CHAPTER 11

Using Tools to Create Mashups

In the previous chapters, we focused on creating mashups through combining a number of technologies: XML, PHP, JavaScript, Python, and so on. As you've seen, creating mashups takes some amount of skill and knowledge. Can specialized tools make it easier to create mashups? In this chapter, I'll introduce several mashup-making tools and show the basics of how to use a few of them. My goal in this chapter is to pick ones that are popular, useful, and illustrative of some important issues or trends.

The focus of this chapter will be on using the Google Mashup Editor (GME) (in conjunction with Yahoo! Pipes) to build a Flickr/Google Maps mashup; it's like the one you saw in Chapter 10, but this one will show you how mashup-making tools make it easier (or different) from straight-up PHP and JavaScript programming, which you saw in Chapter 10. Specifically, this chapter includes the following:

* A tutorial on the GME
* A way of comparing mashup-making tools with the straight-up programming you did in previous chapters

I'll briefly describe some other tools (Microsoft Popfly, Dapper, and others). By essentially re-creating the Flickr/Google Maps mashup using the GME and Yahoo! Pipes, you'll see what mashup tools get you and what they don't get you.

The Problem Mashup Tools Solve

With a mashup-oriented mind-set in which you prize being able to integrate data and services (because integration brings value), you care about how to introduce new things that can be well integrated with the stuff you already use, without a high cost (in learning or effort to make those bridges).

Currently, creating mashups takes a lot of work to knit APIs together. You've already seen that work in the previous chapters and will see it again in the chapters that follow. APIs are similar to one another, but they are all somewhat different. One day, perhaps, you'll be able to mash up different APIs the same way that you can surf from one web site to another and make sense of them all. It may never be that easy, but it should be easier than how it is today.

Case in point: although WSDL is not perfect, something like WADL (roughly WSDL for RESTful APIs) would be really helpful in alleviating the syntactic complications of connecting stuff, if not exactly making autosemantic connections possible.

Another major problem is that getting data to work together from different APIs requires translating among them. RDF promises to be a close-to-universal data

representation. I have always thought there is some merit to the semantic web stack, at the bottom of which is RDF. Hence, I'm keeping an eye on RDF and other semantic web tools. I can certainly believe that if more and more data were expressed in RDF, things would connect better. RDF is, however, not a panacea. (That's why I think a study of the SIMILE project (`http://simile.mit.edu`) is a practical way to start looking at semantic web technology.)

Specifically, when you use a new API, you want to minimize the type of new learning; you want the API to be similar to what you've already seen before. I think of the DRY ("don't repeat yourself") principle: it's nice to not have to tackle the same thing repeatedly—solve it once, run it anywhere. For example, if you write code to access the Flickr API in one language and in one framework, wouldn't it be nice to be able to not have to redo that code when you have another need? That's the ideal anyway.

Tools and services can help you reach these goals. The key aspect of good tools is that they simplify the routine stuff and let you concentrate on the essence of the problem. There are always trade-offs, however—and we'll look at them as we look at the various tools.

What You Are Making in This Chapter

As mentioned, the focus of this chapter will be using the GME in conjunction with Yahoo! Pipes to build a Flickr/Google Maps mashup. Let's first look at the final product that I will show you how to build step by step. Doing so will teach you about the GME and Yahoo! Pipes (when you load the mashup, you will be asked to log in using your Google e-mail address because this app requires authentication):

```
http://mashup-raymond-yee-flickrfeed4.googlemashups.com/
```

This mashup displays geotagged photos on a Google map. There are two tabs: Search and Saved Results. Let's consider the Search tab (see Figure 11-1). Upon loading, the map is set to Berkeley, California—and recent geotagged photos around that area are loaded in a list form and also displayed on a Google map. Remember that when you use the Flickr API to search for geotagged photos, you need to specify a geographic region (a *bounding box*) in which you are searching for photos. This mashup map shows a rectangle to denote this search bounding box.

Insert 858Xf1101.tif

Figure 11-1. The Search tab of the Flickr/Google Maps mashup

When you enter a tag, a search is done on geotagged photos with the specified tag within the current bounding box of the map. Up to 100 results are loaded at a time; you can page through the photos.

Notice that there is a Copy Selected button. You can select a photo and hit that button, which saves the selected photo to your saved entries feed, which you can then see on the Saved Results tab (see Figure 11-2). On this second tab, you find a list of the Flickr photos that you have saved in addition to a map showing the locations of the saved photos. You'll also see a Delete Selected button that lets you remove a selected photo from your feed of saved entries.

Insert 858Xf1102.tif

Figure 11-2. The Saved Results tab of the Flickr/Google Maps mashup

Now let's build this mashup step by step.

Making the Mashup: A Step-by-Step Example

In this section, you will be redoing and extending Chapter 10's Flickr/Google Maps mashup. First you'll see how to reproduce the example, and then you'll extend it because of the data persistence that is possible with the GME. One of the distinctive elements of the GME is its ability to persist data with feeds, which is an elegant approach. It allows for the extensibility of data elements in an easy-to-fashion way (for example, you'll see that there's no need to predefine a data schema).

