PART 3

Making Mashups

Part 3 is the heart of the book. The previous chapters explained how to work with individual APIs and widgets, which are the raw materials of mashups. In this part, Chapter 9 talks about mashups in general and their relationship to APIs; the primary technique shown is to use ProgrammableWeb to learn about mashups. Chapter 10 covers the nitty-gritty of creating a mashup—using a specific example of mashing up Flickr and Google Maps. In Chapter 11, we'll study the tools that help you create mashups, while in Chapter 12, we'll look at the subject of mashup making from the point of view of API creators.

CHAPTER 9

Moving from APIs and Remixable Elements to Mashups

Now that you understand the pieces that go into mashups (remixable elements such as a rich URL language, tags, and feeds—all the subjects of Part 1) and APIs (the subject matter of Part 2), this chapter teaches you how to get a feel for how mashups are created from their pieces. To learn how to create mashups, you should study a lot of examples of mashups. In the next chapter, we will work out all the technical details of how to create a specific mashup. In this chapter, we'll step back to look at a broad range of problems that can be addressed with mashups. I won't work through all the details of how to create the mashup, but by thinking about a variety of problems—how others have created mashups to solve the problems themselves or related ones—you can learn about how to create mashups, what's possible, and what goes into them.

The primary technique we'll use to learn about mashups and APIs in this chapter is to mine ProgrammableWeb for information. ProgrammableWeb is the most useful web site for keeping up with the world of mashups, specifically, the relationships between all the APIs and mashups out there. It's by no means the only resource; you can't learn all you need from using it alone. However, learning how to use it effectively is a great way to make sense of the world of mashups as a whole.

To effectively understand mashup making, you should have a specific problem in mind that you want to solve. There's so much you can do with mashups that you will be

overwhelmed if you set out to assimilate 2,000+ mashups in ProgrammableWeb without a set purpose. In this chapter, I will use some specific problems and show how ProgrammableWeb can help you understand how to use mashups to solve these problems.

Specifically, I'll cover the following situations in this chapter:

Books: Integrating my varied interactions with books through mashups

Real estate search: Tracking houses coming onto the market and comparing them to estimates of worth

Travel search: Knowing when is a good time to buy airplane tickets

News: Using maps to understand current events around the world

Getting Oriented to ProgrammableWeb

You have already learned about ProgrammableWeb (`http://www.programmableweb.com/`) in this book. In Chapter 7, I discussed what you can learn about APIs, which are the major raw ingredients mashups, from ProgrammableWeb.

Here, I'll show you how to use ProgrammableWeb to learn about mashups. ProgrammableWeb is probably the most comprehensive database of web APIs and mashups and how they are related to one another. ProgrammableWeb and this book are complementary resources to learn about mashups. This book is focused on the nitty-gritty programming details of how to use APIs in creating mashups, and ProgrammableWeb covers the field in breadth and keeps pace with the fast-changing field. Note, however, that ProgrammableWeb doesn't claim to be comprehensive:[1]

> *You list a lot of mashups on this site. Are these all the mashups there are?*
>
> *No. This is a subset, or sample, of all mashups. The universe of web mashups is too large and dynamic to be cataloged in one place. And even that assumes that there's an agreed-upon single definition of what a mashup is. Which there isn't. That being said, this is probably the most diverse and structured collection available.*

One of the great features of ProgrammableWeb is that it covers APIs and mashups across a wide range of fields. Whereas API providers often link to applications that build upon their own APIs, ProgrammableWeb not only makes that information organized in a nice fashion, but it also lets you see how these APIs work with other APIs, which is really not usually of interest to any given API provider.

User-Generated Data in ProgrammableWeb

ProgrammableWeb depends deeply on user-generated profiles, as well as content entered by the people who run ProgrammableWeb. To participate in commenting or creating mashup or API entries, you can create an account on ProgrammableWeb here:

`http://www.programmableweb.com/register`

Registered users can create an entry for APIs or mashups and enter data about it here:

`http://www.programmableweb.com/add`

When you list a mashup at ProgrammableWeb, you can indicate what APIs are being used by the mashup.

Can Any Directory of Mashups Keep Up?

