CHAPTER 15

Accessing Online Calendars and Event Aggregators

Online calendars will move from being merely trendy to virtually indispensable as our lives move increasingly to the network. Calendaring (scheduling appointments and coordinating calendars) is something most of us can relate to since we all have appointments that we make and keep.

As we use electronic calendars, there is a good chance that we will have more than one calendar to synchronize—people use different calendars or work with people with other calendars, no matter how much Microsoft, Apple, Google, or Yahoo! might want everyone to use its calendar alone. A lot of this calendaring activity has moved to not only digital form but specifically to a networked digital form. In addition to the old calendars, new generations of online calendars are coming into existence—that's the focus of this chapter.

Online calendars exist in the context of other digital calendars: desktop calendars such as Microsoft Outlook and Apple iCal and calendars on handheld devices such as the Palm calendar. Much work has been done on synchronizing these calendars. Of course, calendar synchronization has been operant for a while, but these approaches (specialized conduits/SyncML[1]) have been more opaque than the APIs now available.[2] Today's online calendars with APIs generally make synchronization easier.

In addition to the proliferation of online calendars, event aggregation sites such as Upcoming.yahoo.com and Eventful.com are starting to create a marketplace of event data. They are focused on public events, whereas online calendars have as their focal point individuals and private events. These worlds intersect, of course, because individual users often track the public events they attend on their individual calendars.

When it comes to public events, the point of focus is different, depending on whether you are an attendee (and consumer of information about the event) or are a publisher or purveyor of event information. As an individual viewer, you want to browse, aggregate, and select events, typically from multiple sources. You might be conducting these tasks in a social context. What are your friends interested in? What do they invite you to, and vice versa? Your friends might know what you care about and direct you to events you'll find interesting. As a publisher of events, you probably want to disseminate information about the event as widely as possible. There are technical mechanisms for supporting the interchange of data between publishers of event data and consumers of event data, which is one of the subjects of this chapter.

This chapter shows the first steps to take in learning this subject:

* It covers what data you can get in and out of calendars without programming using iCalendar and various XML feeds as examples.

* It covers how to program individual calendars using Google Calendar and 30boxes.com, how to move data from a source of event data into calendars, and how to write event information to event aggregators such as Upcoming.yahoo.com and Eventful.com.

1. SyncML is now known as Open Mobile Alliance Data Synchronization and Device Management.
2. http://www.coldsync.org/description.html

Google Calendar

Google Calendar is fast increasing in popularity among online calendars.[3] Not only does it have some clever features, but it is highly remixable with its extensive API and use of feeds and excellent data import and export functionality.

Let's talk about how to use Google Calendar as a user first and then look at how to program it.

Setting Up Google Calendar As an End User

Log in to your Google account here:

```
http://calendar.google.com
```

Google Calendar has some noteworthy features:

* In addition to creating a main calendar, you can create secondary calendars and subscribe to calendars belonging to others. Because you can turn the visibility of any given calendar on and off, you get a composite view of the events of all your visible calendars. (Think of each calendar as a layer.) On the sidebar, you get a list of your own calendars and the other calendars to which you are subscribing.
* You can search for public events and look for public calendars.[4] You can also make your events publicly searchable right within your own calendar—tightly coupling the process of publishing and consuming events.
* You can set the visibility of your calendars to one of three options: make it publicly available to everyone; show only the Free/Busy information availability, that is, show only whether a block of time is occupied; or set it to the Do Not Share with Everyone level, in which case the calendar is visible only to those people with whom you explicitly share your calendar.[5]
* To delete a calendar, you have to click the Manage Calendars link.
* There is Gmail/Google Calendar integration: "Gmail users can send event invitations directly from their Gmail accounts without accessing Google Calendar."[6]
* There is currently no direct offline access to Google Calendar.[7]

3. http://www.techcrunch.com/2007/01/04/online-calendar-wiars/
4. http://www.google.com/calendar/render?mode=gallery&cat=POPULAR
5. http://www.google.com/support/calendar/bin/answer.py?answer=34577&hl=en

Some Usage Patterns for Google Calendar

To show some use case scenarios for Google Calendar, here I list some of the calendars that I have set up and the reasons why:

* A strictly personal calendar for events. I have set this calendar to Do Not Share with Everyone.
* A family and friends calendar for my closest friends. I also use the Do Not Share with Everyone setting here but then add the e-mail addresses of individual friends and family members.
* A calendar called Raymond Yee's Public Events to list events that I plan to be at and don't mind the world knowing about. I use the Share All Information on This Calendar with Everyone setting.
* A calendar called Mashup Guide Demo Calendar, a public calendar I'll use in this chapter to demonstrate how to program Google Calendar.

When I create a new Google calendar, I consider the following factors:

* Who I want to share the calendar with (that is, is the calendar for myself, a specific group of people, or for the whole world?)
* The broad topic of that calendar

Sharing Calendars

There are calendar addresses that are visible to others if the calendar is public. There are three formats:[8]

* HTML
* iCalendar (also known colloquially as iCal)[9]
* XML (specifically, Atom feed)

To illustrate the different feed formats, I'll use a publicly available calendar that I created: the Mashup Guide Demo Calendar, whose sharing status I have set to Share All Information on This Calendar with Everyone.

6. http://www.google.com/support/calendar/bin/answer.py?answer=53231&topic=8556

7. http://www.google.com/support/calendar/bin/answer.py?answer=61527&topic=8556

8. http://www.google.com/support/calendar/bin/answer.py?answer=34578&hl=en and http://www.google.com/support/calendar/bin/answer.py?answer=37104&ctx=sibling

9. http://en.wikipedia.org/wiki/ICalendar

Every Google calendar has an *identifier*. The user ID for a user's main calendar is the user's e-mail address. For other calendars, the user ID is a more complicated e-mail address. For instance, the user ID for the Mashup Guide Demo Calendar is as follows:

```
9imfjk71chkcs66t1i436je0s0%40group.calendar.google.com
```

You can get the HTML feed for a calendar here:

```
http://www.google.com/calendar/embed?src={userID}
```

For example:

```
http://www.google.com/calendar/embed?src=9imfjk71chkcs66t1i436je0s0%40group.cale
ndar. ~CCC
google.com
```

Associated with the iCalendar and XML feeds are two parameters (`visibility` and `projection`) that I'll explain in greater detail in a moment. For instance, you can access an iCalendar feed here:

```
http://www.google.com/calendar/ical/{userID}/{visibility}/{projection}.ics
```

For example:

```
http://www.google.com/calendar/ical/9imfjk71chkcs66t1i436je0s0%40group.calendar.
~CCC
google.com/public/full.ics
```

and for example:

```
http://www.google.com/calendar/ical/9imfjk71chkcs66t1i436je0s0%40group.calendar.
~CCC
google.com/public/basic.ics
```

The Atom feeds are found here:

```
http://www.google.com/calendar/feeds/{userID}/{visibility}/{projection}
```

For example:

```
http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40group.calendar
. ~CCC
google.com/public/basic
```

If your calendar is not public, there are still private addresses that other applications can use to access the calendar. Note that you can reset these URLs too in case you want to reset access.[10]

Exploring the Feed Formats from Google Calendar

The Google Calendar API is built upon GData, the RESTful protocol based on the Atom Publishing Protocol (APP) combined with the Google-specific extensions introduced in Chapter 7.[11] There are API kits for various languages, including PHP and Python (as well as Java, .NET, and JavaScript).[12]

10. http://www.google.com/support/calendar/bin/answer.py?answer=34576&hl=en

11. http://code.google.com/apis/calendar/overview.html

12. http://code.google.com/apis/calendar/developers_guide_protocol.html

Before I cover how to programmatically interact with the Google Calendar, I'll first cover what you can do by changing documents. It's useful to take a look at specific instances of iCalendar and the XML feeds.

iCalendar/iCal

iCalendar is a dominant standard for the exchange of calendar data. Based on the older vCalendar standard, iCalendar is sometimes referred to as iCal, which might be confused with the name of the Apple calendaring program of the same name. The iCalendar standard is supported in a wide range of products.

The official documentation for iCalendar is RFC 2445:

```
http://tools.ietf.org/html/rfc2445
```

Some other allied standards are built around RFC 2445, but they are beyond the scope of this book:

* iCalendar Transport-Independent Interoperability Protocol (iTIP) Scheduling Events, BusyTime, To-dos and Journal Entries (RFC 2446) lays out how calendar servers can exchange calendaring events.[13]
* iCalendar Message-Based Interoperability Protocol (iMIP) (RFC 2447) covers the exchange of calendaring data by e-mail.[14]

See the Wikipedia article on iCalendar for a list of the wide range of products that support iCalendar.[15] Calendaring standards are complex. I recommend a good overview of how standards relate.[16]

The structure of an iCalendar file is *not* based on XML like many of the data exchange formats covered in this book. There have been attempts to cast the iCalendar data model into XML (such as xCal[17]), but none has reached the level of wide adoption that iCalendar has.

iCalendar has many features, but there are a few basic things to know about it:

* iCalendar has a top-level object: `VCALENDAR`.
* There are subobjects, including `VEVENT`, `VTODO`, `VJOURNAL`, and `VFREEBUSY`.

I'll focus mostly on the `VEVENT` object here—though `VFREEBUSY` is generated in Google Calendar when one uses the "Share only my free/busy information (hide details)" mode.

This is a simple example of iCalendar data (with one `VEVENT`), quoted from RFC 2445:[18]

13. http://tools.ietf.org/html/rfc2446

14. http://tools.ietf.org/html/rfc2447

15. http://en.wikipedia.org/wiki/ICalendar

16. http://www.calconnect.org/calendaringstandards.shtml

17. http://en.wikipedia.org/wiki/XCal

18. http://tools.ietf.org/html/rfc2445#section-4.4

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
DTSTART:19970714T170000Z
DTEND:19970715T035959Z
SUMMARY:Bastille Day Party
END:VEVENT
END:VCALENDAR
```

To see a more complicated instance of an iCalendar document, you can use Google Calendar via this:

```
curl
"http://www.google.com/calendar/ical/9imfjk71chkcs66t1i436je0s0%40group.~CCC
calendar.google.com/public/basic.ics"
```

This gets the iCalendar rendition of my public Mashup Guide Demo Calendar, a version of which is as follows:

```
BEGIN:VCALENDAR
PRODID:-//Google Inc//Google Calendar 70.9054//EN
VERSION:2.0
CALSCALE:GREGORIAN
METHOD:PUBLISH
X-WR-CALNAME:Mashup Guide Demo Calendar
X-WR-TIMEZONE:America/Los_Angeles
X-WR-CALDESC:a Google Calendar to support mashupguide.net
BEGIN:VTIMEZONE
TZID:America/Los_Angeles
X-LIC-LOCATION:America/Los_Angeles
BEGIN:DAYLIGHT
TZOFFSETFROM:-0800
TZOFFSETTO:-0700
TZNAME:PDT
DTSTART:19700308T020000
RRULE:FREQ=YEARLY;BYMONTH=3;BYDAY=2SU
END:DAYLIGHT
BEGIN:STANDARD
TZOFFSETFROM:-0700
TZOFFSETTO:-0800
TZNAME:PST
DTSTART:19701101T020000
RRULE:FREQ=YEARLY;BYMONTH=11;BYDAY=1SU
END:STANDARD
END:VTIMEZONE
BEGIN:VEVENT
DTSTART;TZID=America/Los_Angeles:20070507T130000
DTEND;TZID=America/Los_Angeles:20070507T140000
DTSTAMP:20070510T155641Z
ORGANIZER;CN=Mashup Guide Demo
Calendar:MAILTO:9imfjk71chkcs66t1i436je0s0@~CCC
group.calendar.google.com
UID:vk021kggr20ba2jhc3vjg6p8ek@google.com
CLASS:PUBLIC
CREATED:20070510T021623Z
DESCRIPTION:
LAST-MODIFIED:20070510T021623Z
LOCATION:110 South Hall\, UC Berkeley
SEQUENCE:0
STATUS:CONFIRMED
SUMMARY:Mixing and Remixing Information Class Open House
TRANSP:OPAQUE
END:VEVENT
```