Even though this chapter is about mashup tools in general, I will use the GME and Yahoo! Pipes first and then survey a few others. Yahoo! Pipes has been out for a while and has proven itself to be very useful. Though the GME is a new product, it is a promising one, backed also by a big company. There are similarities but also important differences with Yahoo! Pipes; the two are actually complementary, as I will show in the extended example of this chapter. That example will cover the following:

* Familiarizing yourself with the GME
* Reading and displaying a feed
* Introducing a custom template
* Using Yahoo! Pipes to access the Flickr API
* Displaying Flickr photos using `<gm:map>`
* Adding JavaScript to update the feed parameters and give the coordinates of the map
* Learning how to persist feeds and use tabs
* Making the final product

Familiarizing Yourself with the Google Mashup Editor

The GME (`http://editor.googlemashups.com/editor`) is a browser-based environment that makes it easier to create mashups that are hosted on Google.

You can find documentation on how to use the GME here:

```
http://code.google.com/gme/
```

The GME is targeted at programmers familiar with HTML, CSS, and JavaScript, and it is a programming environment. You can intermix browser-side techniques in HTML, CSS, and JavaScript with GME tags. This combination is compiled into web pages (with HTML, CSS, and JavaScript) that run on modern browsers—the GME tags are converted into JavaScript.

Note As this book goes to press, the Google Mashup Editor is still under closed beta.

You don't have to use the browser-based editor to use the GME. See the sidebar "Using Subversion (SVN) to Access Your Project" to learn about accessing your code via SVN.

You should keep the tag reference located at the following URL handy while programming with the GME:

```
http://editor.googlemashups.com/docs/reference.html
```

Contrast the GME to Yahoo! Pipes. The GME is a text-based programming environment, while Yahoo! Pipes is a visual programming environment.

The first step is to save a new project and give it a display title (for example, "Flickr on Google Maps") and a single-word/lowercase string for a project name (`flickrgmap`). Look at the starter code for `index.gml` for a new project:

```
<gm:page title="My App" authenticate="false">

</gm:page>
```

The next step is to customize it to have it say something like this:

```
<gm:page title="Flickr Photos on Google Maps" authenticate="false">

<!--
Displaying Flickr Photos on a Google Map
@author: Raymond Yee
-->

<h1>Flickr Photos</h1>

</gm:page>
```

After you have copied and pasted the previous code, I suggest just getting familiar with the environment. That is, hit Save, and enter a display name and project name. Then hit Test to see your basic application compile; you'll be brought to the Sandbox tab to see the application run.

Reading and Displaying a Feed (Simple Template)

The GME makes it easy to read and write Atom and RSS feeds. Let's get a Flickr Atom or RSS feed flowing into the GME. Go to the Feed Browser tab in the GME; in the list of GData feeds, you can choose from the Test feed (which you can use to test your code), the Remote Feed feed, and the Google Base feed.

Note Currently, the GME is unable to read in XML data other than RSS and Atom 2.0 feeds. You can use Yahoo! Pipes to convert XML to RSS 2.0, which can in turn be processed by GME.

Let's give it a feed of geotagged photos from Flickr Central:

```
http://api.flickr.com/services/feeds/geo/?g=34427469792@N01~CCC
&lang=en-us&format=rss_200
```

Use the Feed Browser tab's Remote Feed to read it in and see how the RSS 2.0 feed is converted to Atom 1.0. The native format for the GME is Atom 1.0, but the GME can accept RSS 2.0 and Atom 1.0. Feeding the RSS 2.0 feed to the GME Feed Browser tab shows how feeds are converted to Atom 1.0, and you refer to the data elements in the Atom format. For example, the GME makes extensive use of *GPath*, a small subset of XPath expressions, to refer to specific elements or attributes in a feed.[1] For instance, the GPath expression `atom:title` will return the title of an RSS item, which has been converted to an Atom element.

1. http://code.google.com/gme/docs/data.html#xpath

Let's create a simple GME application that reads and displays the feed, using a `<gm:list>` tag to specify the URL of the feed. You tell GME how to display this feed by using a *template*. In the next section, I'll show you how to create a custom template. In this section, you'll use one of the built-in templates (specifically `simple`) to display the feed, as shown here:

```
<gm:page title="Flickr Photos on Google Maps" authenticate="false">

<!--
Displaying Flickr Photos on a Google Map
@author: Raymond Yee
-->

<h1>Flickr Photos</h1>

<gm:list id="flickrList" template="simple"
         data="http://api.flickr.com/services/feeds/geo/?g=34427469792@N01
&lang=en-us&format=rss_200"
         pagesize="10"/>

</gm:page>
```

Note the types of built-in templates: `simple` (which lists only the title of each entry), `task` (which displays a check box to mark a task as done, the due date, and the priority), and `blog` (which is a template particularly suited to feeds coming from blogs).