As mashups become more commonplace, we're going to be in a parallel situation to when Yahoo! went from being able to list every web site in a directory to needing search engines to crawl the Web to figure out what's on the Web. There will be way too many web sites that will use APIs in the future. Nonetheless, the practice you get here with the examples listed on ProgrammableWeb will help you recognize others in the wild. Until there is such a search engine for APIs and mashups that can automatically crawl for APIs, we will need a manual approach such as ProgrammableWeb.

1. http://www.programmableweb.com/faq#PWHowMany

Learning About the Overall Mashup Scene

You can follow ProgrammableWeb's own overview here:

`http://www.programmableweb.com/tour`

I will also highlight for you how to use it specifically to learn about mashups. Figure 9-1 shows the portal page for mashups on ProgrammableWeb.

`http://www.programmableweb.com/mashups`

Insert 858Xf0901.tif

Figure 9-1. ProgrammableWeb Mashup Dashboard

That page is a useful snapshot of the world of mashups that are in the ProgrammableWeb database. Here are some statistics listed on the page (as of January 13, 2008):

* The total number of mashups listed (2,661)
* The average rate of new mashups being added to the database (the six-month average was 3.14 per day)

I have found that this rapid growth of mashups makes it a challenge to keep up with everything that happens, even though some trends have remained quite stable (such as the popularity of map-based mashups).

Directory of Mashups

You can get a list of all the mashups in the database (by page) here:

`http://www.programmableweb.com/mashups/directory`

You can sort the list of mashups by the name of a mashup (which is the default view), the date when the mashup's profile was last updated, and the popularity of the mashup (the number of page views on ProgrammableWeb for that mashup). You can view the list as text, as "descriptive" (a mix of text and a thumbnail), or as a pure thumbnail view:

```
http://www.programmableweb.com/mashups/directory/{page}?sort={sort}&view={view}
```

where `sort` is one of `name`, `date`, or `popular` and where `view` is one of `text`, `desc`, or `images`.

For example:

```
http://www.programmableweb.com/mashups/directory/5?sort=popular&view=desc
```

Note that the popularity of APIs is measured by the number of mashups using that API:

```
http://www.programmableweb.com/popular
```

I like the idea of looking at the newest (if you are up on a field and want to see the latest) and the most popular (if you are new to a field and want to get a quick glance of what the scene is like). Comparing the newest and most popular mashups often helps to see what trends are afoot.

Indeed, you might be able to get the best of both by viewing a list of the top "popular new mashups" at `http://www.programmableweb.com/mashups`.

Using Feeds to Track Mashups

ProgrammableWeb uses techniques detailed in earlier chapters to help users not only track mashups but to create data about mashups. For instance, you can use the following RSS 2.0 feed to track new mashups on ProgrammableWeb:

```
http://feeds.feedburner.com/programmableweb/mashup
```

There are other feeds available such as the RSS 2.0 feed for blog entries:

```
http://feeds.feedburner.com/ProgrammableWeb
```

Here is the RSS 2.0 feed for the latest APIs:

```
http://feeds.feedburner.com/programmableweb/apis
```

You will find in the ProgrammableWeb blog (`http://blog.programmableweb.com/`) references to the APIs and mashup profile pages themselves. The ProgrammableWeb blog is an excellent place to read about the latest APIs and mashups of note, and it's also a thoughtful commentary about what these APIs and mashups mean.

Using Tags to Describe Mashups

Tags serve in ProgrammableWeb as thumbnail descriptions of what a given mashup or API is about. They are normalized to some degree to enable comparisons among mashups, in other words, to find similarities and patterns. I'll use these tags in this chapter to relate various mashups.

Tags associated with a given mashup are user-generated. That is, the user who creates a profile for a given mashup is allowed to use up to six tags that can be

associated with the mashup. Note the caveat on the link addition page, specifically, the admonition to "[u]se spaces between tags, no punctuation and limit to six tags please":

`http://www.programmableweb.com/add`

Also, the site will edit the entry to limit spam and ensure consistency—say, among tags.

You can see popular tags for mashups here:

`http://www.programmableweb.com/mashups`

Specifically, on this page you can see a pie chart of the top mashup tags for the last 14 days and for all time; this allows you to see how the current trends may or may not be deviating from long-term averages. Table 9-1 reproduces that information.

