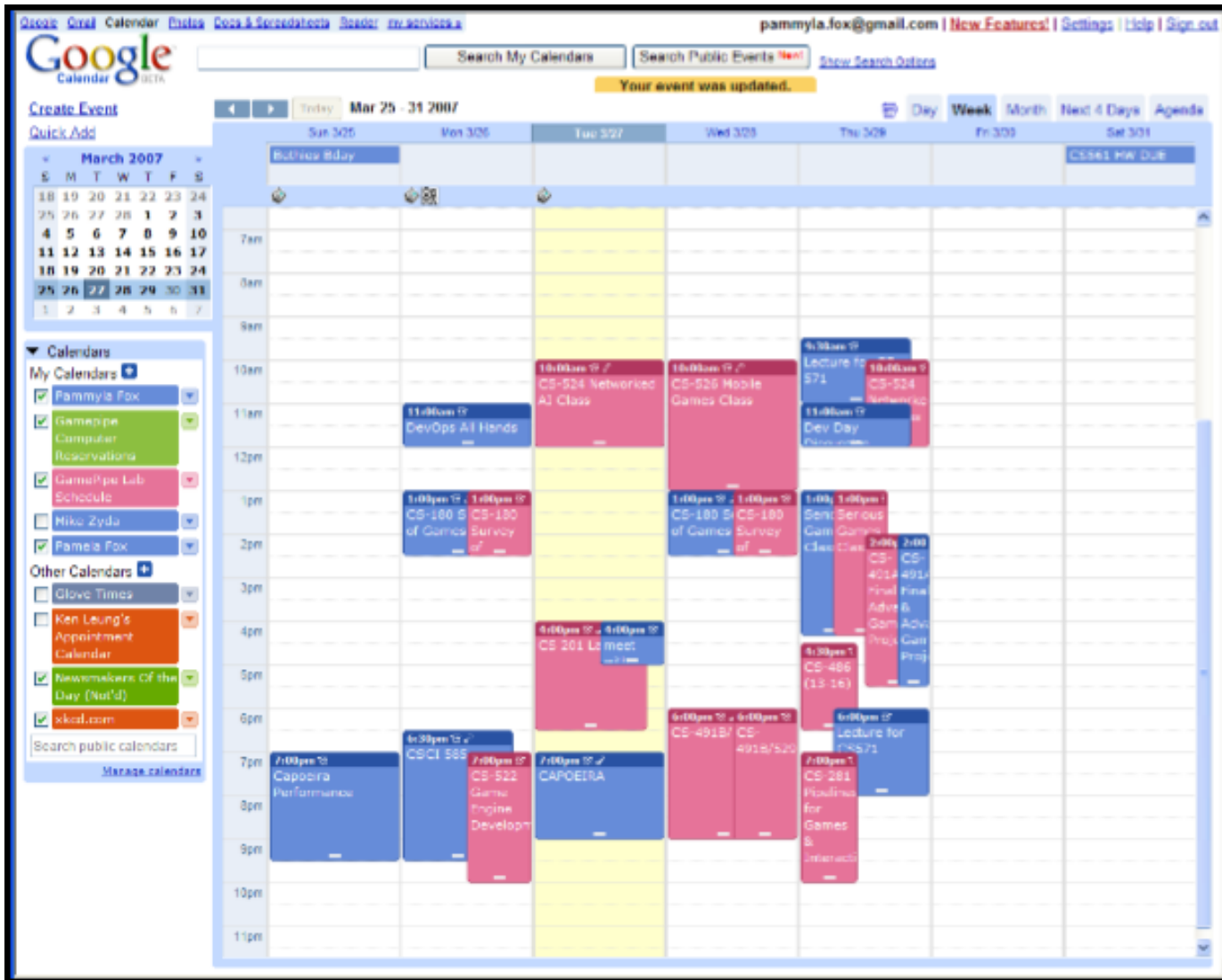
JSON: Google Spreadsheets Example

To read the JSON into the page, we include it as the source for a script tag and assign a callback function. In the call back function, we iterate through the array in entry and read the values of some of the objects.

```
"feed": {
  "entry": [{
    {"title":
    {"type": "text", "$t": "Make google gadget
w/spreadsheet example"},
    "content": {"type": "text", "$t": "Status: Done"}},
    {"title":
    {"type": "text", "$t": "Do final project for class"},
    "content": {"type": "text", "$t": "Status: NotStarted"}
  ]}]
```

```
<script>
function listTasks(root) {
  var feed = root.feed;
  var entries = feed.entry || [];
  var html = [''];
  html.push('<ul>');
  for (var i = 0; i < feed.entry.length; ++i)
  {
    var entry = feed.entry[i];
    var title = entry.title.$t;
    var content = entry.content.$t;
    html.push('<li>', title, ' (' , content, ')
</li>');
  }
  html.push('</ul>');
  document.getElementById("agenda").innerHTML
= html.join("");
}
</script>
<script src="http://spreadsheets.google.
com/feeds/list/
o03712292828507838454.
2635427448373779250/od6/
public/basic?alt=json-in-
script&callback=listTasks"> </script>
```

Google Calendar Data API



Google Calendar lets you view multiple calendars, share calendars, create events, and more. The API provides programmatic access.

Google Calendar Data API: Overview

Functionality available via Google Data APIs protocol

- List a user's calendars
- Get a user's full calendar
 - Retrieve entries in a specific order
 - Query for certain entries in a calendar
 - Retrieve comments in those entries
- Add an entry
 - Regular event
 - Web Content event
- Make changes to an entry
- Delete an entry

Google Calendar Data API: Feed Types

Visibility

- public (only accessible if calendar user enables sharing)
- private (requires authentication)
- private (Does not require authentication. Instead, authentication information is embedded within the feed URI in the magicCookie string. The magic cookie is a random, high-entropy string combined with authentication information.)

Projection

- basic
- full, full-noattendees
- composite
- free-busy
- composite
- attendees-only

Template URL:

<http://www.google.com/calendar/feeds/userID/visibility/projection>

Google Calendar Data API: Feed Types Examples

Template URL:

<http://www.google.com/calendar/feeds/userID/visibility/projection>

Private visibility, Full Projection

- <http://www.google.com/calendar/feeds/default/private/full>

Public visibility, Basic Projection

- <http://www.google.com/calendar/feeds/caltest@ryguy.com/public/basic>

Private visibility, Full Projection (w/magic cookie)

- <http://www.google.com/calendar/feeds/caltest@ryguy.com/private-3977fbab26d5e28a9f/full>

Google Calendar Data API: Special Query Parameters

In addition to the standard Google Data APIs protocol query parameters, you can use these calendar specific ones

Parameter	Meaning	Notes
futureevents	A shortcut to request all events that are scheduled for future times. Overrides the recurrence-expansion-start, recurrence-expansion-end, start-min, and start-max values.	Valid values are true (return all future events) or false (ignore this parameter). Default is false.
orderby	Specifies order of entries in a feed.	Currently, the only supported values are lastmodified (the default) and starttime. If you add orderby=lastmodified as a query parameter, then the returned feed's entries are ordered by their <updated> values, which is the default ordering. If you add orderby=starttime as a query parameter, then the returned feed's entries are in order by the <gd:when> element's starttime attribute.
recurrence-expansion-start	Specifies beginning of time period for which to expand recurring events.	<ul style="list-style-type: none">• Use the RFC 3339 timestamp format. For example: 2005-08-09T10:57:00-08:00.• The lower bound is inclusive, whereas the upper bound is exclusive.
recurrence-expansion-end	Specifies end of time period for which to expand recurring events.	
singleevents	Indicates whether recurring events should be expanded or represented as a single event.	Valid values are true (expand recurring events) or false (leave recurring events represented as single events). Default is false.
sortorder	Specifies direction of sorting.	Valid values are ascending (with synonyms ascend and a) and descending (with synonyms descend and d).
start-min	Earliest event start time to match. If not specified, default is 1970-01-01.	<ul style="list-style-type: none">• Use the RFC 3339 timestamp format. For example: 2005-08-09T10:57:00-08:00.• The lower bound is inclusive, whereas the upper bound is exclusive.• Events that overlap the range are included.