Introducing a Custom Template

Now let's use `<gm:template>` to customize how the feed is displayed—specifically to get the image in the list:

```
<gm:page title="Flickr Photos on Google Maps" authenticate="false">

<!--
Displaying Flickr Photos on a Google Map
@author: Raymond Yee
-->

<h1>Flickr Photos</h1>
```

```
  <gm:list id="flickrList" template="flickrTemplate"

data="http://api.flickr.com/services/feeds/geo/?g=34427469792@N01&lang=~CCC
en-us&format=rss_200" pagesize="10"/>

  <gm:template id="flickrTemplate">
    <table class="blue-theme" style="width:50%">
      <tr repeat="true">
        <td style="padding-bottom:10px">
          <b><gm:text ref="atom:title"/></b>
          <br/>
          <gm:html ref="atom:summary"/>
          <br/>
          <span style="color:#3366cc">
            location: (<gm:text ref="geo:Point/geo:lat"/>,
                       <gm:text ref="geo:Point/geo:long"/>)
          </span>
        </td>
      </tr>
    </table>
  </gm:template>

</gm:page>
```

You can see the code here:

`http://mashup-raymond-yee-flickrfeed1.googlecode.com/svn/trunk/index.gml`

And you can run the app here:

`http://mashup-raymond-yee-flickrfeed1.googlemashups.com/`

Note the following about this template:

* There are built-in CSS classes; this example uses `blue-theme`.
* This example uses `<tr repeat="true">` to repeat a `<tr>` for each Atom entry. This is useful because there is no need to write a loop explicitly.
* The template for each entry displays the text of the title (`<gm:text ref="atom:title">`) and the HTML in the summary (`<gm:html ref="atom:summary"/>`).
* See how you can get at the geotag of each entry through the GPath entries `geo:Point/geo:lat` and `geo:Point/geo:long`. You can use the Feed Browser tab to help you figure out the XPath by hovering over the element you want to access.

USING SUBVERSION (SVN) TO ACCESS YOUR PROJECT

Instead of using the browser-based editor to edit your code, you can use Subversion (`http://en.wikipedia.org/wiki/Subversion_(software)`) to download and check in your edits. You will find basic documentation of how to use SVN in the context of the GME here:

`http://code.google.com/support/bin/answer.py?answer=76145&topic=11689`

Each mashup project you create with the GME generates a separate project hosted by Google Code. To find a list of these projects on Google Code, visit the following URL:

```
http://code.google.com/hosting/
```

Click Settings to get your Google Code password, which you need for Subversion. Click the My Profile tab to go to your list of projects. In my case, the My Profile tab leads to the following:

```
http://code.google.com/u/raymond.yee/
```

Consider a specific project. One project I created through GME is available here:

```
http://code.google.com/p/mashup-raymond-yee-flickrfeed2/
```

You can browse the code here:

```
http://mashup-raymond-yee-flickrfeed2.googlecode.com/svn/
```

The code `index.gml` for the project is available here:

```
http://mashup-raymond-yee-flickrfeed2.googlecode.com/svn/trunk/index.gml
```

Note that, by default, the code you produce using GME is licensed under an Apache 2.0 license. You can change it on the Administer tab:

```
http://code.google.com/p/mashup-raymond-yee-flickrfeed2/admin
```

You can find instructions for using SVN to check out the code here:

```
http://code.google.com/p/mashup-raymond-yee-flickrfeed2/source
```

Once your code is checked out, you use your favorite desktop editor instead of being confined to editing source through a web browser. Moreover, it's also easier to edit many files simultaneously rather than depending on the browser-based editor in which you can currently edit only one file at a time.

Using Yahoo! Pipes to Access Flickr

The GME needs RSS 2.0 or Atom—and cannot read XML feeds in general. Yahoo! Pipes, on the other hand, can be used to read XML in general and emit RSS 2.0. The Flickr API doesn't currently output RSS 2.0 and Atom 1.0, but rather its own custom XML (although as we learned in Chapter 4, there is an extensive selection of feeds from Flickr).

Here I'll show you a Yahoo! Pipe that is created to be an interface to `flickr.photos.search`. That is, you can input the same parameters as you can to `flickr.photos.search`, but instead of getting Flickr XML, you get RSS 2.0. Here's the pipe I generated:

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

You can also find it here:

```
http://pipes.yahoo.com/pipes/pipe.info?_id=YG9eZGWO3BGukZGJTqoASA
```

The parameters for this pipe are similar to the `flickrgeo.php` server-side proxy that you wrote for Chapter 10 (and will see again in Chapter 13). The parameters are almost

the same as the parameters you'll find for `flickr.photos.search` with the following exceptions:

* This pipe handles only unauthenticated searches.
* Instead of `bbox` to denote the bounding box, the pipe uses `lat0,lon0,lat1,lat1`.
* There is no use of a `format` or `o_format` parameter (as for `flickrgeo.php`) since the pipe controls the output.
* The pipe has some default parameters to search for geotagged photos around downtown Berkeley, California.

As to how to create this pipe, refer to the tutorial on pipes in Chapter 4. I'll mention a few possibly tricky parts here. (You can check out the source of the pipe on the Yahoo! Pipes site.) First, you need to create a text input or number input for each of the parameters (this process is a bit tedious given there are 24 input parameters).