Table 9-1. The Percentage of Mashups in ProgrammableWeb Grouped by Tags (January 13, 2008)

Category	All	Last 14 Days
Mapping	40%	27%
Photo	10%	n/a
Shopping	9%	12%
Video	6%	12%
RSS	n/a	6%

This quick comparison attests to the long-term and short-term popularity of mapping. It looks like video mashups are on the rise—but you have to track it more to be sure. At any rate, if you keep an eye on the popular tags associated with mashups over time, you can get a feel for both short-term and long-term trends.

You can find the tag cloud of tags associated with the mashups here:

`http://www.programmableweb.com/search`

There you will find a short list of the top ten tags. Another worthwhile page is here:

`http://www.programmableweb.com/mashups/directory`

On the left side, you will find a list of the top 20 tags for mashups, along with the number of mashups for that tag. Here's a list of the ten most popular tags for mashups as of January 13, 2008:

* `mapping`
* `photo`
* `shopping`
* `search`
* `travel`
* `video`
* `news`
* `sports`

* realestate
* messaging

Note that the URL template to access the list of mashups by tag is as follows:

```
http://www.programmableweb.com/tag/{tag}
```

For example:

```
http://www.programmableweb.com/tag/mapping
```

You can page and sort and change views, too:

```
http://www.programmableweb.com/tag/{tag}/{page}?sort={sort}&view={view}
```

where `sort` is one of `name`, `date`, or `popular` and where `view` is one of `text`, `desc`, or `images`.

For example:

```
http://www.programmableweb.com/tag/mapping/2?sort=date&view=desc
```

Note that the tags associated with mashups are not necessarily the same as those for API tags, though you can expect some overlap. For example:

```
http://www.programmableweb.com/apitag/mapping
```

That brings up APIs that have been tagged with `mapping` and brings up mashups tagged with `mapping`:

```
http://www.programmableweb.com/tag/mapping
```

Note APIs are also classified in categories:

```
http://www.programmableweb.com/apis/directory/1?sort=category
```

API and Mashup Verticals

ProgrammableWeb calls out certain fields or segments with high activity as vertical markets for special attention:

```
http://www.programmableweb.com/markets
```

As of this writing, the special vertical markets (which are correlated to popular tags but not exactly) are as follows with upcoming markets for search, enterprise, and widgets:

* http://www.programmableweb.com/shopping
* http://www.programmableweb.com/government
* http://www.programmableweb.com/mapping
* http://www.programmableweb.com/telephony
* http://www.programmableweb.com/social
* http://www.programmableweb.com/video

If you take a look at one of these segments, you will see a dashboard (much like the main Mashup Dashboard) focused on that segment. One helpful extra is a description of the "big picture" for a segment, such as the one for telephony:

```
http://www.programmableweb.com/featured/telephony-mobile-apis-and-mashups
```

Why are verticals significant? That is, what does distinguishing verticals offer beyond just looking at the top mashup tags? You shouldn't be surprised that there would be significant overlap between the top mashup tags and the verticals. Certain verticals (such as government and telephony) are identified whose importance is not immediately apparent from tag popularity.

Looking at a Specific Mashup Profile

So far we've looked at the directory of mashups or collections of mashups grouped by tags or vertical markets. Let's consider how ProgrammableWeb displays a mashup profile.

You can find a profile for a given mashup here:

```
http://www.programmableweb.com/mashup/{mashup-handle}
```

For example, the profile for the Flash Earth mashup is here:

```
http://www.programmableweb.com/mashup/flash-earth
```

What do you find on the mashup profile page? For each mashup, you get the following:

* A description
* A screenshot
* The APIs involved in the mashup
* Any tags for the mashup
* The URL of the mashup
* When it was added and who added it
* Related mashups
* Associated comments and a rating (as a registered user, you can contribute comments and rate the mashup)

In this case, you learn that Flash Earth is a "[z]oomable mashup of Google Maps, Virtual Earth, and other satellite imagery through a Flash application" found here:

```
http://www.flashearth.com/
```

Tagged with the tag `mashup`, it involves the following APIs: Google Maps, Microsoft Virtual Earth, NASA, OpenLayers, and Yahoo! Maps. Moreover, you learn that Flash Earth is one of the most popular mashups on ProgrammableWeb.

In Chapter 13, you will take a closer look at online maps. Without figuring out how the various online map APIs actually work, you can—through playing with Flash Earth—learn that it is possible to extract tiles that make up various mapping APIs (for example, Google Maps, Yahoo! Maps, and Microsoft Virtual Earth) and recombine

them in a Flash interface. (Figuring out exactly how it's done is not necessarily so easy to do, though.) Flash Earth is a powerful demonstration of what is technically possible with online maps in a mashup.