Google Calendar Data API: Creating an Event (Raw HTTP)

Notice authorization string and use of elements from the event “kind”

```
POST /calendar/feeds/default/private/full HTTP/1.1
Host: www.google.com
Cookie: S=calendar=pzQaa6q-o70
Content-Type: application/atom+xml
Authorization: GoogleLogin auth=DQAAAGsAAAD-SG01E3
Content-Length: 438
<?xml version="1.0"?>
<entry xmlns='http://www.w3.org/2005/Atom'
xmlns:gd='http://schemas.google.com/g/2005'>
<title type='text'>Tennis with Beth</title>
<content type='text'> Quick lesson
</content>
<author>
<name>Ryan Boyd</name>
<email>caltest@ryguy.com</email>
</author>
<gd:when startTime='2007-12-16T06:00:00.000-08:00'
endTime='2007-12-16T07:00:00.000-08:00' />
</entry>
```

Google Calendar Data API: Creating an Event (Java Lib)

Google provides client libraries in various languages (Java, Python) for developers who don't want to deal with creating the XML and issuing the HTTP requests

```
URL feedUrl = new URL("http://www.google.com/calendar/feeds/caltest@ryguy.com/private/full");

EventEntry myEntry = new EventEntry();
myEntry.setTitle(new PlainTextConstruct("Tennis with Beth"));
myEntry.setContent(new PlainTextConstruct("Quick lesson"));
Person author = new Person("Ryan Boyd", null, "caltest@ryguy.com");
myEntry.getAuthors().add(author);

DateTime startTime =
DateTime.parseDateTime("2007-12-16T06:00:00-08:00");
DateTime endTime =
DateTime.parseDateTime("2007-12-16T07:00:00-08:00");
When eventTimes = new When();
eventTimes.setStartTime(startTime);
eventTimes.setEndTime(endTime);
myEntry.addTime(eventTimes);

CalendarService myService = new CalendarService("ryanboyd-powerpoint-v1");
myService.setUserCredentials("caltest@ryguy.com", "mypassword");
EventEntry insertedEntry = myService.insert(postUrl, myEntry);
```

Google Calendar Data API: Example Mashups



- RoboCal:
 - Reads your Google Calendar and uses Voxeo to tell you about your events over the phone, allowing you to navigate through events using the keypad and even leave voice notes