To convert the Flickr XML to RSS 2.0, you can use several *loops*:

* A loop to create an `item.image_prefix` for each item. The `item.image_prefix` is used as the first part of a URL to point to Flickr images of various sizes.
* A second loop to create `item.image_small_URL` by concatenating `item.image_prefix` with `_s`.
* A third loop to calculate and assign `item.link`—a link to the Flickr page of the photo.
* A fourth loop to calculate and assign `item.description`, which holds HTML for the small square version of the Flickr photo.

Note that the last two loops are the ones that directly affect the translation of the Flickr XML to RSS 2.0. Finally, the pipe uses the Location Extractor module[2] to extract the longitude and latitude from the Flickr results into `<geo:lat>` and `<geo:long>` for each photo.

Using the default parameters with RSS 2.0 output, here's the code:

```
http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGW03BGukZGJTqoASA&_render=rss~C
CC
&api_key={api_key}&extras=geo&lat0=37.817785166068&lat1=37.926190569376&lon0=~C
CC
-122.34375&lon1=-122.17208862305&min_upload_date=820483200&per_page=10
```

If you wanted KML output, you'd use the following:

```
http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGW03BGukZGJTqoASA&_render=kml
```

RETAINING NONCORE RSS 2.0 ELEMENTS IN YAHOO! PIPES

I was hoping that the RSS 2.0 feed emitted by the pipe would retain nonstandard elements calculated by the pipe (for example, `<image_prefix>`) that would make calculating the URL for various image sizes more straightforward. It turns out that the RSS 2.0 feed doesn't have this information—although the JSON feed does indeed contain the extra parameters.

```
http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGWO3BGukZGJTqoASA&_render=~C
CC
json&api_key={api_key}&extras=geo&lat0=37.817785166068&lat1=37.926190569376
&~CCC
lon0=-122.34375&lon1=-122.17208862305&min_upload_date=820483200&per_page=10
```

One way to solve this problem is to write a web service in PHP to transform the JSON to RSS 2.0 with extension elements—but that defeats the purpose of trying to make this stuff easier.[3]

Displaying Flickr Photos Using <gm:map>

Remember that the overall goal in this iteration is to display the Flickr photos on a Google map. Now that you have access to the Flickr API (via the pipe you just created), you can change the source of Flickr photos to the pipe that accesses `flickr.photos.search` as the source of images:

```
http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGWO3BGukZGJTqoASA&_render=rss~CC
C
&api_key={api_key}&extras=geo&lat0=37.817785166068&lat1=37.926190569376&lon0=~C
CC
-122.34375&lon1=-122.17208862305&min_upload_date=820483200&per_page=100
```

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

3. http://discuss.pipes.yahoo.com/Message_Boards_for_Pipes/threadview?m=tm&bn=pip-DeveloperHelp&tid=2513&mid=2517&tof=-1&rt=2&frt=2&off=1

Here the parameters to the pipe are hard-coded; later I'll show you how to construct a form to let a user change the parameters.

I'll first show you the resultant code for this iteration and then explain the pieces:

```
<gm:page title="Flickr Photos on Google Maps" authenticate="false">

<!--
Displaying Flickr Thumbnails on a Google Map (hardwired parameters)
@author: Raymond Yee
-->

  <h1>Flickr Photos</h1>

  <gm:list id="flickrList" template="flickrTemplate"

data="http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGWO3BGukZGJTqoAS~CCC
A&_render=rss&api_key=e81ef8102a5160154ef4662adcc9046b&extras=geo&lat0=37.817785
~CCC
166068&lat1=37.926190569376&lon0=-122.34375&lon1=-
122.17208862305&min_upload~CCC
date=820483200&per_page=100" pagesize="10">
     <gm:handleEvent event="select" src="flickrMap"/>
  </gm:list>
```

```
  <gm:map id="flickrMap" style="border:solid black 1px" control="large"
          maptypes="true" data="${flickrList}" latref="geo:lat"
lngref="geo:long"
            infotemplate="FlickrMapDetailsTemplate" height="600">
      <gm:handleEvent event="select" src="flickrList"/>
   </gm:map>

  <!-- flickrTemplate -->

  <gm:template id="flickrTemplate" class="blue-theme">
    <div style="float:left; width:85px" repeat="true">
      <gm:html ref="atom:summary"/>
    </div>
    <br style="clear:both"/>
    <gm:pager/>
  </gm:template>

  <!-- FlickrMapDetailsTemplate -->

  <gm:template id="FlickrMapDetailsTemplate">
    <div >
      <b><gm:link ref="atom:link[@rel='alternate']/@href" labelref="atom:title"
/>
      </b>
      <br/>
      <gm:html ref="atom:summary"/>
      <br/>
      Lat: <gm:text ref="geo:lat"/><br/>
      Long: <gm:text ref="geo:long"/>
    </div>
  </gm:template>

</gm:page>
```

I published this app at the following location:

```
http://mashup-raymond-yee-flickrfeed2.googlemashups.com/
```

You'll see how I changed the `data` attribute in `<gm:list>` to point to the new source of data. In addition, I revised `<gm:template>` to use `<div>` instead of a `<table>` and to display only the thumbnail. Moreover, I added a `<gm:pager/>` to enable the user to page through the images ten at a time.