Going from a Specific API to Mashups

In the previous section, you saw how a mashup profile lists the APIs that are used in the mashup. You can take a given API and find out all the mashups that use that API. For example, you start with a list of the most used APIs:

```
http://www.programmableweb.com/apis/directory/1?sort=mashups
```

Then you find the profile for the Google Maps API, the most popular of all APIs in ProgrammableWeb:

```
http://www.programmableweb.com/api/google-maps
```

From that link, you can click the Mashups link to arrive at the list of the 1,200+ mashups registered that use the Google Maps API:

```
http://www.programmableweb.com/api/google-maps/mashups
```

Sample Problems to Solve Using Mashups

Through a number of scenarios in which I describe some problems that are particularly suited to be solved through mashups, I'll show how you can use ProgrammableWeb to figure out what mashups might already exist to solve these problems. Often, there won't be a perfect—or even a good—solution, but the existing ones show you what is possible, what is easy to do, and what might be difficult to do. Moreover, by using ProgrammableWeb, you can immediately see what APIs are being used, as well as what mashups have gotten a following in the community of ProgrammableWeb readers.

Tracking Interesting Books

One scenario is to develop a system to handle book-related information in all the ways you might deal with books. Such a system would track books that

* you own,
* you have out from the various libraries and when they are due,
* you've lent to others or borrowed from others,
* you would like to read one day,
* you would buy if they dropped below a certain price,
* you'd buy from a used bookstore,
* you have just been published,
* you have just shown up your local libraries, or
* you cite in your writing.

Moreover, you probably want to keep some of this information private, some available only to friends, and some pieces of information completely public.

For my own books, I currently use a mishmash of web sites, desktop applications, and web applications to track books—all of which I would like to mash together:

* Amazon.com to look up and buy new books
* Amazon.com wishlists to store books that I would like to buy, borrow, or just ponder
* `http://worldcatlibraries.org/` to locate the book in my local library
* The online card catalogs of my local libraries (that of Berkeley Public Library and UC Berkeley)
* LibraryThing, a web site where I often enter books I'm reading and follow what others are reading
* Zotero (`http://zotero.org`), a Firefox extension that I use to track references
* Bn.com and other online bookstores
* `http://www.half.ebay.com/` to buy used books

What would I like to a complete book mashup to do? Lots of things—but some scenarios are as follows:

* If I place a book in my Amazon.com wishlist, I want to be informed whenever that book becomes available at any of the bookstores for which I have borrowing privileges.
* I want to synchronize books that I have listed in Zotero and LibraryThing.
* I want the due dates of all my library books to show up on my Google Calendar.
* I want to be able to format any subset of books from anywhere into a citation for the bibliography I'm compiling.

In some ways, the problem I'd like to solve is an elaboration of the problem I first discussed in Chapter 1. There I talked about the LibraryLookup bookmarklet that shows you how to find a library book in your local library catalog based on an ISBN. Here, I'd like my book information to flow easily among all the places I am referring to books.

I don't actually expect any existing mashup to bring them altogether—partly because mashups generally aren't that all-encompassing yet and partly because the mix of elements I want mashed up is rather idiosyncratic. But let's look at what mashups are out there, what they bring together, and whether we can mash up the mashups themselves.

Let's use ProgrammableWeb to help us to find possible solutions. One approach is to do a full-text search for *book* among the mashup profiles, sorting the results by popularity:

```
http://www.programmableweb.com/mashups/directory/1?q=book&sort=popular
```

Another approach is to use the tags. You can look through the tag cloud here to get a sense of popular book-related tags:

```
http://www.programmableweb.com/search
```

You'll see a link listed to the tag `books`. I recommend sorting by popularity first:

```
http://www.programmableweb.com/tag/books/1?sort=popular
```

and then by date to see the latest developments among mashups tagged with `books`:

```
http://www.programmableweb.com/tag/books/1?sort=date
```

I recommend looking through the results and reading the descriptions of each mashup. Try some. You'll get a feel for the range of possibilities among book-related mashups.