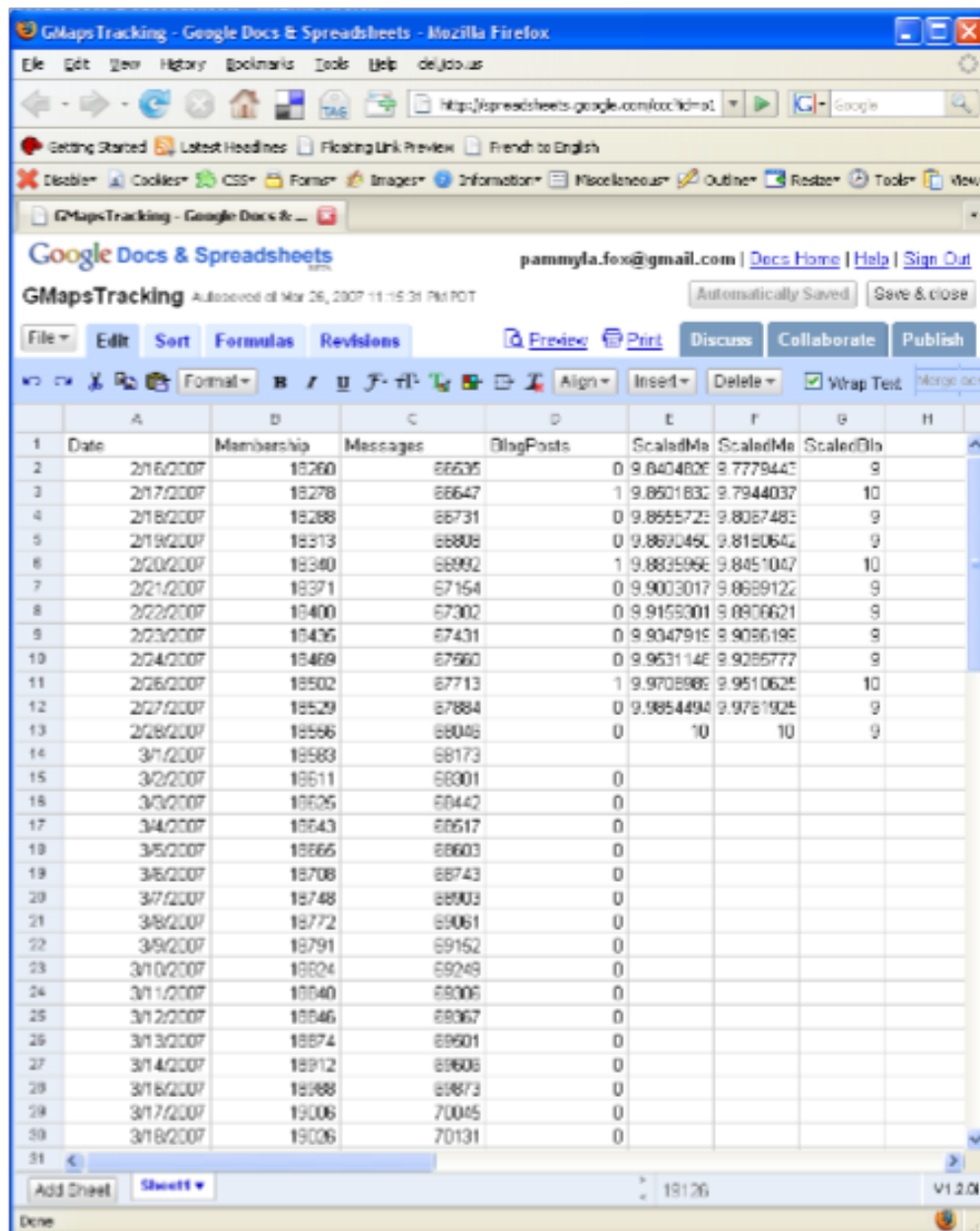
- Newsmakers Of the Day:
 - Cron job at 3am every night sends CNN news through Clearforest Semantic Web Service, calculates top person/company/product/city/country of the day, and inserts a web content event for the day

More at:

<http://programmableweb.com/api/GoogleCalendar/mashups>



Google Spreadsheets Data API



The screenshot shows a Google Spreadsheet titled "GMapsTracking" with the following data:

	A	B	C	D	E	F	G	H
1	Date	Membership	Messages	BlogPosts	ScaledMe	ScaledMe	ScaledBlh	
2	2/16/2007	15260	6635		0.8404826	3.7779447	9	
3	2/17/2007	15278	6647		1.9.8501832	3.7944037	10	
4	2/18/2007	15288	66731		0.9.855722	3.8087483	9	
5	2/19/2007	15313	66808		0.9.8620456	3.8180642	9	
6	2/20/2007	15340	66992		1.9.8825956	3.8451047	10	
7	2/21/2007	15371	67154		0.9.9003017	3.8689122	9	
8	2/22/2007	15400	67302		0.9.9155001	3.8906621	9	
9	2/23/2007	15435	67431		0.9.9347915	3.9056195	9	
10	2/24/2007	15469	67560		0.9.9531146	3.9285777	9	
11	2/25/2007	15502	67713		1.9.9708985	3.9510625	10	
12	2/27/2007	15529	67884		0.9.9854494	3.9761925	9	
13	2/28/2007	15566	68045		10	10	9	
14	3/1/2007	15583	68173					
15	3/2/2007	15611	68301	0				
16	3/3/2007	15625	68442	0				
17	3/4/2007	15643	68517	0				
18	3/5/2007	15665	68603	0				
19	3/6/2007	15708	68743	0				
20	3/7/2007	15748	68803	0				
21	3/8/2007	15772	69061	0				
22	3/9/2007	15791	69152	0				
23	3/10/2007	15824	69248	0				
24	3/11/2007	15840	69305	0				
25	3/12/2007	15846	69367	0				
26	3/13/2007	15874	69601	0				
27	3/14/2007	15912	69608	0				
28	3/15/2007	15988	69673	0				
29	3/17/2007	16006	70045	0				
30	3/18/2007	16026	70131	0				
31								

Google Spreadsheets lets you create rows, add data (inc. formulas), sort data, collaborate with other users, format your data, and more. Google Spreadsheets Data API gives you access to read/write the content of a spreadsheet (not the format).

Google Spreadsheets Data API: Overview

Data organization:

- Resources are organized into a series of feeds
- Data can be expressed in multiple formats (Cell feed, List feed)
- Ask for a different feed, get a different view of the data

Available Feeds

- Spreadsheets
- Worksheets
- List
- Cells

View Parameters

- Visibility (private, public)
- Projection (full, values, basic)

Google Spreadsheets Data API: Example Query

An example query

- Look at the rows in a spreadsheet using the list feed
- To preview XML, try in browser:
 - <http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/>