Next, you use a `<gm:map>` element to instantiate a map:

```
  <gm:map id="flickrMap" style="border:solid black 1px" control="large"
          maptypes="true" data="${flickrList}" latref="geo:lat"
lngref="geo:long"
          infotemplate="FlickrMapDetailsTemplate" height="600">
      <gm:handleEvent event="select" src="flickrList"/>
   </gm:map>
```

Note how the parameters for `<gm:map>` are constructed:

* The `data` attribute (set to `${flickrList}`, which is the ID of your `<gm:list>`) makes the tie to the input data source defined in the `<gm:list>` element.

* The `latref` and `lngref` attributes are the XPath (or in the parlance of the GME, the GPath) expressions relative to a feed entry to get at the latitude and longitude. (You can use the Feed Browser tab to determine this quantity.)
* The `<gm:handleEvent>` tells the map to respond to a `select` event from the `flickrList`. That is, when a user clicks a photo in the `<gm:list>`, the corresponding marker on the map pops open.
* In a fashion similar to the template for the Flickr thumbnails, you can create a template to control how the bubbles on the map are displayed.

Note in general how this declarative approach replaces having to write a lot of HTML and JavaScript. Indeed, the mashup we created in Chapter 10 doesn't have this interaction between the display of thumbnails and the markers on the map.

Adding JavaScript to the Mashup

The goals of the next pass of development are as follows:

* Pass a subset of the parameters (instead of having hard-coded parameters) to Yahoo! Pipes.
* Let the user set the bounding box of the search, and draw a bounding box on the map to indicate this bounding box.

Before jumping into doing this, you might want to consult the sidebar "Introducing Custom JavaScript into the GME."

INTRODUCING CUSTOM JAVASCRIPT INTO THE GME

Before I try to introduce a new element (in this case some JavaScript event handling) into the main code I'm working on, I often like to write a little side program to test this idea. The following is a simple program that involves an input form and a submission event, reminiscent of a simple example from Chapter 10 (`http://examples.mashupguide.net/ch10/square2.html`):

```
<gm:page title="Squaring Input and Flickr feed" authenticate="false">
<!--
  Introducing custom JavaScript into a GME mashup
-->

    <form action="#" onsubmit="calc_square(); return false;">
      <label>Input a number:</label>
      <input type="text" size="5" name="num" value="4" />
      <input type="submit" value="Square it!" />
    </form>

    <p>The square of the input is: <span id="answer">16</span></p>

  <gm:list id="flickrList" template="flickrTemplate"

data="http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGWO3BGukZGJTqoA~CCC
SA&_render=rss&api_key={api_key}&extras=geo~CCC
```

```
   &lat0=37.817785166068&lat1=37.926190569376&lon0=-122.34375&lon1=-
   122.17208862305~CCC
   &min_upload_date=820483200&per_page=100"
              pagesize="10" />

     <!-- flickrTemplate -->

     <gm:template id="flickrTemplate" class="blue-theme">
       <div>
         <span repeat="true" style="padding: 5px">
           <gm:html ref="atom:summary"/>
         </span>
         <gm:pager/>
       </div>
     </gm:template>

     <script type="text/javascript">
       //<![CDATA[
         function calc_square() {
            var n = document.forms[0].num.value;
            document.getElementById('answer').innerHTML = n*n;
         }

         document.forms[0].num.onchange = calc_square;  //register an event

       //]]>
     </script>
   </gm:page>
```

When you run this code, you'll see that a lot of the JavaScript techniques for event handling can be directly interspersed with the GME tags.

You can find the code I created to accomplish these goals here:

```
http://mashup-raymond-yee-flickrfeed3.googlecode.com/svn/trunk/index.gml
```

You can run the code here:

```
http://mashup-raymond-yee-flickrfeed3.googlemashups.com/
```