Here, I'll highlight ones that stand out in my viewing. One question to ask while looking through the mashup profiles is, what are the important APIs involved in these mashups? It would be nice to have ProgrammableWeb return the list of APIs involved in a given set of mashups sorted by the number of times it is used. In this case, such a feature would make it easy to see what the most commonly used APIs for mashups tagged with `books` are. However, even with a casual glance through this:

```
http://www.programmableweb.com/tag/books/1?sort=popular
```

you'll see several mentions of the Amazon.com E-Commerce Service API among the various APIs:

```
http://www.programmableweb.com/api/amazon-ecommerce
```

I'm sure you won't be surprised to see the Amazon.com web services show up in this context, given Amazon.com's prominence in two areas: online book retailing and web APIs. It's still helpful, though, to see how people have used the Amazon.com APIs in book-related mashups. You can use the advanced search on ProgrammableWeb (see the "Using the Advanced Search for Mashups and APIs" sidebar) to narrow down the list of mashups tagged with `books` to ones that use the Amazon.com E-Commerce Service API:

```
http://www.programmableweb.com/tag/books/1?apis=Amazon+eCommerce&sort=popular
```

This ability to focus a search on a specific API (or a combination of APIs) in conjunction with a specific mashup tag can often demonstrate the capabilities of an API for a specific context more vividly than reading the documentation for the API!

USING THE ADVANCED SEARCH FOR MASHUPS AND APIS

You can use an advanced search form to search for mashup and API profiles. For mashups, go here:

```
http://www.programmableweb.com/mashups/directory
```

Then hit the Advanced Search link, which will then open an Advanced Search option. With the Advanced Search option, you can specify the following:

* Up to three APIs used by the mashup (as you type the name of an API, the name will be autocompleted)
* Up to three tags associated with the mashup
* One of an optional date range when the mashup profile was created

For searching APIs, go here first:

http://www.programmableweb.com/apis/directory

Then click the Advanced Search link, which opens an Advanced Search form that lets you specify the following:

* Up to three tags associated with the API
* A category
* A company
* A protocol (for example, REST or JavaScript)
* An optional date range when the API profile was created

Let's now look at a few of the books-tagged mashups and how they can help in my quest to bring together all aspects of my book-related activities.

BlueOrganizer

BlueOrganizer is a Firefox extension that attempts to recognize when a web page is referring to items of certain categories such as wine, music, stocks, and—most important in this context—books:

http://www.programmableweb.com/mashup/blueorganizer

This ProgrammableWeb profile links to the URL for BlueOrganizer:

http://www.adaptiveblue.com/

specifically:

http://www.adaptiveblue.com/smartlinks_books.html

From reading the documentation for the plug-in, or actually installing it and trying it, you'll see that it recognizes books and gives you a button to take a variety of actions, including the following:

* Adding the book to the Amazon.com wishlist
* Adding the book to LibraryThing or Shelfari (a LibraryThing competitor)

From studying BlueOrganizer, you can learn about web sites that might provide useful sources of book information such as AbeBooks, which advertises an API in its affiliate program (but you need to contact them: http://www.abebooks.com/docs/AffiliateProgram/).

GuruLib, BookBump, and Other LibraryThing Analogs

Although I am a fan of LibraryThing, I follow the development of other web sites that allow readers to track books that they read and share that information with friends or the world. The *New York Times* covered this genre here:

http://www.nytimes.com/2007/03/04/business/yourmoney/04novel.html

Although I knew about Shelfari (http://www.shelfari.com) and Goodreads (http://www.goodreads.com) before consulting ProgrammableWeb, I learned about GuruLib and BookBump from ProgrammableWeb:

* http://www.programmableweb.com/mashup/gurulib
* http://www.programmableweb.com/mashup/bookbump

One thing that keeps me from investing too heavily in these sites is the struggle of how to move my book data in and out of any of these sites. For any given site, I look for APIs that help in that regard as well as any feeds that might allow users to easily import and export data.

Some Conclusions About Book Mashups

Here are some things that we learned by thinking through how to create a full-featured book mashup with the help of ProgrammableWeb:

* When it comes to book-related information, Amazon.com is a good API to start with in regard to book searching. The API gives you access to the Amazon.com wishlist (as you will see in Chapter 17).
* Don't expect all APIs of interest to be listed on ProgrammableWeb. As of writing, there is no mention of Zotero, WorldCat, and LibraryThing—even though they are all programmable to one degree or another.
* Lots of web sites I use don't have APIs at all, such as Bn.com and my local library catalogs. We are left with the question of how to deal with those sites. Should we screen-scrape the sites?
* There are other angles that won't be covered by looking only at ProgrammableWeb. For example, for the latest books at my library, I have to look at http://www.berkeley-public.org/ftlist for recent arrivals. There is no API.