```
BEGIN:VEVENT
DTSTART;TZID=America/Los_Angeles:20070411T123000
DTEND;TZID=America/Los_Angeles:20070411T140000
DTSTAMP:20070510T155641Z
ORGANIZER;CN=Mashup Guide Demo
Calendar:MAILTO:9imfjk71chkcs66t1i436je0s0@~CCC
group.calendar.google.com
UID:d9btebsfd121lhqc4arhj9727s@google.com
CLASS:PUBLIC
CREATED:20070411T144226Z
DESCRIPTION:
LAST-MODIFIED:20070411T144226Z
LOCATION:
SEQUENCE:0
STATUS:CONFIRMED
SUMMARY:Day 22
TRANSP:OPAQUE
END:VEVENT
END:VCALENDAR
```

This chapter does not cover the ins and outs of the iCalendar format. I recommend the following ways to learn more about iCalendar:

* Read the "Guide to Internet Calendaring" (`http://www.ietf.org/rfc/rfc3283.txt`).
* There are many standards (`http://www.calconnect.org/calendaringstandards.shtml`), but keep especially RFC 2445 in mind.
* Know that since iCalendar is rich in features, these features are not evenly implemented among calendars, servers, or libraries that claim to work with iCalendar.
* The community is wrestling with a lot of subtleties. That's why you have organizations such as CalConnect making recommendations about handling recurring events and time zones (`http://calconnect.org/recommendations.shtml`).
* Interoperability among iCalendar implementations remains a challenge,[19] so don't be surprised if you run into problems using one system to interpret an iCalendar file produced by another system.
* Have some good programming libraries on hand to parse and create iCalendar (although it's hard to know for sure the quality of any given iCalendar library).
* Note that work is underway to update the standards: `http://www.ietf.org/html.charters/calsify-charter.html`.

In working with iCalendar, I've found the iCalendar Validator (`http://severinghaus.org/projects/icv/`), based on the `iCal4j` library (`http://ical4j.sourceforge.net/`), to be useful. You can use it to validate the iCalendar feed for the Mashup Guide Demo Calendar:

```
http://severinghaus.org/projects/icv/?url=http%3A%2F%2Fwww.google.com%2Fcalendar
%2Fi~CCC
```

```
cal%2F9imfjk71chkcs66t1i436je0s0%2540group.calendar.google.com%2Fpublic%2Fbasic.
ics
```

19. http://www.calconnect.org/ioptesting.shtml and http://www.calconnect.org/interop/

uc%20berkeley%20interop%20testing.pdf

Google Calendar Atom Data

Now compare Google Calendar data formatted as an Atom XML feed, which you can get using this:

```
curl
http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40group.~CCC
calendar.google.com/public/basic
```

This will return a feed that looks something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:openSearch="http://a9.com/-
/spec/~CCC
opensearchrss/1.0/"
      xmlns:gd="http://schemas.google.com/g/2005"
      xmlns:gCal="http://schemas.google.com/gCal/2005">

<id>http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40group.cale
ndar.~CCCgoogle.com/public/basic</id>
  <updated>2007-05-10T02:16:23.000Z</updated>
  <category scheme="http://schemas.google.com/g/2005#kind"
            term="http://schemas.google.com/g/2005#event"/>
  <title type="text">Mashup Guide Demo Calendar</title>
  <subtitle type="text">a Google Calendar to support mashupguide.net</subtitle>
  <link rel="http://schemas.google.com/g/2005#feed" type="application/atom+xml"
        href="http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40
group.calendar.google.com/public/basic"/>
  <link rel="self" type="application/atom+xml"
        href="http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40
group.calendar.google.com/public/basic?max-results=25"/>
  <author>
    <name>Raymond Yee</name>
    <email>raymond.yee@gmail.com</email>
  </author>
  <generator version="1.0" uri="http://www.google.com/calendar">Google Calendar
</generator>
  <openSearch:totalResults>2</openSearch:totalResults>
  <openSearch:startIndex>1</openSearch:startIndex>
  <openSearch:itemsPerPage>25</openSearch:itemsPerPage>
  <gd:where valueString=""/>
  <gCal:timezone value="America/Los_Angeles"/>
  <entry>

<id>http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40group.~C
CCcalendar.google.com/public/basic/vk021kggr20ba2jhc3vjg6p8ek</id>
    <published>2007-05-10T02:16:23.000Z</published>
    <updated>2007-05-10T02:16:23.000Z</updated>
    <category scheme="http://schemas.google.com/g/2005#kind"
              term="http://schemas.google.com/g/2005#event"/>
    <title type="text">Mixing and Remixing Information Class Open House</title>
    <summary type="html">When: Mon May 7, 2007 1pm to 2pm&amp;nbsp;
PDT&lt;br&gt;
&lt;br&gt;Where: 110 South Hall, UC Berkeley &lt;br&gt;Event Status:
confirmed</summary>
    <content type="text">When: Mon May 7, 2007 1pm to 2pm&amp;nbsp;
PDT&lt;br&gt;
&lt;br&gt;Where: 110 South Hall, UC Berkeley &lt;br&gt;Event Status:
confirmed</content>
    <link rel="alternate" type="text/html" ~CCC

href="http://www.google.com/calendar/event?eid=dmswMjFrZ2dyMjBiYTJqaGMzd~CCC
mpnNnA4ZWsgOWltZmprNzFjaGtjczY2dDFpNDM2amUwczBAZw" title="alternate"/>
    <link rel="self" type="application/atom+xml" ~CCC

href="http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40~CCC
group.calendar.google.com/public/basic/vk021kggr20ba2jhc3vjg6p8ek"/>
    <author>
      <name>Mashup Guide Demo Calendar</name>
    </author>
    <gCal:sendEventNotifications value="false"/>
  </entry>
  <entry>

<id>http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40group.cale
ndar.~CCC
google.com/public/basic/d9btebsfd121lhqc4arhj9727s</id>
    <published>2007-04-11T14:42:26.000Z</published>
    <updated>2007-04-11T14:42:26.000Z</updated>
    <category scheme="http://schemas.google.com/g/2005#kind"
              term="http://schemas.google.com/g/2005#event"/>
    <title type="text">Day 22</title>
    <summary type="html">When: Wed Apr 11, 2007 12:30pm to 2pm&amp;nbsp;
PDT&lt;br&gt;   &lt;br&gt;Event Status:     confirmed</summary>
    <content type="text">When: Wed Apr 11, 2007 12:30pm to 2pm&amp;nbsp;
PDT&lt;br&gt;   &lt;br&gt;Event Status:     confirmed</content>
    <link rel="alternate" type="text/html" ~CCC

href="http://www.google.com/calendar/event?eid=ZDlidGVic2ZkMTIxbGhxYzRhcmh~CC
C
qOTcyN3MgOWltZmprNzFjaGtjczY2dDFpNDM2amUwczBAZw" title="alternate"/>
    <link rel="self" type="application/atom+xml" ~CCC

href="http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40~CCC
group.calendar.google.com/public/basic/d9btebsfd121lhqc4arhj9727s"/>
    <author>
      <name>Mashup Guide Demo Calendar</name>
    </author>
    <gCal:sendEventNotifications value="false"/>
  </entry>
</feed>
```

Note the following about this data:

* The feed is expressed in Atom format (which you learned about in Chapter 4).
* It uses common GData extension elements,[20] OpenSearch, and Google Calendar extensions.[21]

Using the GData-Based Calendar API Directly

In this section, I will lead you through the basics of programming the Google Calendar API. Since I won't cover all the details of the API, I refer you to "Google Calendar Data API Developer's Guide: Protocol" documentation as an excellent place to start. You'll learn how to set up some calendars and access the right URLs for various feeds.[22]

As with most APIs, you can take two basic approaches: you can work directly with the protocol, which in this case is based on the GData protocol that underlies many Google APIs, including that for Blogger (see Chapter 7), or you can use a language-specific API kit. Here I'll show you both approaches. Although the latter approach is often more practical, I'll use this explication of the Calendar API as a chance to review GData (and the concepts of REST in general). To work with the specific language-specific libraries, consult the documentation here:

```
http://code.google.com/apis/gdata/clientlibs.html
```

Later, I'll give a quick rundown on how to use the PHP and Python API kits. You can get started with the documentation for the Calendar API here:

```
http://code.google.com/apis/calendar/developers_guide_protocol.html
```

The reference for the API is here:

```
http://code.google.com/apis/calendar/reference.html
```

The Google Calendar API is based on GData, which in turn is based on APP with Google-specific extensions. APP is a strictly REST protocol; remember, that means resources are represented as Atom feeds, and you use standard HTTP methods (`GET`, `POST`, `PUT`, and `DELETE`) to read, update, create, and delete elements. Here I'll show you some of the key feeds and how to use them. Before diving into doing so, I'll first show you how to obtain an authentication token, which you need in order to make full use of these feeds (that is, beyond issuing `GET` requests for public feeds).

20. http://code.google.com/apis/gdata/elements.html

21. http://code.google.com/apis/calendar/reference.html#Elements

22. http://code.google.com/apis/calendar/developers_guide_protocol.html

Obtaining an Authentication Token

One of the two authentication methods available to you is documented here:

```
http://code.google.com/apis/gdata/auth.html
```

I'll show you how to use the `ClientLogin` technique here. To make authorized access to the API, you will need an authentication token, which you can obtain by making an HTTP `POST` request (using the `application/x-www-form-urlencoded` content type) to here:

```
https://www.google.com/accounts/ClientLogin
```

with a body that contains the following parameters:

Email: Your Google e-mail (for example, raymond.yee@gmail.com)

Password: Your Google password

source: A string of the form *companyName-applicationName-versionID* to identify your program (for example, mashupguide.net-Chap15-v1)

service: The name of the Google service, which in this case is cl

Using the example parameters listed here, you can package the authentication request with the following curl invocation:

```
curl -v -X POST -d "Passwd={passwd}&source=mashupguide.net-Chap15-
v1&Email=~CCC
raymond.yee%40gmail.com&service=cl" https://www.google.com/accounts/ClientLogin
```

If this call succeeds, you will get in the body of the response an Auth token (of the form Auth=[AUTH-TOKEN]). Retain the Auth token for your next calls. You will embed the authentication token in your calls by including the following HTTP request header:

```
Authorization: GoogleLogin auth=[AUTH-TOKEN]
```

Tip In curl, you do so with the -H option: -H "Authorization: GoogleLogin auth=[AUTH-TOKEN]".

On occasion, you will need to handle HTTP 302 redirects from the API. That is, instead of fulfilling a request, the Google Calendar API sends you a response with a *redirect URL* appended with the new query parameter gsessionid. You then reissue your request to this new URL.

Tip For HTTP GET, use the -L option in curl to automatically handle a redirect.

Feeds Available from Google Calendar

There are three feed types: *calendar* (for managing calendars), *event* (for events contained by calendars), and *comment* (for representing comments attached to events). Each of the feeds is qualified by two parameters: visibility and projection. After I describe visibility and projection, I'll list the various feeds and show how you can access them via curl. For more details about the feeds, consult this page:

```
http://code.google.com/apis/calendar/reference.html#Feeds
```

visibility and projection

There are two parameters for "specifying" the representation of feeds: visibility and projection. The visibility parameter can be one of public, private, or private-[magicCookie]. Feeds that are public do not require authorization and are always read-only; public feeds are inaccessible if the user has turned off sharing for the calendar. Feeds that are private do require authentication to use and are potentially writable in addition to being readable (that is, read/write). Finally, feeds that have a visibility of `private-[magicCookie]` are read-only and enable private information to be read without authorization. (The `magicCookie` encapsulates authentication information.)

The `projection` values are listed here:

```
http://code.google.com/apis/calendar/reference.html#Projection
```

They include the following:

* `full` (potentially read/write).
* `free-busy` (always read-only). This feed shows minimal information about events but does include data about the duration of events (in other words, the `<gd:when>` element).
* `basic` (always read-only). The basic projection produces Atom feeds without any extension elements; the `<atom:summary>` and `<atom:content>` elements contain HTML descriptions with embedded data about the events.

Calendar Feeds

There are three types of calendar feeds—*meta-feed*, *allcalendars*, and *owncalendars*—which I'll cover in turn.

meta-feed

The private and read-only meta-feed contains an `<entry>` element for each calendar to which the user has access. This list includes both calendars that are owned by the user and ones to which the user is subscribed. You can access the feed at the following URL:

```
http://www.google.com/calendar/feeds/default
```

by using this:

```
curl -L -X GET -H "Authorization: GoogleLogin  auth=[AUTH-TOKEN]"  ~CCC
http://www.google.com/calendar/feeds/default
```

Let's look at an instance of an `<entry>`. Here is my own default calendar:

```
<entry>

<id>http://www.google.com/calendar/feeds/default/raymond.yee%40gmail.com</id>
    <published>2007-10-20T18:46:01.839Z</published>
    <updated>2007-10-19T23:18:04.000Z</updated>
    <title type="text">Raymond Yee</title>
    <link rel="alternate" type="application/atom+xml" ~CCC

href="http://www.google.com/calendar/feeds/raymond.yee%40gmail.com/~CCC
private/full"/>
    <link rel="http://schemas.google.com/acl/2007#accessControlList" ~CCC
          type="application/atom+xml"~CCC

href="http://www.google.com/calendar/feeds/raymond.yee%40gmail.com/acl/~CCC
```

```
full"/>
    <link rel="self" type="application/atom+xml"

href="http://www.google.com/calendar/feeds/default/raymond.yee%40gmail.com"/>
    <author>
      <name>Raymond Yee</name>
      <email>raymond.yee@gmail.com</email>
    </author>
    <gCal:timezone value="America/Los_Angeles"/>
    <gCal:hidden value="false"/>
    <gCal:color value="#2952A3"/>
    <gCal:selected value="true"/>
    <gCal:accesslevel value="owner"/>
  </entry>
```

Note the three link elements in the entry for the meta-feed:

* rel="alternate" whose href is as follows:

```
http://www.google.com/calendar/feeds/raymond.yee%40gmail.com/private/full
```

If you were to do an authenticated GET on this feed, you'd see that this is an event feed containing all the events for the default calendar.