Here's the code:

```
<gm:page title="Flickr Photos on Google Maps" authenticate="false"
         onload="init_data();">

<!--
Displaying Flickr Thumbnails on a Google Map
@author: Raymond Yee
-->

  <h1>Flickr Photos</h1>

  <form action="#" onsubmit="update_feed(); return false;">
    <label>Input tags:</label><input type="text" size="30" name="tags" value=""
/>
    <input type="submit" value="Update feed" />
  </form>
```

```
  <p>URL of current feed: <span id="current_tags">.</span></p>

  <gm:list id="flickrList" template="flickrTemplate" pagesize="10">
    <gm:handleEvent event="select" src="flickrMap"/>
  </gm:list>

  <gm:map id="flickrMap" style="border:solid black 1px" control="large"
          maptypes="true" data="${flickrList}" latref="geo:lat"
lngref="geo:long"
             infotemplate="FlickrMapDetailsTemplate" height="600">
      <gm:handleEvent event="select" src="flickrList"/>
  </gm:map>

  <!-- flickrTemplate -->

  <gm:template id="flickrTemplate" class="blue-theme">
    <div style="float:left; width:85px" repeat="true">
      <gm:html ref="atom:summary"/>
    </div>
    <br style="clear:both"/>
    <gm:pager/>
  </gm:template>

  <!-- FlickrMapDetailsTemplate -->

  <gm:template id="FlickrMapDetailsTemplate">
    <div >
      <b><gm:link ref="atom:link[@rel='alternate']/@href" labelref="atom:title"
/>
      </b>
      <br/>
      <gm:html ref="atom:summary"/>
      <br/>
      Lat: <gm:text ref="geo:lat"/><br/>
      Long: <gm:text ref="geo:long"/>
    </div>
  </gm:template>

  <script type="text/javascript">
    //<![CDATA[
     function update_feed() {
         var tags = document.forms[0].tags.value;

      // let's get the bounds of the map
         var flickrMap = google.mashups.getObjectById('flickrMap');

         var bounds = flickrMap.getBounds();
         var lat0 = bounds.getSouthWest().lat();
         var lon0 = bounds.getSouthWest().lng();
         var lat1 = bounds.getNorthEast().lat();
         var lon1 = bounds.getNorthEast().lng();

         update_feed0 (tags,lat0,lon0,lat1,lon1);
```

```
      } // update_feed

  function update_feed0(tags,lat0,lon0,lat1,lon1) {

      var flickrList = google.mashups.getObjectById('flickrList');
      var flickrMap = google.mashups.getObjectById('flickrMap');
      var url =
'http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGW03BGukZGJ~CCC
TqoASA&_render=rss&api_key=e81ef8102a5160154ef4662adcc9046b&extras=geo&min_~CC
C
upload_date=820483200&per_page=100' + '&tags=' + escape (tags) + "&lat0="~CCC
        + lat0 + "&lon0=" + lon0 + "&lat1=" + lat1 + "&lon1=" + lon1;

      // clear the old overlays (I'm doing this to get rid of the boundary

      flickrMap.getMap().clearOverlays();

      document.getElementById('current_tags').innerHTML =
        "<a href='" + url + "'>Feed Link</a>";

      flickrList.setData(url);
      flickrList.setPage(0);  // reset the pager

      // now draw a bounding box

      border = new GPolygon([
      new GLatLng(lat0, lon0),
      new GLatLng(lat1, lon0),
      new GLatLng(lat1, lon1),
      new GLatLng(lat0,lon1),
      new GLatLng(lat0,lon0)
      ], "#ff0000", 2);

      flickrMap.getMap().addOverlay(border);

  } // update_feed0

    function init_data() {

        var lat0=37.817785166068;
        var lat1=37.926190569376
        var lon0=-122.34375;
        var lon1=-122.17208862305;

        update_feed0("",lat0,lon0,lat1,lon1);

    } // init_data

    //]]>
  </script>

</gm:page>
```

The code in bold was what changed from the previous code. Recall that the overall goal of this iteration is to go from a hard-coded Flickr search with a set of hard-coded parameters to a search that uses the map to determine the bounding box and also that allows the user to specify the Flickr tag on which to search. To implement this functionality, you can add an HTML form to accept input tags. You also create the function `update_feed()` to respond to a form submission in the following ways:

* Calculates the appropriate URL to the Yahoo! Pipes to search for geotagged Flickr photos that match the tags and are found in the given bounding box. The `init_data()` method, which sets the initial location for the map, calls `update_feed()`, which in turn calculates the URL to Yahoo! Pipes and uses the `setData(url)` method of the `<gm:list>` to load the data.
* Draws a bounding box on the map to mark the current search area.

How to Persist Feeds and Use Tabs

The next major task to try is the data persistence aspects of the GME. In this section, you'll learn how to create a simple database, specifically, a feed to save Flickr images that a user finds interesting. These feeds persist between sessions. (That is, when users log out and return to the mashup, they can find their saved results as they left them.) You will learn also how to copy data from one feed to another. Finally, you will learn how to use the tab support in the GME.

Specifically, I'll show you how to build a mashup that has two tabs. The first tab (Search) shows Flickr images from a hard-coded feed. Each photo is displayed with a button to let a user copy the image to a feed of saved photos. The second tab (Saved Results) displays this feed of saved photos.

The code I wrote to implement this design, which can be found here:

`http://mashup-raymond-yee-tabs0.googlecode.com/svn/trunk/index.gml`

is shown here:

```
<gm:page title="Tabs0" authenticate="true">
  <!--
  Load the feeds in one tab and allow to copy selected entries to a data source
  in another tab
  -->

  <gm:tabs target="myContainer"/>

  <gm:container id="myContainer"
                style="padding:3px;border:1px solid #369;width:600px;">

    <gm:section id="sectionFlickrSearch" title="Search">
       <gm:list id="myList" template="flickrTemplate"

data="http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGW03BGukZGJTqoA~CCC
SA&_render=rss&extras=geo&lat0=37.817785166068&lat1=37.926190569376&lon0=-
122.34375&~CCC
lon1=-122.17208862305&min_upload_date=820483200&per_page=10" pagesize="10"/>
    </gm:section>
```

```
  <gm:section id="sectionSavedEntries" title="Saved Results">
    <gm:list id="savedEntries" data="${user}/crud"
             template="savedEntryTemplate" />
  </gm:section>

</gm:container>

 <gm:template id="flickrTemplate">
   <table class="blue-theme" style="width:50%">
      <tr repeat="true">
        <td style="padding-bottom:10px">
          <b><gm:text ref="atom:title"/></b>
          <br/>
          <gm:html ref="atom:summary"/>
          <br/>
          <span style="color:#3366cc">
             location: (<gm:text ref="geo:lat"/>, <gm:text ref="geo:long"/>)
          </span>
          <br/>
          <input type="button" value="Copy" onclick="copy_this(this)" />
        </td>
      </tr>
  </table>
</gm:template>

 <gm:template id="savedEntryTemplate">
   <div>Your saved entries</div>
   <table class="blue-theme" style="width:50%">
      <tr repeat="true">
        <td style="padding-bottom:10px">
          <b><gm:text ref="atom:title"/></b>
          <br/>
          <gm:html ref="atom:summary"/>
          <br/>
          <span style="color:#3366cc">
             location: (<gm:text ref="geo:lat"/>, <gm:text ref="geo:long"/>)
          </span>
          <gm:editButtons deleteonly="true" />
        </td>
      </tr>
  </table>
</gm:template>

<script type="text/javascript">
//<![CDATA[

function copy_this(DOMElement) {
  var entry = google.mashups.getEntryForElement(DOMElement);
  var myList = google.mashups.getObjectById('myList');
  var savedEntries = google.mashups.getObjectById('savedEntries');
  savedEntries.getData().addEntry(entry);
}

//]]>
```

```
  </script>

</gm:page>
```

You can run this mashup here:

```
http://mashup-raymond-yee-tabs0a.googlemashups.com/
```

Let's look at how this code works:

* Three tags are used to create the two tabs. A `<gm:tabs>` tag is used to instantiate a set of tabs, with a target attribute pointing to a `<gm:container>` element that in turn holds a `<gm:section>` for each tab.[4]
* The `copy_this()` function is invoked when the Copy button corresponding to a thumbnail is clicked. This function identifies the feed entry matching the selected DOM element and copies the element to the feed whose ID is `savedEntries`.
* The `authenticate` attribute of the `<gm:page>` element is set to `true`, so users must sign in to a Google account to use the application.

Once a user creates a collection, you can access the resulting feeds. See the instructions here:

```
http://code.google.com/support/bin/answer.py?answer=76140&topic=11689
```

The generic URL of a user feed is as follows:

```
{PUBLISHED_MASHUP_NAME}.googlemashups.com/feeds/public/user/{USER_EMAIL}/~CCC
<STRIPE_NAME>
```

`STRIPE_NAME` is essentially an identifier for a feed. In the case of our mashup, the feed of my saved entries is available here:

```
http://mashup-raymond-yee-
tabs0a.googlemashups.com/feeds/public/user/raymond.yee~CCC
%40gmail.com/crud
```

The following relative URL—relative to the logged-in user, that is—also works:

```
http://mashup-raymond-yee-tabs0a.googlemashups.com/feeds/user/crud
```

This feed works only from a browser with the right credentials (that is, cookies from a logged-in user in a browser).

Note When writing GME feeds, keep in mind that the maximum number of entries in a custom feed is 1000.

---

4. http://code.google.com/gme/docs/samples.html#tabs

Changing the Selection and Deletion Process for the Photos

The previous code attaches a copy button to each image. In this section, you'll rewrite the code to switch to showing small thumbnails horizontally and to have a Copy

Selected button on the Search tab. Similarly, let's add a Delete Selected button to the Saved Results tab.

The code for the new version, available from here:

http://mashup-raymond-yee-tabs1.googlecode.com/svn/trunk/index.gml

is as follows:

```
<gm:page title="Tabs1" authenticate="true">
  <!--
  Load the feeds in one tab and allow to copy selected entries to a data source
  in another tab
  -->

  <gm:tabs target="myContainer"/>

  <gm:container id="myContainer" style="padding:3px;border:1px solid #369;">

    <gm:section id="sectionFlickrSearch" title="Search">
      <gm:list id="flickrList" template="flickrTemplate"

data="http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGW03BGukZGJTqoAS~CCC
A&_render=rss&api_key=e81ef8102a5160154ef4662adcc9046b&extras=geo&lat0=37.817785
1660~CCC
68&lat1=37.926190569376&lon0=-122.34375&lon1=-
122.17208862305&min_upload_date=820483~CCC
200&per_page=100"
               pagesize="10" />
      <input type="button" value="Copy Selected" onclick="copy_selected()" />
    </gm:section>

    <gm:section id="sectionSavedEntries" title="Saved Results">
      <gm:list id="savedEntries" data="${user}/crud"
               template="savedEntryTemplate" />
      <input type="button" value="Delete Selected" onclick="delete_selected()"
/>
    </gm:section>

  </gm:container>

  <!-- flickrTemplate -->

  <gm:template id="flickrTemplate" class="blue-theme">
    <div style="float:left; width:85px" repeat="true">
      <gm:html ref="atom:summary"/>
    </div>
    <br style="clear:both"/>
    <gm:pager/>
  </gm:template>

  <!-- savedEntryTemplate -->

  <gm:template id="savedEntryTemplate">
     <div>Your saved entries</div>
     <div style="float:left; width:85px" repeat="true">
```

```
      <gm:html ref="atom:summary"/>
    </div>
    <br style="clear:both"/>
    <gm:pager/>
  </gm:template>

  <script type="text/javascript">
  //<![CDATA[

  // figure what is the currently selected entry and copy that over
  function copy_selected() {

    var flickrList = google.mashups.getObjectById('flickrList');
    var entry = flickrList.getSelectedEntry();
    if (entry) {
      var savedEntries = google.mashups.getObjectById('savedEntries');
      savedEntries.getData().addEntry(entry);
    }

  } // copy_selected

  function delete_selected() {

    var savedEntries = google.mashups.getObjectById('savedEntries');
    var entry = savedEntries.getSelectedEntry();
    if (entry) {
      savedEntries.getData().removeEntry(entry);
    }

  } // delete_selected

  //]]>
  </script>

</gm:page>
```