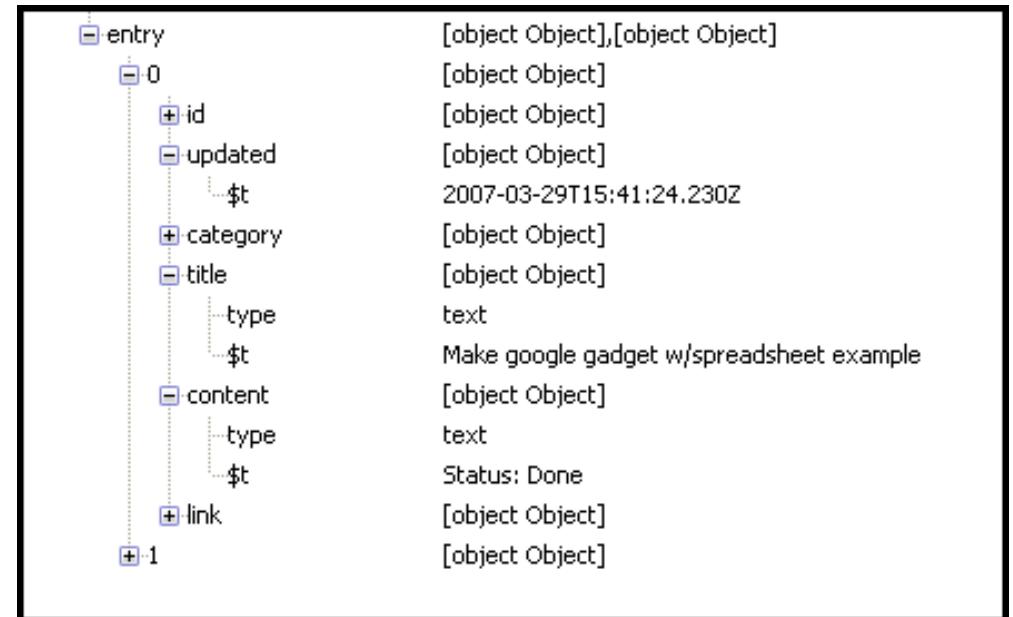
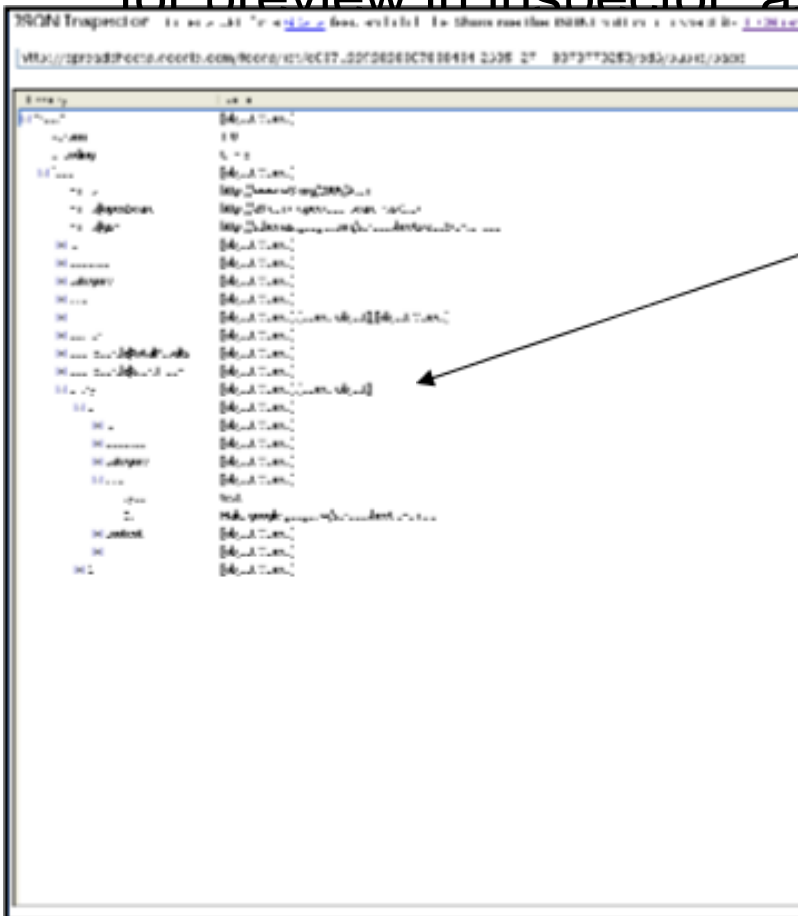
```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:gs="http://schemas.google.com/spreadsheets/feeds/1.0/">
  <title>http://www.w3.org/2005/Atom</title>
  <id>http://www.w3.org/2005/Atom</id>
  <updated>2007-03-29T15:41:24.230Z</updated>
  <category scheme="http://schemas.google.com/spreadsheets/feeds/1.0/">
    <title type="text">Make google gadget w/spreadsheet example</title>
  </category>
  <entry>
    <id>http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/od6</id>
    <updated>2007-03-29T15:41:24.230Z</updated>
    <category scheme="http://schemas.google.com/spreadsheets/feeds/1.0/">
      <title type="text">Make google gadget w/spreadsheet example</title>
    </category>
    <content type="text">Status: Done</content>
    <link rel="self" type="application/atom+xml" href="http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/od6" />
  </entry>
  <entry>
    <id>http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/od6</id>
    <updated>2007-03-29T15:41:24.230Z</updated>
    <category scheme="http://schemas.google.com/spreadsheets/feeds/1.0/">
      <title type="text">Do final project for class</title>
    </category>
    <content type="text">Status: NotStarted</content>
    <link rel="self" type="application/atom+xml" href="http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/od6" />
  </entry>
</feed>
```

```
- <entry>
  <id>http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/od6</id>
  <updated>2007-03-29T15:41:24.230Z</updated>
  <category scheme="http://schemas.google.com/spreadsheets/feeds/1.0/">
    <title type="text">Make google gadget w/spreadsheet example</title>
  </category>
  <content type="text">Status: Done</content>
  <link rel="self" type="application/atom+xml" href="http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/od6" />
</entry>
- <entry>
  <id>http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/od6</id>
  <updated>2007-03-29T15:41:24.230Z</updated>
  <category scheme="http://schemas.google.com/spreadsheets/feeds/1.0/">
    <title type="text">Do final project for class</title>
  </category>
  <content type="text">Status: NotStarted</content>
  <link rel="self" type="application/atom+xml" href="http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/od6" />
</entry>
</feed>
```