Note how the URL of this feed maps to the following form:

```
http://www.google.com/calendar/feeds/{userID}/{privacy}/{projection}
```

Here the user ID is raymond.yee%40gmail.com, visibility is private, and projection is full.

* rel="http://schemas.google.com/acl/2007#accessControlList". The following feed gives you the access control list for the given calendar.

```
http://www.google.com/calendar/feeds/raymond.yee%40gmail.com/acl/full
```

For this calendar, there is a single entry (I'm the only person who has permissions associated with my default calendar):

```
<entry>
<id>http://www.google.com/calendar/feeds/raymond.yee%40gmail.com/acl/~CCC
full/user%3Araymond.yee%40gmail.com</id>
    <updated>2007-10-20T23:14:47.000Z</updated>
    <category scheme="http://schemas.google.com/g/2005#kind"
      term="http://schemas.google.com/acl/2007#accessRule"/>
    <title type="text">owner</title>
    <content type="text"/>
    <link rel="self" type="application/atom+xml" ~CCC

href="http://www.google.com/calendar/feeds/raymond.yee%40gmail.com/acl/~CCC
full/user%3Araymond.yee%40gmail.com"/>
    <link rel="edit" type="application/atom+xml" ~CCC

href="http://www.google.com/calendar/feeds/raymond.yee%40gmail.com/acl/~CCC
full/user%3Araymond.yee%40gmail.com"/>
```

```
    <author>
      <name>Raymond Yee</name>
      <email>raymond.yee@gmail.com</email>
    </author>
    <gAcl:scope type="user" value="raymond.yee@gmail.com"/>
    <gAcl:role value="http://schemas.google.com/gCal/2005#owner"/>
  </entry>
```

* rel="self"

`http://www.google.com/calendar/feeds/default/raymond.yee%40gmail.com`

This feed returns one entry for the default calendar—instead of all the calendars to which the user (`raymond.yee@gmail.com`) has access.

allcalendars

The allcalendars feed is a private, potentially read/write feed for controlling subscriptions and settings (such as the display color) for calendars. Inserting or deleting entries to the allcalendars feed is tantamount to subscribing or unsubscribing to existing calendars. You can update personalization settings for your calendars: the color, whether it is hidden, and whether it is selected. You can't create or delete calendars by manipulating the allcalendars feed; for those actions, you need to use the owncalendars feed.

The URL for the allcalendars feed is here:

```
http://www.google.com/calendar/feeds/default/allcalendars/full
```

which you can access with this:

```
curl -L -X GET -H "Authorization: GoogleLogin  auth=[AUTH-TOKEN]"  ~CCC
http://www.google.com/calendar/feeds/default/allcalendars/full
```

Note You might wonder about the difference between meta-feed and allcalendars since both of them list all the calendars to which a user has access. The allcalendars feed with a `projection` value of `full` is read/write, while the meta-feed is read-only. If you try to access the allcalendars feed with a `projection` value of `basic` (to get something akin to the meta-feed), you'll get an "unknown visibility found" error.

I'll now walk you through how to manipulate the allcalendars feed to add and delete a subscription to the Phases of the Moon calendar, one of Google's public calendars, which is available here:

```
http://www.google.com/calendar/embed?src=ht3jlfaac5lfd6263ulfh4tql8%40group.cale
ndar.~CCC
google.com
```

Note the user ID of the calendar:

```
ht3jlfaac5lfd6263ulfh4tql8%40group.calendar.google.com
```

To subscribe to the calendar, create a file (called `phases_moon_entry.xml`) with the minimal entry element needed to be the body of the post as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
  <atom:id>ht3jlfaac5lfd6263ulfh4tql8%40group.calendar.google.com</atom:id>
</atom:entry>
```

Next, issue an HTTP `POST` request:

```
curl -v  -X POST --data-binary "@phases_of_moon_entry.xml"  -H "Content-Type:
~CCC
application/atom+xml "  -H "Authorization: GoogleLogin  auth=[AUTH-TOKEN]"
~CCC
http://www.google.com/calendar/feeds/default/allcalendars/full
```

As mentioned earlier, there's a good chance you'll get a 302 HTTP response code to this call:

```
http://www.google.com/calendar/feeds/default/allcalendars/full?gsessionid=~CC
C
{gessionid}
```

For example:

```
http://www.google.com/calendar/feeds/default/allcalendars/full?gsessionid=~CC
C
GUWxgPh61GQ
```

If you do get a 302 HTTP response code, reissue the call to the new URL with this:

```
curl -v  -X POST --data-binary "@phases_of_moon_entry.xml" -H "Content-Type:
~CCC
application/atom+xml "  -H "Authorization: GoogleLogin  auth=[AUTH-TOKEN]"
~CCC
http://www.google.com/calendar/feeds/default/allcalendars/full?gsessionid=~CC
C
{gessionid}
```

If the request to subscribe to the Phases of the Moon calendar is successful, you'll get a 201 HTTP response code to indicate a created calendar, along with a response body akin to this:

```
  <entry>
<id>http://www.google.com/calendar/feeds/default/allcalendars/full/~CCC
ht3jlfaac5lfd6263ulfh4tql8%40group.calendar.google.com</id>
    <published>2007-10-20T23:55:52.611Z</published>
    <updated>2007-10-14T07:19:30.000Z</updated>
    <title type="text">Phases of the Moon</title>
    <summary type="text"/>
    <link rel="alternate" type="application/atom+xml" ~CCC

href="http://www.google.com/calendar/feeds/ht3jlfaac5lfd6263ulfh4tql8%40~CCC
group.calendar.google.com/private/full"/>
    <link rel="self" type="application/atom+xml"~CCC

href="http://www.google.com/calendar/feeds/default/allcalendars/full/~CCC
ht3jlfaac5lfd6263ulfh4tql8%40group.calendar.google.com"/>
    <link rel="edit" type="application/atom+xml" ~CCC
```

```
href="http://www.google.com/calendar/feeds/default/allcalendars/full/~CCC
ht3jlfaac5lfd6263ulfh4tql8%40group.calendar.google.com"/>
    <author>
      <name>Phases of the Moon</name>
    </author>
    <gCal:timezone value="Etc/GMT"/>
    <gCal:hidden value="false"/>
    <gCal:color value="#7A367A"/>
    <gCal:selected value="false"/>
    <gCal:accesslevel value="read"/>
    <gd:where valueString=""/>
  </entry>
```

You can then unsubscribe to the Phases of the Moon calendar with the following HTTP DELETE request:

```
curl -v -X DELETE -H "Authorization: GoogleLogin  auth=[AUTH-TOKEN]" ~CCC
http://www.google.com/calendar/feeds/default/allcalendars/full/ht3jlfaac5lfd6263
ulfh~CCC
4tql8%40group.calendar.google.com?gsessionid={gsessionid}
```

owncalendars

The owncalendars feeds hold data about the calendars that a user owns. This feed is conceptually similar to the allcalendars feed, with one important difference. Instead of subscribing and unsubscribing to calendars, actions on the owncalendars feed are equivalent to creating and destroying calendars. The syntax for manipulating the owncalendars feed is similar to that for the allcallendars feed. For instance, to retrieve the feed, do a GET to this:

```
http://www.google.com/calendar/feeds/default/owncalendars/full
```

For example:

```
curl -v -L -X GET -H "Authorization: GoogleLogin  auth=[AUTH-TOKEN]" ~CCC
http://www.google.com/calendar/feeds/default/owncalendars/full
```

To create a new book-writing calendar, create a file entitled book_writing_calendar_entry.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns='http://www.w3.org/2005/Atom'
  xmlns:gd='http://schemas.google.com/g/2005'
  xmlns:gCal='http://schemas.google.com/gCal/2005'>
  <title type='text'>Book Writing Schedule</title>
  <summary type='text'>A calendar to track when I write my book.</summary>
  <gCal:timezone value='America/Los_Angeles'></gCal:timezone>
  <gCal:hidden value='false'></gCal:hidden>
  <gCal:color value='#2952A3'></gCal:color>
  <gd:where rel='' label='' valueString='Berkeley, CA'></gd:where>
</entry>
```

and do the following POST (after handling the HTTP 302 redirect):

```
curl -v  -X POST --data-binary "@book_writing_calendar_entry.xml"  -H "Content-
Type: ~CCC
application/atom+xml "  -H "Authorization: GoogleLogin  auth=[AUTH-TOKEN]"
~CCC
http://www.google.com/calendar/feeds/default/owncalendars/full?gsessionid=~CC
C
{gsession-id}
```

Furthermore, you can then update an existing calendar by issuing the appropriate PUT:

```
http://www.google.com/calendar/feeds/default/owncalendars/full/{userID}
```

And you can delete an existing calendar by using DELETE:

```
http://www.google.com/calendar/feeds/default/owncalendars/full/{userID}
```

Event Feeds

Now that you have studied the three types of calendar feeds, you'll look at how to use the event feeds. (I won't cover comment feeds in this book.) Specifically, let's look at the simple case of retrieving all the events from a given feed for which you have write privileges. To work with a given calendar, you need to know its user ID. In the instance of my own calendars (the Mashup Guide Demo calendar), the user ID is as follows:

```
9imfjk71chkcs66t1i436je0s0%40group.calendar.google.com
```

The syntax of the URL to the feed of the events is as follows:

```
http://www.google.com/calendar/feeds/{userID}/{privacy}/{projection}
```

Specifically, you can use a privacy value of public and a projection value of full since the calendar is a public one to arrive here:

```
http://www.google.com/calendar/feeds/{userID}/public/full
```

For example:

```
http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40group.calendar
.~CCC
google.com/public/full
```

which you can confirm is a URL to a feed of all the events on the calendar. To add an event, you need to send an HTTP POST request (with the proper authentication) here:

```
http://www.google.com/calendar/feeds/{userID}/private/full
```

For example:

```
http://www.google.com/calendar/feeds/9imfjk71chkcs66t1i436je0s0%40group.calendar
.~CCC
google.com/private/full
```

That is, you create a file by the name of project_showcase_event.xml with the following content:

```
<?xml version='1.0' encoding='UTF-8'?>
<entry xmlns='http://www.w3.org/2005/Atom'
  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'></category>
  <title type='text'>Project Showcase</title>
  <content type='text'>A chance for the class to show off their
projects</content>
  <gd:where valueString='110 South Hall'></gd:where>
  <gd:when startTime="2008-05-12T13:00:00.000-07:00"
           endTime="2008-05-12T14:00:00.000-07:00"/>
</entry>
```

and issue the following request:

```
curl -v  -X POST  --data-binary "@project_showcase_event.xml"  -H "Content-Type:
~CCC
application/atom+xml "  -H "Authorization: GoogleLogin  auth=[AUTH-TOKEN]"
~CCC
http://www.google.com/calendar/feeds/{userID}/private/full?gsessionid={gsessioni
d}
```

where the `gsessionid` is the one given in the 302 redirect to create an event on the Mashup Guide Demo calendar.

With an analogous procedure to how you subscribe or unsubscribe to calendars in the allcalendars feed or create calendars through the owncalendars feed, you can create and delete events through the events feed.

Using the PHP API Kit for Google Calendar

Working directly with the GData interface to Google Calendar gives you a lot of flexibility at the cost of tedium. Now we'll turn to studying how to use two of the API wrappers for Google Calendar. In the next section, I'll show you how to use the Python API kit. Here, we'll study the PHP wrapper.