Knowing When to Buy Airplane Tickets

Let's move from tracking books to tracking plane tickets. Suppose I want to buy a round-trip ticket between San Francisco and New York City. I know roughly when I want to travel but have some flexibility in terms of exactly when I can leave and return (within a day or two) and which airlines I can take. I'm planning far ahead of time to try to get the best price. However, I really don't want to have to leave before 8 a.m. or arrive in New York City after 9 p.m.

For a long time, it would be difficult for me as a typical consumer to be able to monitor over periods of weeks or months airfares for trips that meet such criteria so that I could wait for a good time to buy. However, some of the newer travel sites are giving users the ability to perform increasingly sophisticated searches, filter results by such criteria as the time of day of departure, and receive e-mail alerts for canned searches. Now, given that there are even travel web sites with APIs, I'm wondering whether I could use these APIs to get closer to finding the plane tickets at the prices I want.

We can use ProgrammableWeb to look through a collection of travel-tagged mashups:

http://www.programmableweb.com/tag/travel

John Musser wrote a recent analysis of the travel APIs:

http://blog.programmableweb.com/2007/10/29/5-travel-apis-from-comparison-to-booking/

You can search for travel-tagged APIs here:

http://www.programmableweb.com/apis/directory/1?q=travel

but if you limit the display to APIs tied to any mashup profiles (and sorting by popularity), like so:

http://www.programmableweb.com/apis/directory/1?q=travel&sort=mashups

you quickly find only two APIs at the time of writing:

* http://www.programmableweb.com/api/yahoo-travel
* http://www.programmableweb.com/api/kayak

Since the Yahoo! Travel API is focused on travel plans made by users on the Yahoo! Travel web site, and not on the purchase of airplane tickets, we'll focus then on the Kayak Search API:

http://www.kayak.com/labs/api/search/

Kayak (http://www.kayak.com/) is a web site like Expedia and Travelocity that allows users to search for flights. Given that APIs for travel sites are a new concept, I wasn't surprised that there were few mashups listed as using the Kayak Search API (http://www.programmableweb.com/api/kayak/mashups). Kayak's deals from cell phones profiled here made me think of alternate interfaces to Kayak's travel information:

http://www.programmableweb.com/mashup/kayak.com-deals-from-cell-phone

One mashup that I have yet to see is one of Kayak with Google Calendar. When I schedule flights, it's useful to see what else I have going on in my personal schedule. A Kayak/Google Calendar mashup could present possible flights as a Google calendar that I could juxtapose with my personal calendar. The mashup might even read my personal schedule to filter out prospective flights to begin with. (See Chapter 15 for how to use the Google Calendar API.)

Apart from connecting Kayak to alternative interfaces such as cell phones and calendars, a mashup of Kayak could allow you to conduct a more thorough search through the complicated combinations of parameters possible when flying. I found that manually varying parameters such as departure dates and return dates and keeping the best deals in my head got rather tiring after five to ten tries. I suspect that a mashup of the Kayak Search API and a smart search algorithm could possibly find better flights than I could find manually.

Finding That Dream House

Real estate–oriented APIs and mashups promise to make home buying a bit easier and maybe more fun. Specifically, let's look at how we might use a variety of web feeds and APIs to track houses that come on the market in a given area.

You can start on ProgrammableWeb by searching for mashups tagged with realestate and sorting the results by popularity:

http://www.programmableweb.com/tag/realestate/1?view=desc

More to the point, by searching for APIs tagged with `realestate` and listing them by popularity by going here:

```
http://www.programmableweb.com/apitag/realestate/1?sort=mashups
```

you find two relevant APIs for American real estate:

* Zillow (`http://www.programmableweb.com/api/zillow`)
* Trulia (`http://www.programmableweb.com/api/trulia`)

Zillow (`http://www.zillow.com/`) focuses on providing estimates of home valuations and details of properties, while also listing homes for sale and that have been recently sold. Trulia (`http://www.trulia.com/`) aggregates listings of homes for sale. Note that although the Trulia API (`http://developer.trulia.com/`) doesn't currently return any individual listings, you can use Trulia feeds to access some of the listings. For example, the following is an RSS 2.0 feed of some of the current properties available in Berkeley, California:

```
http://www.trulia.com/rss2/CA/Berkeley/
```