Google Spreadsheets Data API: Example JSON Query

To preview JSON, try with JSON data inspector: <http://bolinfest.com/json/inspect.html>

- <http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic?alt=json> (remove the alt for preview in inspector as it auto-adds it)



Google Spreadsheets Data API: Example Mashups

Server-side (XML output, HTTP):

- Asset List checker – reads in spreadsheet, looks for files on server that have prefix according to first column, and suffix according to subsequent columns, for each row, and updates hyperlink if exists
- Automation + Collaboration = Optimal

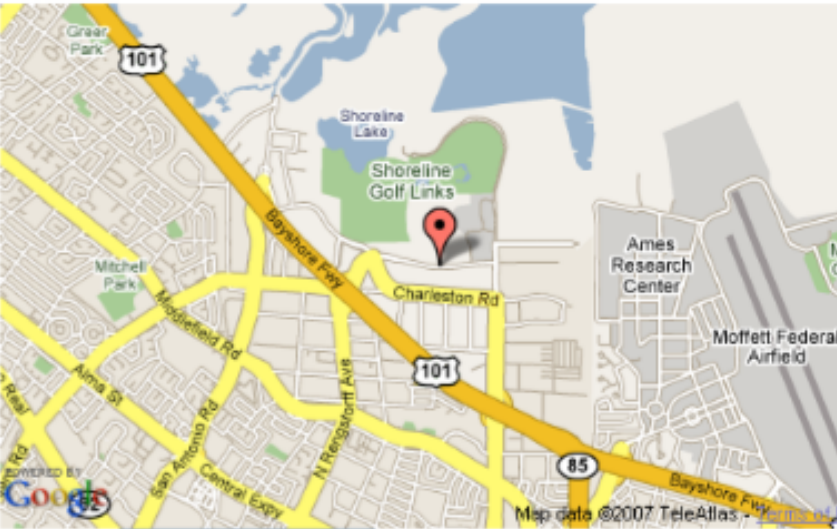
	A	B	C	D	E	F	G	H
	Asset Models	Art URL	Size	DUE DATE	.sc.jpg	texture.jp	.mesh	.materia
3	-CHARACTERS							
4	CLOWN	http://game	6		http://gamepipe.u	X	X	X
5	*throw							
6	*drive							
7	*turn_left							
8	*turn_right							
9	*idle			X				
10	*hit							
11	*nosegrab							
12	*death							
13	BEAR	http://game	10		http://gamepipe.u	X	X	X
14	*attack							
15	*peddle							
16	*turn_left							
17	*turn_right							

Google Spreadsheets Data API: Example Mashups

Server-side (XML output, HTTP):

- Map Signup – Lets people enter in their information and location (using Maps geocoders) and then calls python script that adds their info to spreadsheet