The PHP API kit is documented here:

```
http://code.google.com/apis/calendar/developers_guide_php.html
```

The PHP library for accessing Google Calendar is part of the Zend Google Data Client Library, which, in turn, is available as part of the Zend Framework or as a separate download. Note that the library is developed by Zend and works with PHP 5.1.4 or newer. You can download the Zend Framework from this location:

```
http://framework.zend.com/
```

You can read about how to use the Zend Framework to access Google Calendar here:

```
http://framework.zend.com/manual/en/zend.gdata.calendar.html
```

You install the Zend framework by copying the files to a directory of your choice. I set up the Zend Framework in this location:

```
http://examples.mashupguide.net/lib/ZendFramework/
```

I'll now illustrate the basics of using this library through two code snippets. Both use the `ClientLogin` form of authorization. The first example retrieves a list of a user's calendars:

```
<?php
```

```
require_once 'Zend/Loader.php';
Zend_Loader::loadClass('Zend_Gdata');
Zend_Loader::loadClass('Zend_Gdata_ClientLogin');
Zend_Loader::loadClass('Zend_Gdata_Calendar');

function getGDataClient($user, $pass)
{
  $service = Zend_Gdata_Calendar::AUTH_SERVICE_NAME;

  $client = Zend_Gdata_ClientLogin::getHttpClient($user, $pass, $service);
  return $client;
}

function printCalendarList($client)
{
  $gdataCal = new Zend_Gdata_Calendar($client);
  $calFeed = $gdataCal->getCalendarListFeed();
  echo $calFeed->title->text . "\n";
  echo "\n";
  foreach ($calFeed as $calendar) {
    echo $calendar->title->text, "\n";
  }
}

$USER = "[USER]";
$PASSWORD = "[PASSWORD]";

$client = getGDataClient($USER, $PASSWORD);
printCalendarList($client);

?>
```

The second code sample retrieves a list of events for a given calendar and prints basic elements for a given event: its ID, title, content, and details about the "where" and "when" of the event:

```
<?php

require_once 'Zend/Loader.php';
Zend_Loader::loadClass('Zend_Gdata');
Zend_Loader::loadClass('Zend_Gdata_ClientLogin');
Zend_Loader::loadClass('Zend_Gdata_Calendar');

function getGDataClient($user, $pass)
{
  $service = Zend_Gdata_Calendar::AUTH_SERVICE_NAME;

  $client = Zend_Gdata_ClientLogin::getHttpClient($user, $pass, $service);
  return $client;
}

function printEventsForCalendar($client, $userID)
{
  $gdataCal = new Zend_Gdata_Calendar($client);

  $query = $gdataCal->newEventQuery();
  $query->setUser($userID);
  $query->setVisibility('private');
  $query->setProjection('full');

  $eventFeed = $gdataCal->getCalendarEventFeed($query);

  echo $eventFeed->title->text . "\n";
  echo "\n";
  foreach ($eventFeed as $event) {
    echo $event->title->text, "\t", $event->id->text, "\n" ;
    echo $event->content->text, "\n";
    foreach ($event->where as $where) {
      echo $where, "\n";
    }
    foreach ($event->when as $when) {
      echo "Starts: " . $when->startTime . "\n";
      echo "Ends: " . $when->endTime . "\n";
    }

    # check for recurring events
    if ($recurrence = $event->getRecurrence()) {
      echo "recurrence: ", $recurrence, "\n";
    }

    print "\n";
  }
}

$USER = "[USER]";
$PASSWORD = "[PASSWORD]";

# userID for the Mashup Guide Demo calendar
$userID = "9imfjk71chkcs66t1i436je0s0%40group.calendar.google.com";

$client = getGDataClient($USER, $PASSWORD);
printEventsForCalendar($client, $userID);

?>
```

Later in the chapter, you will see how to use the PHP Google Calendar library to create events.

Using the Python API Kit for Google Calendar

You can find the documentation on the Python API kit here:

`http://code.google.com/apis/calendar/developers_guide_python.html`

To install the library, you can download it from here:

`http://code.google.com/p/gdata-python-client/downloads/list`

Or you can access the access the Subversion repository for the project here:

```
http://gdata-python-client.googlecode.com/svn/trunk/
```

Note The following code depends on the `ElementTree` library, which ships with Python 2.5 and newer. You can find instructions for downloading `ElementTree` at `http://effbot.org/zone/element-index.htm`.

Here's some Python code to demonstrate how to list all of your calendars and to list the events on a specific calendar:

```
"""
Chapter 15:  simple facade for Python Google Calendar library
"""
__author__ = 'raymond.yee@gmail.com (Raymond Yee)'

EMAIL = '[USER]'
PASSWORD = '[PASSWORD]'

try:
  from xml.etree import ElementTree
except ImportError:
  from elementtree import ElementTree

import gdata.calendar.service
import gdata.calendar
import atom

class MyGCal:
    def __init__(self):
        self.client = gdata.calendar.service.CalendarService()
        self.client.email = EMAIL
        self.client.password = PASSWORD
        self.client.source = 'GCalendarUtil-raymondyee.net-v1.0'
        self.client.ProgrammaticLogin()
    def listAllCalendars(self):
        feed = self.client.GetAllCalendarsFeed()
        print 'Printing allcalendars: %s' % feed.title.text
        for calendar in feed.entry:
          print calendar.title.text
    def listOwnCalendars(self):
        feed = self.client.GetOwnCalendarsFeed()
        print 'Printing owncalendars: %s' % feed.title.text
        for calendar in feed.entry:
          print calendar.title.text
    def listEventsOnCalendar(self,userID='default'):
      """
      list all events on the calendar with userID
      """
      query = gdata.calendar.service.CalendarEventQuery(userID, 'private',
'full')
```

```
        feed =  self.client.CalendarQuery(query)
        for event in feed.entry:
          print event.title.text, event.id.text, event.content.text
          for where in event.where:
            print where.value_string
          for when in event.when:
            print when.start_time, when.end_time
          if event.recurrence:
            print "recurrence:", event.recurrence.text

if __name__ == '__main__':
    gc = MyGCal()
    gc.listAllCalendars()
    # userID for Mashup Guide Demo calendar
    userID = '9imfjk71chkcs66t1i436je0s0%40group.calendar.google.com'
    gc.listEventsOnCalendar(userID)
```

30boxes.com

30boxes.com is another online calendar service, one that has won some rave reviews.[23] It has very noteworthy features, in addition to an API, making it worthwhile to describe it here.

23. http://30boxes.com/press

For information about the 30boxes.com API, go here:

* http://30boxes.com/developers
* http://30boxes.com/api/

An End User Tutorial

Before programming 30boxes.com, it's useful of course to view it as an end user:

1. Sign up for an account if you don't already have one:

`http://30boxes.com/signup`

2. Once you have an account, log into it:

`http://30boxes.com/login`

3. You can learn how to do various tasks at 30boxes.com by consulting the help section (http://30boxes.com/help).

One noteworthy feature from an end user's point of view is that, in terms of sharing, it seems that all calendars are completely private by default. You can add buddies and set options as to how much a given buddy can see:

* Buddies can see your entire calendar unless you mark an event as private.
* Buddies can see events that are marked with a certain tag.
* Buddies can see only the stuff on the buddy page.

30boxes.com API

The main documentation is at this location:

`http://30boxes.com/api/`

You have to get a key here:

`http://30boxes.com/api/api.php?method=getKeyForUser`

In this section, we'll exercise the API. Please substitute your own `[APIKEY]` and `[AUTHTOKEN]`. You can do HTTP `GET` requests on the following URLs:

* `test.ping`:[24]

```
http://30boxes.com/api/api.php?method=test.Ping&apiKey={APIKEY}
```

24. http://30boxes.com/api/#t

* `user.FindByEmail`:

```
http://30boxes.com/api/api.php?method=user.FindByEmail&apiKey={APIKEY}&email=yee
@~CCC
berkeley.edu
```

* `user.Authorize`: Many methods require authorization, which then yields an authorization token. In this example, I use a small picture of me as the application icon.[25] When calling `user.FindByEmail`, I also drop the optional `returnURL` argument:

25. http://farm1.static.flickr.com/4/5530475_48f80eece8_s.jpg

```
http://30boxes.com/api/api.php?method=user.Authorize&apiKey={APIKEY}~CCC&appl
icationName={application-name}&applicationLogoUrl={url}
```

For example:

```
http://30boxes.com/api/api.php?method=user.Authorize&apiKey={APIKEY}~CCC&appl
icationName=Raymond+Yee&applicationLogoUrl=http%3A%2F%2Ffarm1.static.~CCCflic
kr.com%2F4~CCC%2F5530475_48f80eece8_s.jpg
```

You will get an authentication token, which I show here as `{AUTHTOKEN}`.

* `user.GetAllInfo`:

```
http://30boxes.com/api/api.php?method=user.GetAllInfo&apiKey={APIKEY}~CCC&aut
horizedUserToken={AUTHTOKEN}
```

to which you will get something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<rsp stat="ok">
  <user>
    <id>40756</id>
    <facebookId>1229336</facebookId>
    <firstName>Raymond</firstName>
    <lastName>Yee</lastName>
    <avatar>http://farm1.static.flickr.com/4/5530475_48f80eece8_s.jpg</avatar>
    <status>sweeping stuff under the carpet while he writes.</status>
    <bio/>
    <dateFormat>MM-DD-YYYY</dateFormat>
    <timeZone>US/Pacific</timeZone>
    <createDate>2006-03-17</createDate>
    <startDay>0</startDay>
    <use24HourClock>0</use24HourClock>
    <feed>
      <name>Raymond - MySpace Blog</name>
      <url>http://blog.myspace.com/blog/rss.cfm?friendID=82943257</url>
    </feed>
    <email>
      <address>yee@berkeley.edu</address>
      <primary>1</primary>
    </email>
    <email>
      <address>raymond.yee@gmail.com</address>
      <primary>0</primary>
    </email>
    <otherContact>
      <type>Yahoo</type>
      <value>rdhyee</value>
    </otherContact>
    <otherContact>
      <type>Personal Site</type>
      <value>http://hypotyposis.net/blog</value>
    </otherContact>

  </user>
</rsp>
```

* events.Get:

```
http://30boxes.com/api/api.php?method=events.Get&apiKey={APIKEY}~CCC&authoriz
edUserToken={AUTHTOKEN}&start=2007-01-01&end=2007-09-01
```

to which you will get something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<rsp stat="ok">
  <eventList>
    <userId>40756</userId>
    <listStart>2007-01-01</listStart>
    <listEnd>2007-06-30</listEnd>
    <event>
      <id>1767437</id>
      <summary>Y!RB Brain Jam: A CHI2007 Sampler</summary>
      <notes>[....]</notes>
      <start>2007-04-27 14:00:00</start>
      <end>2007-04-27 14:00:00</end>
      <lastUpdate>2007-04-11 15:08:58</lastUpdate>
      <allDayEvent>0</allDayEvent>
      <repeatType>no</repeatType>
      <repeatEndDate>0000-00-00</repeatEndDate>
      <repeatSkipDates/>
      <repeatICal/>
      <reminder>-1</reminder>
      <tags/>
      <externalUID>http://upcoming.org/event/172254/</externalUID>
      <privacy>shared</privacy>
      <invitation>
        <isInvitation>0</isInvitation>
      </invitation>
    </event>
[....]
  </eventList>

</rsp>
```

Note The end parameter cannot be more than 180 days after start.

---

* events.GetDisplayList (to get an expanded and sorted list of events):

```
http://30boxes.com/api/api.php?method=events.GetDisplayList&apiKey={APIKEY}~CC
C&authorizedUserToken={AUTHTOKEN}&start=2007-01-01&end=2007-09-01
```

* todos.Get:

```
http://30boxes.com/api/api.php?method=todos.Get&apiKey={APIKEY}&authorizedUser~
CCCToken={AUTHTOKEN}
```

* todos.Add:

```
http://30boxes.com/api/api.php?method=todos.Add&apiKey={APIKEY}&authorizedUser~
CCCToken={AUTHTOKEN}&text=Eat+more+veggies&externalUID=123456x
```

* todos.Update:

```
http://30boxes.com/api/api.php?method=todos.Update&apiKey={APIKEY}&authorized~C
CC
UserToken={AUTHTOKEN}&text=Eat+more+veggies+and+fruit&todoId=123110&externalUID=
~CCC

123456x
```

* todos.Delete:

```
http://30boxes.com/api/api.php?method=todos.Delete&apiKey={APIKEY}&authorized~C
CCUserToken={AUTHTOKEN}&text=Eat+more+veggies+and+fruit&todoId=123110
```

* events.AddByOneBox:

```
http://30boxes.com/api/api.php?method=events.AddByOneBox&apiKey={APIKEY}~CCC&
authorizedUserToken={AUTHTOKEN}&event=eat+some+sushi+tomorrow+at+7pm
```

Note You can find a Python API wrapper for 30boxes.com at http://trentm.com/projects/thirtyboxes/.

Event Aggregators

Google Calendar and 30boxes.com are examples of online calendars meant to allow individuals and small groups of people to coordinate their appointments. Complementing such calendars are event aggregators that gather and list events, many of which are public events. In the following sections, I'll cover two event aggregators that are programmable and hence mashable: Upcoming.yahoo.com and Eventful.com.

Upcoming.yahoo.com

The URL for Upcoming.yahoo.com is as follows:

```
http://upcoming.yahoo.com/
```

The URL for a specific event is as follows:

```
http://upcoming.yahoo.com/event/{event-id}/
```

For example, the following is the URL for CHI2007:

```
http://upcoming.yahoo.com/event/76140/
```

Feeds from Search Results

Upcoming.yahoo.com makes much of its data available through RSS 2.0 feeds. Let's consider an example. To look for events with the keyword `Bach` in the San Francisco Bay Area, you can use the following search:

```
http://upcoming.yahoo.com/search/?type=Events&rt=1&q=bach&loc=Berkeley%2C+Califo
rnia~CCC
%2C+United+States
```

In general, the URL for searching events is as follows:

```
http://upcoming.yahoo.com/search/?type=Events&rt=1&q={q}&loc={location}&sort={so
rt}
```

where you can set `sort` to `w` (to sort by popularity), `r` (by relevance), and `p` (by recently added).