Currently, I do not know of what seems to be an obvious combination—a mashup of the Zillow and Trulia APIs, one that, for instance, would compare the sale price of houses listed for sale on Trulia with what Zillow estimates to be the value of the house. ProgrammableWeb doesn't list any such mashup:

```
http://www.programmableweb.com/mashups/directory/1?apis=trulia%2Czillow
```

Something I learned by looking through the `realestate` mashups is that Google Base is an interesting source of real estate data. Take a look at mashups tagged by `realestate` using the Google Base API:

```
http://www.programmableweb.com/tag/realestate?apis=Google+Base
```

Mapping Breaking News

In Chapter 4, you learned how to use Yahoo! Pipes to pull together various news feeds into a single feed. In this section, I'll cover how to plot those current events on a map.

I often read about places in the world for which I have only the vaguest idea where they are located. Online maps certainly make it easy to look places up now. But much like how Housingmaps.com helps with visualizing real estate on a map, perhaps displaying news on a map of the world could have the similar benefits.

Let's see what ProgrammableWeb has to say about mashups of news and maps. The `news` tag is a popular tag for mashups on ProgrammableWeb.

```
http://www.programmableweb.com/tag/news/1?view=desc
```

When you look through this section, you'll see several mashups of interest:

* `http://www.programmableweb.com/mashup/bbc-news-map` points to a now-defunct mashup that mapped BBC News items about the United Kingdom on a map.
* `http://www.programmableweb.com/mashup/ap-national-news-google-maps` points to `http://www.81nassau.com/apnews/`, which displays items from a choice of Associated Press feeds (including national news, sports, and business) on a Google map.

* http://www.programmableweb.com/mashup/mapified-rss points to http://o.gosselin.free.fr/Projects/MapifiedRss/, which maps to Google Maps entries from one of the preconfigured RSS feeds (for example, Reuters, Associated Press top headlines, or Google News) or from the URL of a feed entered the user.

Seeing these mashups reminded me how easy it is now to display feeds that contain geographic locations on a map. Let's use Yahoo! Pipes, which you have already learned how to use in Chapter 4. The key to georeferencing a feed so that it can be displayed on a map is the Location Extractor Operator in Yahoo! Pipes:

```
http://pipes.yahoo.com/pipes/docs?doc=operators#LocationExtractor
```

Another thing you need to know is that Yahoo! Pipes is able to output KML that can then be displayed on Google Earth and Google Maps. (Chapter 13 contains more details about KML.)

I constructed a Yahoo! pipe that takes as input a URL to a feed to be georeferenced:

```
http://pipes.yahoo.com/raymondyee/locationextractor
```

The default value for this URL is that for the *New York Times* International News feed:

```
http://www.nytimes.com/services/xml/rss/nyt/International.xml
```

The KML output for this default feed is as follows:

```
http://pipes.yahoo.com/pipes/pipe.run?InputURL=http%3A%2F%2Fwww.nytimes.com%2Fservices.....................................................................
...
```

or as follows:

```
http://tinyurl.com/yvx8qy
```

You can display this KML feed on a Google map, like so:

```
http://maps.google.com/maps?f=q&hl=en&geocode=&time=&date=&ttype=&q=http:%2F%2F~CCC
pipes.yahoo.com%2Fpipes%2Fpipe.run%3FInputURL%3Dhttp%253A%252F%252Fwww.nytimes.com%252F~CCC
services%252Fxml%252Frss%252Fnyt%252FInternational.xml%26_id%3DcInT4D7B3BGMoxPNiXrLOA%~CCC
26_render%3Dkml&ie=UTF8&ll=28.921631,53.4375&spn=150.976999,360&z=2&om=1
```

or like so:

```
http://tinyurl.com/yp8k2b
```

Since in this section we're looking at mapping, it's helpful to look at the mapping vertical market coverage on ProgrammableWeb:

```
http://www.programmableweb.com/mapping
```

Summary

In this chapter, you learned about mashups and their relationships to APIs by studying a series of specific problems for which mashups can provide useful solutions. You looked

at how you can track books, real estate, airfare, and current events by combining various APIs. You used ProgrammableWeb to help analyze these problems.