1600 Amphitheatre Pky, Mountain View, CA



Username: pammy

Bio: hi im cool

Pic: bla.jpg

Lat: 37.423021

Lng: -122.083739

GGroupsMembers Autosaved at Mar 12, 2007 9:27:18 PM PDT

	A	B	C	D	E	F	
1	username	bio	url	pic	lat	lng	descrip
2		Google Maps API Support Engineer, MS/BS in Computer Science					
3	pamela	from USC.	http://imagin	http://stati	34.032315	-118.27961	
4	pamela2	delete this soon	http://imagin	fake.jpg	37.21720E	-122.1899.	
5	pammywa	Hi im maps support.	http://imagin	me.jpg	37.423021	-122.0837:	
6	pammylafox	me		me.png	37.423021	-122.0837:	
7	Shnerp	bom		;-) ...another side view	37.420747	-122.0740!	
8	pammylafox2	me		me.png	37.423021	-122.0837:	
9	dot	.		.	37.33527E	-121.8938!	
10	pammylafox2	me		me.png	37.423021	-122.0837:	
11	pammy	hi im cool		test.png	37.423021	-122.0837:	
12	pammy	hi im cool		test.png	37.423021	-122.0837:	
13	pammy	hi im cool		NOT ENTEREC	37.423021	-122.0837:	

Google Gears

Google Gears is an open source browser extension that enables web applications to provide offline functionality using the following 3 JavaScript API components:



LocalServer



Database



WorkerPool



The LocalServer module is a specialized URL cache that the web application controls. Browser requests for URLs in the LocalServer's cache are intercepted and served locally from the user's disk.

**Security! These URLs must be on same domain as the page asking for them to be cached.*

The LocalServer consists of resource stores, of 2 types:

- **ResourceStore**: Used for capturing an array of resource filenames.
- **ManagedResourceStore**: Used for capturing a set of related files defined in a manifest file. Has benefit of re-capturing files when version number in manifest file changes. (Give your users the most recent version!)

http://code.google.com/apis/gears/api_localserver.html



ResourceStore example code snippet:

```
var pageFiles = [
  location.pathname,
  'gears_init.js',
  'foo.html'
];

var localServer = google.gears.factory.create('beta.localserver',
'1.0');

var store = localServer.openStore(this.storeName) ||
localServer.createStore(this.storeName);

store.capture(pageFiles, function(url, success, captureId) {
  console.log(url + ' capture ' + (success ? 'succeeded' :
'failed'));
});
```



The **Database** is used to persistently store an application user's data on the user's computer in the form of a SQLite database (Think of it as a Super-Cache - it only goes away when you want it to!).

**Security! Apps can only open the databases they created.*

Data is stored and retrieved by executing SQL statements.

In addition to standard SQLite commands, the Gears version has an extension called fts2 that allows for full text search in **TEXT** fields using the **MATCH** keyword.



Database sample code snippet that opens a DB and executes SQL statements for create/insert/select:

```
var db = google.gears.factory.create('beta.  
database', '1.0');  
db.open('database-demo');  
db.execute('create table if not exists Demo (Phrase  
varchar(255), Timestamp int)');  
db.execute('insert into Demo values (?, ?)',  
[phrase, currTime]);  
var rs = db.execute('select * from Demo order by  
Timestamp desc');
```



Make your web applications more responsive by performing resource-intensive operations asynchronously

*Not used for adding offline capability, just used for better performing javascript for intensive tasks (e.g. encryption).



Walkthrough of sample that combines Blogger JS Client Lib with Google Gears for an offline blogger:

<http://gdata-javascript-client.googlecode.com/svn/trunk/samples/blogger/bloggears/bloggears.html>

Questions



Google Data APIs Protocol: ATOM to JSON conversion

Basic:

The feed is represented as a JSON object; each nested element or attribute is represented as a name/value property of the object.

Attributes are converted to String properties.

Child elements are converted to Object properties.

Elements that may appear more than once are converted to Array properties.

Text values of tags are converted to \$t properties.

Namespace:

If an element has a namespace alias, the alias and element are concatenated using "\$".

For example, ns:element becomes ns\$element.

XML:

XML version and encoding attributes are converted to attribute version and encoding of the root element, respectively.