The previous search gives you HTML. You can also get feeds out of the search results as either RSS 2.0 or iCalendar. The RSS 2.0 feed includes Dublin Core data, uses the xCal extension (`http://en.wikipedia.org/wiki/XCal`) to encode calendaring information, and includes latitude and longitude data encoded with the Compact W3C Basic Geo encoding (see Chapter 13 for details on this encoding):

```
http://upcoming.yahoo.com/syndicate/v2/search_all/?q=bach&loc=Berkeley%2C+Califo
rnia~CCC
%2C+United+States&rt=1
```

Take a look at a specific instance of an event:

```
    <geo:lat>37.7774</geo:lat>
    <geo:long>-122.4198</geo:long>
[....]
    <dc:date>2007-03-18T17:59:58-07:00</dc:date>
    <xCal:summary>San Francisco Symphony: Bach and Handel</xCal:summary>
    <xCal:dtstart>2008-04-05T20:00:00Z</xCal:dtstart>
    <xCal:dtend></xCal:dtend>
    <xCal:location>http://upcoming.yahoo.com/venue/17246/</xCal:location>
    <xCal:x-calconnect-venue>
     <xCal:x-calconnect-venue-id>http://upcoming.yahoo.com/venue/17246/
</xCal:x-calconnect-venue-id>
     <xCal:adr>
      <xCal:x-calconnect-venue-name>Davies Symphony Hall</xCal:x-calconnect-
venue-~CCC
name>
      <xCal:x-calconnect-street>201 Van Ness Avenue</xCal:x-calconnect-street>
      <xCal:x-calconnect-city>San Francisco Bay Area</xCal:x-calconnect-city>
      <xCal:x-calconnect-region>California</xCal:x-calconnect-region>
      <xCal:x-calconnect-postalcode>94102</xCal:x-calconnect-postalcode>
      <xCal:x-calconnect-country>United States</xCal:x-calconnect-country>
     </xCal:adr>
     <xCal:url type='Venue Website'>http://upcoming.yahoo.com/venue/17246/
</xCal:url>
     <xCal:x-calconnect-tel></xCal:x-calconnect-tel>
    </xCal:x-calconnect-venue>
```

You can get an iCalendar version of the results, which you can subscribe to using an iCalendar-cognizant calendar (for example, Apple iCal, Google Calendar, or Microsoft Outlook 2007):

```
webcal://upcoming.yahoo.com/calendar/v2/search_all/?q=bach&loc=Berkeley%2C+~CC
C
California%2C+United+States&rt=1
```

Note the use of the `webcal` URI scheme (`http://en.wikipedia.org/wiki/Webcal`). The `webcal` scheme tells the recipient to subscribe to the feed—to track updates—rather than just doing a one-time import of the iCalendar feed. (Note that you can replace `webcal` with `http` to get the contents of the iCalendar feed.)

```
http://upcoming.yahoo.com/calendar/v2/search_all/?q=bach&loc=Berkeley%2C+Califor
nia%~CCC
2C+United+States&rt=1
```

What can you do with these feeds coming from Upcoming.yahoo.com? One example is to generate KML out of the RSS 2.0 feeds, which already contain geolocations for the events. In fact, you can use Yahoo! Pipes for this very task:

```
http://pipes.yahoo.com/pipes/pipe.info?_id=GlqEg8WA3BGZNw9ELO2fWQ
```

This pipe takes as input the parameters that can be used to generate an upcoming RSS 2.0 feed from Upcoming.yahoo.com (`q`, `loc`, and `sort`) and uses the Location Extractor operator to extract the geoRSS elements from the feed.

Note You can extend the pipe to encompass the other search options at Upcoming.yahoo.com, such as date ranges or categories.

You can run the pipe for `Bach` events close to Berkeley, California, sorted by relevance:

```
http://pipes.yahoo.com/pipes/pipe.info?q=Bach&loc=Berkeley%2C+CA&sort=r&_cmd=Run
+~CCC
Pipe&_id=GlqEg8WA3BGZNw9ELO2fWQ&_run=1
```

Note that running this pipe generates a Yahoo! map showing the events contained in the feed. In addition to the RSS 2.0 feed version here:

```
http://pipes.yahoo.com/pipes/pipe.run?_id=GlqEg8WA3BGZNw9ELO2fWQ&_render=rss&loc
=~CCC
Berkeley%2C+CA&q=Bach&sort=r
```

which isn't that interesting (since Upcoming.yahoo.com already generates an RSS 2.0 feed), you can get a KML version of this feed (by changing the `_render` parameter to `kml`):

```
http://pipes.yahoo.com/pipes/pipe.run?_id=GlqEg8WA3BGZNw9ELO2fWQ&_render=kml&loc
=~CCC
Berkeley%2C+CA&q=Bach&sort=r
```

From Chapter 13, you learned how to sort KML feeds on Google Maps:

```
http://maps.google.com/maps?q=http:%2F%2Fpipes.yahoo.com%2Fpipes%2Fpipe.run%3F_i
d%3~CCC
DGlqEg8WA3BGZNw9ELO2fWQ%26_render%3Dkml%26loc%3DBerkeley%252C%2BCA%26q%3DBach%26
~CCC
sort%3Dr&ie=UTF8
```

Read-Only Parts of the API

Let's now turn to the Upcoming.yahoo.com API. You can find the documentation for the API here:

```
http://upcoming.yahoo.com/services/api/
```

You can generate a key to use for the API here:

```
http://upcoming.yahoo.com/services/api/keygen.php
```

The upcoming API is structured to be similar (but not identical) in detail to the Flickr REST API. The authentication is simpler and less sophisticated, but you'll see the `method` parameter and `api_key` (similar naming). The base URL for the API is as follows:

```
http://upcoming.yahooapis.com/services/rest/
```

Like the Flickr API, you need a method (`event.search`), an `api_key`, and other parameters for the given method, which are documented here:

```
http://upcoming.yahoo.com/services/api/event.search.php
```

There is a wide range of options (such as date range and precise location, in addition to paging parameters such as `per_page` and `page`). In this case, we're using the `search_text` and `location` parameters to put together an HTTP `GET` request:

```
http://upcoming.yahooapis.com/services/rest/?api_key={api_key}&method=event.sear
ch&~CCC
```

```
search_text=bach&location=Berkeley%@C+California
```

to which you get back a series of event elements:

```
<event id="166104" name="San Francisco Symphony: Bach and Handel"
       description="Christophers makes music of three centuries ago sound
~CCC
contemporary and utterly vital. Here, he conducts Baroque blockbusters, music of
~CCC
dazzling color and invention."
       start_date="2008-04-05" end_date="" start_time="20:00:00" end_time=""
       personal="0"
       selfpromotion="0"
metro_id="2;1311;1403;1849;1934;2122;2289;2466;2638;2962"
       venue_id="17246"
       user_id="59509" category_id="1" date_posted="2007-03-18 10:59:58"
       watchlist_count="6"

url="http://www.sfsymphony.org/templates/event_info.asp?nodeid=250&amp;~CCC
eventid=1188"
       distance="10.91" distance_units="miles" latitude="37.7774"
       longitude="-122.4198"
       geocoding_precision="address" geocoding_ambiguous="0"
       venue_name="Davies Symphony Hall"
       venue_address="201 Van Ness Avenue" venue_city="San Francisco Bay Area"
       venue_state_name="California" venue_state_code="ca" venue_state_id="5"
       venue_country_name="United States" venue_country_code="us"
       venue_country_id="1"
       venue_zip="94102"/>
```

Note what you get back. In addition to the “what” and “when” of the event, there is also specific geocoding. You can make a map (for example, converting this KML and displaying it on a map), which I showed earlier in the case of using the RSS 2.0 feed.

What else can do you with the API without authentication?

* You can use `event.getInfo` to retrieve information about public events given its `event_id`. For example, you can use the WWW2008 Conference (`http://upcoming.yahoo.com/event/205875`) here:

```
http://upcoming.yahooapis.com/services/rest/?method=event.getInfo&api_key=~CC
C
{api-key}&event_id=205875
```

to get the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
  <event id="205875" name="WWW 2008 (17th International World Wide Web
~CCCConference)"
    tags="www,web,www2008,ydn"
    description="&quot;The World Wide Web Conference is a global event bringing
~CCCtogether key researchers, innovators, decision-makers, technologists,
~CCCbusinesses, and standards bodies working to shape the Web. Since its
inception ~CCCin 1994, the WWW conference has become the annual venue for
```

```
international ~CCCdiscussions and debate on the future evolution of the
Web.&quot;"
    start_date="2008-04-21" end_date="2008-04-25" start_time="" end_time=""
    personal="0"
    selfpromotion="0" metro_id="420" venue_id="33275" user_id="18772"
    category_id="5"
    url="http://www2008.org/" date_posted="2007-06-12" latitude=""
~CCClongitude=""
    geocoding_precision="" geocoding_ambiguous=""
    venue_name="Beijing International Conference Center"
    venue_address="No.8 Beichendong Road Chaoyang District"
~CCCvenue_city="Beijing"
    venue_state_name="Beijing" venue_state_code="bj" venue_state_id="171"
    venue_country_name="China"
    venue_country_code="cn" venue_country_id="44" venue_zip="" venue_url=""
    venue_phone="+86-10-64910248"/>
</rsp>
```

* You can use `metro.getForLatLon` to retrieve a venue for a given latitude and longitude. Let's use the latitude and longitude for a building on the UC Berkeley campus in Berkeley, California:

```
37.869111,-122.260634
```

to formulate the following request:

```
http://upcoming.yahooapis.com/services/rest/?method=metro.getForLatLon&~CCC
api_key={api-key}&latitude=37.869111&longitude=-122.260634
```

which returns this:

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
  <metro id="2" name="San Francisco" code="sf" state_id="5"
state_name="California"
         state_code="ca"
         country_id="1" country_name="United States" country_code="us"/>
</rsp>
```

Parts of the API That Require Authentication

You will need to supply a callback URL for token-based authorization if you need that. How do you authenticate? The documentation is here:

```
http://upcoming.yahoo.com/services/api/token_auth.php
```

Getting the Token

The documentation tells you how to set up a callback URL for web-based applications. I consider this a simpler case in which you don't set any callback URL and manually read off a token. That is, load up this in your browser, and read the frob:

```
http://upcoming.yahoo.com/services/auth/?api_key={api-key}
```

Then get a token with an `auth.getToken` call:

```
http://upcoming.yahooapis.com/services/rest/?method=auth.getToken&api_key={api-
key}&~CCC
frob={frob}
```

to which you will get the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
<token token="[TOKEN]" user_id="[USER_ID]" user_username="[USERNAME]"
       user_name="[FULLNAME]" />
</rsp>
```

Adding an Event with the API

Let's use the API to add an event with the `event.add` method, which is documented here:

`http://upcoming.yahoo.com/services/api/event.add.php`

To add an event, issue an HTTP `POST` request with the following parameters:

* `api_key` (required)
* `token` (required)
* `name` (required)
* `venue_id` (numeric, required)
* `category_id` (numeric, required)
* `start_date` (YYYY-MM-DD, required)
* `end_date` (YYYY-MM-DD, optional)
* `start_time` (HH:MM:SS, optional)
* `end_time` (HH:MM:SS, optional)
* `description` (optional)
* `url` (optional)
* `personal` (1=visible to friends only or 0=public, optional, defaults to 0)
* `selfpromotion` (1=self-promotion or 0=normal, optional, defaults to 0)

For an example, I added the JCDL 2008 conference to Upcoming.yahoo.com:

`http://www.jcdl2008.org/`

The best way is to practice using the user interface of Upcoming.yahoo.com to help you pick out the venue ID and category ID:

`http://upcoming.yahoo.com/event/add/`

The location (found at `http://www.jcdl2008.org/location.html`) is the Omni William Penn Hotel in Pittsburgh, Pennsylvania. When you type the name of the hotel and its city into Upcoming.yahoo.com, it locates a venue. But how do you get the ID? You can use the API method `venue.search` (`http://upcoming.yahoo.com/services/api/venue.search.php`):

```
http://upcoming.yahooapis.com/services/rest/?api_key={api_key}&method=venue.sear
ch&~CCCsearch_text=Omni+William+Penn+Hotel&location=Pittsburgh%@C+PA
```

to which you get the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
  <venue id="56189" name="Omni William Penn Hotel" address="530 William Penn
Place"
    city="Pittsburgh" state="Pennsylvania" zip="" country="United States"
    url="http://www.omnihotels.com/FindAHotel/PittsburghWilliamPenn.aspx"
    description=""
    user_id="120115" metro_id="77" private="0" distance="0.14"
    distance_units="miles"
    latitude="40.4406" longitude="-79.997" geocoding_precision="address"
    geocoding_ambiguous="0"
    state_code="pa" state_id="39" country_code="us" country_id="1"/>
</rsp>
```

The conclusion is that the venue ID is `56189`.

The next question is, how do you get the category ID? You can use the `category.getList` method (`http://upcoming.yahoo.com/services/api/category.getList.php`):

```
http://upcoming.yahooapis.com/services/rest/?api_key={api_key}&method=category.g
etList
```

to get:

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
<category id="1" name="Music" description="Concerts, nightlife, raves" />
<category id="2" name="Performing/Visual Arts" description="Theatre, dance,
opera, ~CCC
exhibitions" />
<category id="3" name="Media" description="Film, book readings" />
<category id="4" name="Social" description="Rallies, gatherings, user groups" />
<category id="5" name="Education" description="Lectures, workshops" />
<category id="6" name="Commercial" description="Conventions, expos, flea
markets" />
<category id="7" name="Festivals" description="Big events, often multiple days"
/>
<category id="8" name="Sports" description="Sporting events, recreation" />
<category id="10" name="Other" description="Who knows?" />
<category id="11" name="Comedy" description="Stand-up, improv, comic theatre" />
<category id="12" name="Politics" description="Rallies, fundraisers, meetings"
/>
<category id="13" name="Family" description="Family/kid-oriented music, shows,
theatre" />
</rsp>
```

For this event, let’s pick `Education` (category 5).

Finally, I grab the description from here:

```
http://www.jcdl2008.org/index.html
```

Since 2001, the Joint Conference on Digital Libraries has served as the major international forum focused on digital libraries and associated technical, practical, and social issues . . .

Four hundred attendees are expected for the five days of events including a day of cutting edge tutorials; 3 days of papers, panels, and keynotes; and a day of research workshops.

OK—let's piece together a `curl` invocation that will create a new event in Upcoming.yahoo.com. Here is a Python program to generate the `curl` command:

```
import urllib

# parameters for creating the upcoming event
method = 'event.add'
api_key = '[API-KEY]'
token = '[TOKEN]'
name = 'Joint Conference on Digital Libraries (JCDL) 2008'
venue_id = '56189'
category_id = '5'  #education
start_date = '2008-06-15'
end_date = '2008-06-20'
description = """
[DESCRIPTION]
"""
url = 'http://www.jcdl2008.org/'
params = {'api_key': api_key, 'method':method, 'token':token, 'name':name,
~CCC
'venue_id':venue_id, 'category_id': category_id, ~CCC
          'start_date':start_date, 'end_date':end_date, 'description':
description, ~CCC
          'url': url}

command = 'curl -v -X POST -d "%s" %s' % (urllib.urlencode(params),
"http://upcoming.~CCCyahooapis.com/services/rest/")
print command
```

The resulting `curl` command is as follows:

```
curl -v -X POST -d
"venue_id=56189&name=Joint+Conference+on+Digital+Libraries+~CCC
%28JCDL%29+2008&end_date=2008-06-
20&url=http%3A%2F%2Fwww.jcdl2008.org%2F&description~CCC{description}&start_da
te=2008-06-15&token=[TOKEN]&api_key=[API-
KEY]&method=event.add~CCC&category_id=5"
http://upcoming.yahooapis.com/services/rest/
```

Remember that `[TOKEN]` is the authentication token received from the `auth.getToken` call issued earlier. The resulting event in Upcoming.yahoo.com is as follows:

```
http://upcoming.yahoo.com/event/300826/
```

API Kits for Upcoming.yahoo.com

To find API kits for Upcoming.yahoo.com, you can start with the links here:

```
http://upcoming.yahoo.com/help/w/Language-specific_Libraries
```

Although there does not seem to be any publicly available PHP kits at this point, you can find one for Python here:

```
http://code.google.com/p/upcoming-python-api/
```

Since this project currently has no downloads, you get the source via Subversion:

```
svn checkout http://upcoming-python-api.googlecode.com/svn/trunk/ upcoming-
python-api
```

The following code searches for events with the `Bach` keyword that are within five miles of Berkeley, California:

```
UPCOMING_API_KEY = '[UPCOMING_API_KEY]'

#from upcoming_api import Upcoming
from upcoming_api import UpcomingCached
import string

#upcoming = Upcoming(UPCOMING_API_KEY)

upcoming = UpcomingCached(UPCOMING_API_KEY)
bach_events = upcoming.event.search(search_text='Bach', location="Berkeley, CA")
print "There are %s events." % (len(bach_events))
for event in bach_events:

    print "%s\t%s\t%s" % (event['id'], event['name'], event['description']),

    v = upcoming.venue.getInfo(venue_id=event['venue_id'])
    print "%s\t%s\t%s\t%s" % (v[0]['name'], v[0]['address'], v[0]['city'],
~CCC
v[0]['zip']),

    # metro_id are ;-delimited list.  Sometimes  the metro list is empty....
    try:
        m_ids = string.split(event['metro_id'],";")
    # deal with only the first metro on the list
        m = upcoming.metro.getInfo(metro_id=m_ids[0])
        print 'metro name: ', m[0]['name']
    except:
        print "no metro name"
```

Here is an additional line of Python to add an event to Upcoming.yahoo.com:

```
new_event = upcoming.event.add(token=token,name=name,venue_id=venue_id, \
category_id=category_id, start_date=start_date,end_date=end_date, \
                   description=description)
```

Caution   As of this writing, I had to update `upcoming_api.py` to make sure `UPCOMING_API` is set to http://upcoming.yahooapis.com/services/rest/.

Eventful.com

Eventful.com is another event aggregator that has an API. You can find the web site here:

```
http://eventful.com/
```

Its API is documented is here:

```
http://api.eventful.com/
```

The list of methods in the API is here:

```
http://api.eventful.com/docs/
```

To use the API, you need to request a key from here:

```
http://api.eventful.com/keys/
```

The base URL for RESTful calls is here:

```
http://api.evdb.com/rest/{path for methods}
```

For example:

```
http://api.evdb.com/rest/events/search
```

Searching for Events (Using Feeds)

Before we jump into the API, let's see how to look at the URL language to search for events in the user interface and to return feeds. You can search for `Bach` events within five miles of Berkeley, California, with this:

```
http://eventful.com/events?page_size=50&sort_order=Date&within=5&units=mi&q=bach
&l=~CCC
berkeley%2C+ca&t=Future&c=
```

You can get these results as an RSS 2.0 feed:

```
http://eventful.com/rss/events/?page_size=50&sort_order=Date&within=5&units=mi&q
=~CCC
bach&l=berkeley%2C+ca&t=Future&c=
```

or as an Atom 1.0 feed:

```
http://eventful.com/atom/events/?page_size=50&sort_order=Date&within=5&units=mi&
q=~CCC
bach&l=berkeley%2C+ca&t=Future&c=
```

You can change this Atom feed into KML using Yahoo! Pipes the way we did so for Upcoming.yahoo.com. The Eventful.com feeds have latitude/longitude information embedded (specifically, in the GeoRSS GML encoding). For example:

```
<georss:where>
  <gml:Point>
    <gml:pos>37.72084 -122.476619</gml:pos>
  </gml:Point>
</georss:where>
```

You can run Yahoo! Pipes here:

```
http://pipes.yahoo.com/pipes/pipe.info?_id=lJPPcvWA3BGvrWbY6kjTQA
```

to generate a KML feed for Bach-related events in the Berkeley area:

```
http://pipes.yahoo.com/pipes/pipe.run?_id=lJPPcvWA3BGvrWbY6kjTQA&_render=kml&l=
~CCC
Berkeley%2C+CA&page_size=50&q=Bach&t=Future&units=mi&within=5
```

Searching for Events (Using the API)

Let's first get an XML response from the `/events/search` method, which is documented here:

```
http://api.eventful.com/docs/events/search
http://api.eventful.com/rest/events/search?app_key={api-
key}&keywords=Bach&location=~CCC
Berkeley%2C%20CA&within=5&units=5&page_size=50
```

to which you get event elements like this:

```
<event id="E0-001-005962514-3">
  <title>SF State Recital by Roger Woodward, piano faculty</title>
  <description> &lt;b&gt;Details:&lt;/b&gt;&lt;br&gt;Program: J.S. Bach: Well-
~CCCTempered Clavier, Book I </description>
  <start_time>2007-10-23 20:00:00</start_time>
  <stop_time/>
  <tz_id/>
  <tz_olson_path/>
  <tz_country/>
  <tz_city/>
  <venue_id>V0-001-000550476-8</venue_id>
  <venue_name>San Francisco State University</venue_name>
  <venue_display>1</venue_display>
  <venue_address>1600 Holloway Avenue</venue_address>
  <city_name>San Francisco</city_name>
  <region_name>California</region_name>
  <region_abbr>CA</region_abbr>
  <postal_code>94132</postal_code>
  <country_name>United States</country_name>
  <country_abbr2>US</country_abbr2>
  <country_abbr>USA</country_abbr>
  <latitude>37.72084</latitude>
  <longitude>-122.476619</longitude>
  <geocode_type>EVDB Geocoder</geocode_type>
  <all_day>0</all_day>
  <recur_string/>
  <trackback_count>0</trackback_count>
  <calendar_count>0</calendar_count>
  <comment_count>0</comment_count>
  <link_count>1</link_count>
  <going_count>0</going_count>
  <watching_count>0</watching_count>
  <created>2007-09-02 00:19:50</created>
  <owner>evdb</owner>
  <modified>2007-09-02 04:07:16</modified>
  <performers/>
  <image/>
  <privacy>1</privacy>
  <calendars/>
  <groups/>
  <going/>
</event>
```

Interestingly enough, we can also get iCalendar and RSS directly from the API. To get iCalendar, you use the `/events/ical` method documented here:

`http://api.eventful.com/docs/events/ical`

To get the `Bach` keyword–related events within five miles of Berkeley as an iCalendar feed, use this:

```
http://api.eventful.com/rest/events/ical?app_key={api-
key}&keywords=Bach&location=~CCC
Berkeley%2C%20CA&within=5&units=5&page_size=50
```

You can also change `http` to `webcal` and feed it to Google Calendar.

PHP API Kit for Eventful.com

You can find a list of API kits for Eventful.com here:

`http://api.eventful.com/`

For PHP, there are two choices. One is `Services_Eventful`, which we won't cover here, and the other is `Services_EVDB` (which seems to be compatible with PHP 4 and 5). You can find the code here:

`http://api.eventful.com/libs/Services_EVDB`

Let's say you want to extract this:

```
http://eventful.com/events/categories/technology?l=Berkeley%2C%20California%2C%2
0USA
```

The corresponding REST call is as follows:

```
http://api.evdb.com/rest/events/search?category=technology&location=Berkeley%2C%
20~CCC
California%2C%20USA&within=25&page_size=5&app_key={api-key}
```

Note that the default is a 25-mile radius of the location. This shows how you can do this with the `Services_EVDB` PHP API kit:

```
<?php
// http://api.eventful.com/libs/Services_EVDB

ini_set(
  'include_path',
    ini_get( 'include_path' ) . PATH_SEPARATOR . "/home/rdhyee/pear/lib/php" .
PATH_SEPARATOR . '/usr/local/lib/php'
    );

require 'Services/EVDB.php';

// Enter your application key here. (See http://api.evdb.com/keys/)
$app_key = '[APP_KEY]';

$evdb = &new Services_EVDB($app_key);

// Authentication is required for some API methods.
$user     = $_REQUEST['user'];
$password = $_REQUEST['password'];

if ($user and $password)
{
  $l = $evdb->login($user, $password);

  if ( PEAR::isError($l) )
  {
      print("Can't log in: " . $l->getMessage() . "\n");
  }
}

// All method calls other than login() go through call().
$args = array(
  'id' => $_REQUEST['id'],
);
$event = $evdb->call('events/get', $args);

if ( PEAR::isError($event) )
{
    print("An error occurred: " . $event->getMessage() . "\n");
    print_r( $evdb );
}

// The return value from a call is an XML_Unserializer data structure.
print_r( $event );
?>
```

To see this code in action on Eventful.com, the event number is `E0-001-004433237-3`:[26]

```
http://examples.mashupguide.net/ch15/evdb1.php?id=E0-001-004433237-3
```

Python API Kit for Eventful.com

You can find the documentation for `eventfulpy` here:

```
http://api.eventful.com/libs/python/
```

As of writing, the latest version is as follows:

```
http://api.eventful.com/libs/python/eventfulpy-0.3.tar.gz
```

See “Installing simplejson and httplib2 on Windows Python” in case you run into problems installing the dependencies for eventfulpy.

INSTALLING SIMPLEJSON AND HTTPLIB2 ON WINDOWS PYTHON

eventfulpy depends on two other libraries: simplejson (http://undefined.org/python/#simple_json) and httplib2 (http://bitworking.org/projects/httplib2/). When I installed simplesjon for Python 2.5 for Windows, I needed to do the following (I'm using the default directory for Python 2.5 on Windows: C:\Python25):

1. Install setuptools (http://pypi.python.org/pypi/setuptools). The easiest way is to run the .exe installer (for example, setuptools-0.6c7.win32-py2.5.exe).

2. Use Subversion svn to check out simplejson from http://svn.red-bean.com/bob/simplejson/trunk/.

3. I installed mingw32 (http://www.mingw.org/) because I didn't have Visual Studio installed.

4. Build the simplejson library with the following command:

```
c:\python25\python.exe setup.py build -c mingw32 --force
```

5. Install the library by copying the resulting build\lib.win32-2.5\simplejson to C:\Python25\Lib\site-packages (I manually copied the directory because I could not find a way to coax the standard installation command (c:\python25\python.exe setup.py install) into working.

   You can find an alternative approach here:

http://maurus.net/weblog/2007/10/02/simplejson-17x-activestate-python-~CCCand-the-visual-studio-2003-compiler/

I found installing httplib2 to be more straightforward. For instance, you can download the latest distribution from http://code.google.com/p/httplib2/downloads/list and run c:\python25\python.exe setup.py install.

26. http://eventful.com/events/E0-001-004433237-3

The following code shows how to query for events and list the results:

```
import eventful

api = eventful.API('[API-KEY]')

# If you need to log in:
api.login('[USER]','[PASSWORD]')

events = api.call('/events/search', q='Bach', l='Berkeley, CA', within='5', \
units='mi', time='future', page_size=50)
for event in events['events']['event']:
    print "%s at %s" % (event['title'], event['venue_name'])
```

Let's now write JCDL 2008 to Eventful.com. Note that like Upcoming.yahoo.com, Eventful.com also uses IDs for venues. The following code has a `venue_search` method to help locate venues and their corresponding IDs:

```
# parameters for creating the upcoming event -- now I want to write it to
eventful

name = 'Joint Conference on Digital Libraries (JCDL) 2008'
start_date = '2008-06-15'
end_date = '2008-06-20'
description = """
[DESCRIPTION]
"""
url = 'http://www.jcdl2008.org/'

def venue_search(keywords,location):
    """
    print out possibilities...
    """
    import eventful

    api = eventful.API('[API-KEY]')
    api.login('[USER]','[PASSWORD]')
    vs = api.call('/venues/search', keywords = keywords, location=location)
    for v in vs['venues']['venue']:
        print "%s\t%s\t%s" % (v['id'], v['name'], v['address'])

import eventful

api = eventful.API('[API-KEY]')
api.login('[USER]','[PASSWORD]')

#http://api.eventful.com/docs/events/new
tz_olsen_path = 'America/New_York'
all_day = '1'
privacy = 1
tags = ''
free = 0

# this is the eventful venue_id for the hotel.
eventful_venue_id = 'V0-001-000412401-5'

ev = api.call('/events/new', title=name, start_time=start_date, \
              stop_time=end_date, tz_olsen_path=tz_olsen_path, all_day=all_day,
\
              description=description, privacy=privacy,
venue_id=eventful_venue_id)

import pprint
pprint(ev)
```

With success, you get back an ID for the event (`http://eventful.com/events/E0-001-006801918-6`):

```
{u'id': u'E0-001-006801918-6',
 u'message': u'Add event complete',
 u'status': u'ok'}
```

Programming with iCalendar

Since iCalendar is an important data format, it's worth looking a bit more at how to manipulate it in PHP and Python.

Note The hCalendar microformat is designed to express the same information as iCalendar but in a form that is embeddable in HTML and RSS. See Chapter 18 on microformats for how to use and create hCalendar.

Python and iCalendar

A good Python module to use is iCalendar:

```
http://codespeak.net/icalendar/
```

As of this writing, the latest version is 1.2. You download this code here:

```
http://codespeak.net/icalendar/iCalendar-1.2.tgz
```

To run a basic test of iCalendar interoperability, I created an event on Apple iCal and e-mailed it to myself. On my notebook, the filename is as follows:

```
D:\Document\Docs\2007\05\iCal-20070508-082112.ics
```

What's actually in the file?

```
BEGIN:VCALENDAR
VERSION:2.0
X-WR-CALNAME:open house at the Academy
PRODID:-//Apple Computer\, Inc//iCal 2.0//EN
CALSCALE:GREGORIAN
METHOD:PUBLISH
BEGIN:VTIMEZONE
TZID:US/Pacific
LAST-MODIFIED:20070508T152112Z
BEGIN:DAYLIGHT
DTSTART:20070311T100000
TZOFFSETTO:-0700
TZOFFSETFROM:+0000
TZNAME:PDT
END:DAYLIGHT
BEGIN:STANDARD
DTSTART:20071104T020000
TZOFFSETTO:-0800
TZOFFSETFROM:-0700
TZNAME:PST
END:STANDARD
END:VTIMEZONE
```

```
BEGIN:VEVENT
DTSTART;TZID=US/Pacific:20070510T190000
DTEND;TZID=US/Pacific:20070510T200000
SUMMARY:open house at the Academy
UID:AAE603F6-A5A1-4E11-91CF-E6B06649A756
ORGANIZER;CN="Raymond Yee":mailto:rdhyee@yahoo.com
SEQUENCE:6
DTSTAMP:20070508T152047Z
END:VEVENT
END:VCALENDAR
```

Now, I want to read it in using Python. Let's also consult the documentation to build a simple example:[27]

27. http://codespeak.net/icalendar/, http://codespeak.net/icalendar/example.html, http://codespeak.net/icalendar/small.html, and http://codespeak.net/icalendar/groupscheduled.html

```
from icalendar import Calendar
fname = r'D:\Document\Docs\2007\05\iCal-20070508-082112.ics'
cal = Calendar.from_string(open(fname,'rb').read())
ev0 = cal.walk('vevent')[0]
print ev0.keys()
print "summary: ", str(ev0['SUMMARY'])
print "start:", str(ev0['DTSTART'])
# ev0['DTSTART'] is datetime.date() object
print "end:", str(ev0['DTEND'])
```

If you run it, you get this:

```
['DTSTAMP', 'UID', 'SEQUENCE', 'SUMMARY', 'DTEND', 'DTSTART', 'ORGANIZER']
summary:  open house at the Academy
start: 20070510T190000
end: 20070510T200000
```

Another Python iCalendar library is `vobject`:

http://vobject.skyhouseconsulting.com/usage.html

The following code shows how to use `vobject` to parse the same iCalendar file:

```
import vobject
fname = r'D:\Document\Docs\2007\05\iCal-20070508-082112.ics'
cal = vobject.readOne(open(fname,'rb').read())
event = cal.vevent
print event.sortChildKeys()
print "summary: ", event.getChildValue('summary')
print "start:", str(event.getChildValue('dtstart'))
# event.getChildValue('dtstart') is datetime.date() object
print "end:", str(event.getChildValue('dtend'))
```

PHP and iCalendar

You can download iCalcreator, a PHP library for parsing and creating iCalendar files, here:

http://www.kigkonsult.se/iCalcreator/index.php

The module is documented here:

```
http://www.kigkonsult.se/iCalcreator/docs/using.html
```

Here is some code using iCalcreator to read and parse the same iCalendar file from the previous section:

```
<?php

require_once 'iCalcreator/iCalcreator.class.php';

  $filename = 'D:\Document\Docs\2007\05\iCal-20070508-082112.ics';

  $v = new vcalendar(); // initiate new CALENDAR
  $v->parse($filename);

  # get first vevent
  $comp = $v->getComponent("VEVENT");

  #print_r($comp);
  $summary_array = $comp->getProperty("summary", 1, TRUE);
  echo "summary: ", $summary_array["value"], "\n";

  $dtstart_array = $comp->getProperty("dtstart", 1, TRUE);
  $dtstart = $dtstart_array["value"];
  $startDate = "{$dtstart["year"]}-{$dtstart["month"]}-{$dtstart["day"]}";
  $startTime = "{$dtstart["hour"]}:{$dtstart["min"]}:{$dtstart["sec"]}";

  $dtend_array = $comp->getProperty("dtend", 1, TRUE);
  $dtend = $dtend_array["value"];
  $endDate = "{$dtend["year"]}-{$dtend["month"]}-{$dtend["day"]}";
  $endTime = "{$dtend["hour"]}:{$dtend["min"]}:{$dtend["sec"]}";

  echo "start: ",  $startDate,"T",$startTime, "\n";
  echo "end: ",  $endDate,"T",$endTime, "\n";

?>
```

The output of the code is as follows:

```
summary: open house at the Academy
start: 2007-05-10T19:00:00
end: 2007-05-10T20:00:00
```

I will use iCalcreator in the following section to convert iCalendar feeds into Google calendar entries.

Exporting an Events Calendar to iCalendar and Google Calendar

In this section, I'll show you how to use what you've learned so far to solve a specific problem. After you have used event aggregators such as Upcoming.yahoo.com and Eventful.com, you'll get used to the idea of having a single (or at least a small number) of places to see all your events. iCalendar-savvy calendars (such as Google Calendar, Apple iCal, and Microsoft Outlook 2007) have also become unifying interfaces by letting you subscribe to iCalendar feeds containing events that might be of interest to you. As extensive as Upcoming.yahoo.com, Eventful.com, and Google Calendar (which has been a marketplace of events by letting users author publicly available calendars) might be, there are still many sources of events that are not covered by such services. This section teaches you how to turn event-related information toward destinations where you might like to see them.

Specifically, I will work through the following example: converting events listed under the Critic's Choice section of UC Berkeley's online event calendar (`http://events.berkeley.edu`) into two different formats:

* An iCalendar feed
* A Google calendar

I use this example to demonstrate how to use Python and PHP libraries to parse and write iCalendar feeds and to write to a Google calendar. I've chosen the UC Berkeley event calendar because it already has calendaring information in a structured form (XML and iCalendar), but as of the time of writing, it's not quite in the configuration that I create here. You can generalize this example to the event calendars that you might be interested in, some with more structured information than others. Moreover, instead of writing to Google Calendar, you can use the techniques I showed earlier in the chapter to write the events to Upcoming.yahoo.com or Eventful.com.

The Source: UC Berkeley Event Calendars

The Critic's Choice section of the UC Berkeley event calendar highlights some of the many events that happen on the campus:

```
http://events.berkeley.edu/
```

As documented here:

```
http://events.berkeley.edu/documentation/user/rss.html
```

the calendar provides feeds in three formats: RSS 2.0, a `live_export` XML format, and iCalendar. Of particular interest is that every event in the calendar, which is referenced by an event ID (for example, `3950`), is accessible in a number of representations:

* As HTML:

```
http://events.berkeley.edu/?event_ID={event_ID}
```

* As RSS 2.0:

```
http://events.berkeley.edu/index.php/rss/sn/pubaff/?event_ID={event_ID}
```

* As iCalendar:

```
http://events.berkeley.edu/index.php/ical/event_ID/{event_ID}/.ics
```

* As `live_export` XML:

```
http://events.berkeley.edu/index.php/live_export/sn/pubaff/?event_ID={event_ID}
```

You can get feeds for many parts of the event calendar (including feeds for events for today, this week, or this month), but there is currently no Critic's Choice iCalendar feed. Having such a feed would enable one to track Critic's Choice events in Google Calendar or Apple iCal. The Critic's Choice is, however, available as an RSS 2.0 feed here:

```
http://events.berkeley.edu/index.php/critics_choice_rss.html
```

The following two sections show you how to extract the event ID for each of the events listed as part of Critic's Choice, read the iCalendar instance for an event to create a synthesized iCalendar feed, and write those events to Google Calendar.

Creating an iCalendar Feed of Critic's Choice Using Python

The following code, written in Python, knits together the iCalendar entries for each of the Critic's Choice events into a single iCalendar feed through the following steps:

1. Parsing the list event_ID from here:

```
http://events.berkeley.edu/index.php/critics_choice_rss.html
```

2. Reading the individual iCalendar entries and adding it to the one for the Critic's Choice

Note that this code treats iCalendar essentially as a black box. In the next section, we'll parse data from iCalendar and rewrite it in a format demanded of Google Calendar:

```
"""
generate iCalendar feed out of the UC Berkeley events calendar
"""

import sys
try:
    from xml.etree import ElementTree
except:
    from elementtree import ElementTree

import httplib2
client = httplib2.Http(".cache")

import vobject

# a function to get individual iCalendar feeds for each event.
# http://events.berkeley.edu/index.php/ical/event_ID/3950/.ics

def retrieve_ical(event_id):
    ical_url = "http://events.berkeley.edu/index.php/ical/event_ID/%s/.ics" %
(event_id)
    response, body = client.request(ical_url)
    return body

# read the RSS 2.0 feed for the Critic's Choice
```

```
from elementtree import ElementTree

cc_RSS = "http://events.berkeley.edu/index.php/critics_choice_rss.html"
response, xml = client.request(cc_RSS)
doc = ElementTree.fromstring(xml)

from pprint import pprint
import urlparse

# create a blank iCalendar
ical = vobject.iCalendar()

for item in doc.findall('.//item'):
    # extract the anchor to get the elementID
    # http://events.berkeley.edu/index.php/critics_choice.html#2875
    ev_url = item.find('link').text
    # grab the anchor of the URL, which is the event_ID
    event_id = urlparse.urlparse(ev_url)[5]
    print event_id
    s = retrieve_ical(event_id)
    try:
        ev0 = vobject.readOne(s).vevent
        ical.add(ev0)
    except:
        print "problem in generating iCalendar for event # %s " % (event_id)


ical_fname =
r'D:\Document\PersonalInfoRemixBook\examples\ch15\critics_choice.ics'
f = open(ical_fname, "wb")
f.write(ical.serialize())
f.close()

# upload my feed to the server
# http://examples.mashupguide.net/ch15/critics_choice.ics

import os
os.popen('scp2 critics_choice.ics ~CCC
"rdhyee@pepsi.dreamhost.com:/home/rdhyee/examples.mashupguide.net/ch15')
```

By automatically running this script every day, whenever the RSS for the Critic's Choice is regenerated, the resulting iCalendar feed will be kept up-to-date:

`http://examples.mashupguide.net/ch15/critics_choice.ics`

Writing the Events to Google Calendar

In this section, instead of generating an iCalendar feed directly, I will instead write the events to Google Calendar using the PHP Zend Calendar API library. I created a new calendar for this purpose, whose user ID is as follows:

`n7irauk3nns30fuku1anh43j5s@group.calendar.google.com`

Hence, the public calendar is viewable here:

```
http://www.google.com/calendar/embed?src=n7irauk3nns30fuku1anh43j5s@group.calend
ar.~CCC
google.com
```

The following code loops through the events listed in the Critic's Choice RSS feed, extracts all the corresponding iCalendar entries, and then writes those events to the Google Calendar. The code first clears out the old events in the calendar before writing new events.

Perhaps the trickiest part of this code is handling recurring events. The relevant documentation in the Google Calendar API on recurring events includes the following:

* http://code.google.com/apis/calendar/developers_guide_php.html#Creating~CCCRecurring
* http://code.google.com/apis/gdata/elements.html#gdRecurrence

The Google Calendar API expresses recurrence using the syntax and data model of recurring events in iCalendar, which you can learn about in the following sections of the iCalendar specification (section 4.3.10 on RECUR, section 4.8.5.1 on EXDATE [exception dates/times], and section 4.8.5.4 on the Recurrence Rule):

* http://www.w3.org/2002/12/cal/rfc2445#sec4.3.10
* http://www.w3.org/2002/12/cal/rfc2445#sec4.8.5.1
* http://www.w3.org/2002/12/cal/rfc2445#sec4.8.5.4

More to the point, the following code captures information about recurring events by using regular expressions to extract occurrences of the `DTSTART`, `DTEND`, `RRULE`, `RDATE`, `EXDATE`, and `EXRULE` statements to pass to the Google Calendar API as recurrence data. (Remember to substitute your own Google username and password and the user ID for a Google Calendar for which you have write permission.)

```php
<?php

/*
 *
 * ucb_critics_gcal.php
 */

require_once 'Zend/Loader.php';
Zend_Loader::loadClass('Zend_Gdata');
Zend_Loader::loadClass('Zend_Gdata_ClientLogin');
Zend_Loader::loadClass('Zend_Gdata_Calendar');

require_once 'iCalcreator/iCalcreator.class.php';

function getResource($url){
  $chandle = curl_init();
  curl_setopt($chandle, CURLOPT_URL, $url);
  curl_setopt($chandle, CURLOPT_RETURNTRANSFER, 1);
  $result = curl_exec($chandle);
  curl_close($chandle);
  return $result;
}

// UCB events calendar

# gets all relevant rules for the first VEVENT in $ical_string
function extract_recurrence($ical_string) {

  $vevent_rawstr = "/(?ims)BEGIN:VEVENT(.*)END:VEVENT/";
  preg_match($vevent_rawstr, $ical_string, $matches);

  $vevent_str = $matches[1];

  # now look for DTSTART, DTEND, RRULE, RDATE, EXDATE, and EXRULE

  $rep_tags = array('DTSTART', 'DTEND', 'RRULE', 'RDATE', 'EXDATE', 'EXRULE');

  $recur_list = array();

  foreach ($rep_tags as $rep) {

    $rep_regexp = "/({$rep}(.*))/i";
    if (preg_match_all($rep_regexp, $vevent_str, $rmatches)) {
      foreach ($rmatches[0] as $match) {
         $recur_list[]= $match;
      }
    }

  } //foreach $rep

  return implode($recur_list,"\r\n");

}

function parse_UCB_Event($event_id) {

  $ical_url =
"http://events.berkeley.edu/index.php/ical/event_ID/{$event_id}/.ics";
  $rsp = getResource($ical_url);

  # write out the file
  $tempfile = "temp.ics";
  $fh = fopen($tempfile,"wb");
  $numbytes = fwrite($fh, $rsp);
  fclose($fh);

  $v = new vcalendar(); // initiate new CALENDAR
  $v->parse($tempfile);

  # how to get to the prelude to the vevent? (timezone)

  #echo $v->getProperty("prodid");
  # get first vevent
  $comp = $v->getComponent("VEVENT");

  #print_r($comp);

  $event = array();

  $event["summary"] = $comp->getProperty("summary");
  $event["description"] = $comp->getProperty("description");

# optional -- but once and only once if these elements are here:
# dtstart, description,summary, url

  $dtstart = $comp->getProperty("dtstart", 1, TRUE);
  $event["dtstart"] = $dtstart;

# assume that dtend is used and not duration

  $event["dtend"] = $comp->getProperty("dtend", 1, TRUE);

  $event["location"] = $comp->getProperty("location");
  $event["url"] = $comp->getProperty("url");

# check for recurrence -- RRULE, RDATE, EXDATE, EXRULE

  $recurrence = extract_recurrence($rsp);

  $event_data = array();
  $event_data['event'] = $event;
  $event_data['recurrence'] = $recurrence;
  return $event_data;

} // parse_calendar

function extract_eventIDs($xml)
{

 $ev_list = array();

 foreach ($xml->channel->item as $item) {

   $link = $item->link;
   $k = parse_url($link);
   $ev_list[] = $k['fragment'];
 }
 return $ev_list;
}

// Google Calendar facade

function getClientLoginHttpClient($user, $pass)
{
  $service = Zend_Gdata_Calendar::AUTH_SERVICE_NAME;
  $client = Zend_Gdata_ClientLogin::getHttpClient($user, $pass, $service);
  return $client;
}

// code adapted from the Google documentation
// this posts to the DEFAULT calendar -- how do I change to post elsewhere?

function createGCalEvent ($client, $title, $desc, $where, $startDate = '2008-01-
20',
     $startTime = '10:00:00',
     $endDate = '2008-01-20', $endTime = '11:00:00', $tzOffset = '-08',
     $recurrence=null, $calendar_uri=null)
{
  $gdataCal = new Zend_Gdata_Calendar($client);
  $newEvent = $gdataCal->newEventEntry();

  $newEvent->title = $gdataCal->newTitle($title);
  $newEvent->where = array($gdataCal->newWhere($where));
  $newEvent->content = $gdataCal->newContent("$desc");

# if $recurrence is not null then set recurrence -- else set the start and
enddate:

  if ($recurrence) {
    $newEvent->recurrence = $gdataCal->newRecurrence($recurrence);
  } else {
    $when = $gdataCal->newWhen();
    $when->startTime = "{$startDate}T{$startTime}{$tzOffset}:00";
    $when->endTime = "{$endDate}T{$endTime}{$tzOffset}:00";
    $newEvent->when = array($when);
  } //if recurrence

// Upload the event to the calendar server
// A copy of the event as it is recorded on the server is returned

    $createdEvent = $gdataCal->insertEvent($newEvent,$calendar_uri);
    return $createdEvent;
}

function listEventsForCalendar($client,$calendar_uri=null) {

  $gdataCal = new Zend_Gdata_Calendar($client);

  $eventFeed = $gdataCal->getCalendarEventFeed($calendar_uri);
  foreach ($eventFeed as $event) {
    echo $event->title->text, "\t", $event->id->text, "\n";
    foreach ($event->when as $when) {
      echo "Starts: " . $when->startTime . "\n";
    }
  }
  echo "\n";
}

function clearAllEventsForCalendar($client, $calendar_uri=null) {
  $gdataCal = new Zend_Gdata_Calendar($client);

  $eventFeed = $gdataCal->getCalendarEventFeed($calendar_uri);
  foreach ($eventFeed as $event) {
    $event->delete();
  }

}

// bridge between UCB events calendar and GCal

function postUCBEventToGCal($client,$event_id, $calendar_uri=null) {

  $event_data = parse_UCB_Event($event_id);

  $event = $event_data['event'];
  $recurrence = $event_data['recurrence'];

  #print_r($event);
  #echo $recurrence;

  $title = $event["summary"];
  $description = $event["description"];
  $where = $event["location"];

# there is a possible parameter that might have TZ info. Ignore for now.
  $dtstart = $event["dtstart"]["value"];
  $startDate = "{$dtstart["year"]}-{$dtstart["month"]}-{$dtstart["day"]}";
  $startTime = "{$dtstart["hour"]}:{$dtstart["min"]}:{$dtstart["sec"]}";

# there is a possible parameter that might have TZ info. Ignore for now.
  $dtend = $event["dtend"]["value"];
  $endDate = "{$dtend["year"]}-{$dtend["month"]}-{$dtend["day"]}";
  $endTime = "{$dtend["hour"]}:{$dtend["min"]}:{$dtend["sec"]}";

  # explicitly set for now instead of calculating.
  $tzOffset = '-07';

  # I might want to do something with the url
  $description .= "\n" . $event["url"];

  echo "Event: ", $title,$description, $where, $startDate, $startTime, $endDate,
       $endTime, $tzOffset, $recurrence, "\n";

  $new_event = createGCalEvent($client,$title,$description, $where, $startDate,
       $startTime, $endDate, $endTime, $tzOffset,$recurrence, $calendar_uri);

}

# credentials for Google calendar

$USER = "[USER]";
$PASSWORD = "[PASSWORD]";
```

```
# the calendar to write to has a userID of
# n7irauk3nns30fuku1anh43j5s@group.calendar.google.com
# substitute the userID of your own calendar
$userID = urlencode("[USERID]");
$calendar_uri = "http://www.google.com/calendar/feeds/{$userID}/private/full";

$client = getClientLoginHttpClient($USER, $PASSWORD);

# get UCB events list

$cc_RSS = "http://events.berkeley.edu/index.php/critics_choice_rss.html";
$rsp = getResource($cc_RSS);

# for now, read the cached file
#$fname = "D:\Document\PersonalInfoRemixBook\examples\ch15\cc_RSS.xml";
#$fh = fopen($fname, "r");

#$rsp = fread($fh, filesize($fname));
#fclose($fh);

$xml = simplexml_load_string($rsp);
$ev_list = extract_eventIDs($xml);

echo "list of events to add:";
print_r($ev_list);

# loop through events list

# limit the number of events to do
$maxevent = 200;
$count = 0;

# clear the existing calendar

echo "Deleting existing events....";
clearAllEventsForCalendar($client,$calendar_uri);

# Add the events
foreach ($ev_list as $event_id) {

  $count +=1;
  if ($count > $maxevent) {
    break;
  }
  echo "Adding event: {$event_id}", "\n";
  postUCBEventToGCal($client,$event_id,$calendar_uri);

}

# list the events on the calendar
listEventsForCalendar($client,$calendar_uri);
?>
```

Summary

Here are some of things you learned in this chapter:

* You spent a considerable amount of time studying Google Calendar because of its sophisticated API and use of feeds including Atom feeds and iCalendar.
* You learned how to access and manipulate the feeds in Google Calendar, either by directly issuing the relevant RESTful HTTP requests with `curl` or by using the PHP and Python API kits.
* You took a quick look at 30boxes.com as another example of a web-based calendar with an API.
* You then studied how to consume feeds and exercise the APIs of two event aggregators: Upcoming.yahoo.com and Eventful.com.
* You studied how to program with iCalendar in PHP and Python.
* Finally, you learned how to synthesize an iCalendar feed from other iCalendar entries and how to write iCalendar information to a Google Calendar.

These are some key points to note:

* Online calendars are becoming more popular; they are especially useful when they have APIs and feeds to help with data integration.
* Event aggregators are interesting complements in this space to the online calendars.
* iCalendar is an important data exchange standard. There are variant forms that play off of it: hCalendar and parts of the Google Atom format for calendars.