You can run the new code here:

```
http://mashup-raymond-yee-tabs1.googlemashups.com/
```

The key changes to the code are as follows:

* Switching from a vertical to a horizontal template and adding a single Save Selected button and Delete Selected button to the tabs—instead of having a separate button for each photo entry
* Adding the appropriate event handlers (`copy_selected()` and `delete_selected()`)

The Final Product: Showing the Saved Entries on a Map

You are now ready to create the final product with the GME. You can do so by embedding the search and display of geotagged Flickr photos on a Google map from here:

```
http://mashup-raymond-yee-tabs1.googlemashups.com/
```

with the overall framework of two tabs to hold search results and saved entries:

http://mashup-raymond-yee-flickrfeed3.googlemashups.com/

The final product is at http://mashup-raymond-yee-flickrfeed4.googlemashups.com/.

The code is at http://mashup-raymond-yee-flickrfeed4.googlecode.com/svn/trunk/index.gml; it's as follows:

```
<gm:page title="Flickr Photos on Google Maps" authenticate="true"
         onload="init_data();">

<!--
Displaying Flickr Thumbnails on a Google Map (flickrfeed4)
@author: Raymond Yee
-->

  <gm:tabs target="myContainer"/>

  <gm:container id="myContainer" style="padding:3px;border:1px solid #369;">

      <!-- searchFlickrSearch section
      -->

    <gm:section id="sectionFlickrSearch" title="Search">

      <h1>Flickr Photos</h1>

      <form action="#" onsubmit="update_feed(); return false;">
        <label>Input tags:</label>
        <input type="text" size="30" name="tags" value="" />
        <input type="submit" value="Update feed" />
      </form>

      <p>URL of current feed: <span id="current_tags">.</span></p>

      <gm:list id="flickrList" template="flickrTemplate" pagesize="10">
        <gm:handleEvent event="select" src="flickrMap"/>
      </gm:list>
      <input type="button" value="Copy Selected" onclick="copy_selected()" />

      <gm:map id="flickrMap" style="border:solid black 1px" control="large"
              maptypes="true" data="${flickrList}" latref="geo:lat"
              lngref="geo:long"
              infotemplate="FlickrMapDetailsTemplate" height="600">
          <gm:handleEvent event="select" src="flickrList"/>
       </gm:map>

   </gm:section>

   <!-- sectionSavedEntries -->

   <gm:section id="sectionSavedEntries" title="Saved Results">

      <gm:list id="savedEntries" data="${user}/crud"
template="savedEntryTemplate" />
```

```
      <input type="button" value="Delete Selected" onclick="delete_selected()"
/>

      <gm:map id="flickrMap2" style="border:solid black 1px" control="large"
              maptypes="true" data="${savedEntries}" latref="geo:lat"
              lngref="geo:long"
              infotemplate="FlickrMapDetailsTemplate" height="600">
          <gm:handleEvent event="select" src="savedEntries"/>
       </gm:map>

    </gm:section>

  </gm:container>

<!-- flickrTemplate -->

  <gm:template id="flickrTemplate" class="blue-theme">
    <div style="float:left; width:85px" repeat="true">
      <gm:html ref="atom:summary"/>
    </div>
    <br style="clear:both"/>
    <gm:pager/>
  </gm:template>

<!-- FlickrMapDetailsTemplate -->

  <gm:template id="FlickrMapDetailsTemplate">
    <div >
      <b><gm:link ref="atom:link[@rel='alternate']/@href" labelref="atom:title"
/>
      </b>
      <br/>
      <gm:html ref="atom:summary"/>
      <br/>
      Lat: <gm:text ref="geo:lat"/><br/>
      Long: <gm:text ref="geo:long"/>
    </div>
  </gm:template>

  <!-- savedEntryTemplate -->

  <gm:template id="savedEntryTemplate">
     <div>Your saved entries</div>
     <div style="float:left; width:85px" repeat="true">
      <gm:html ref="atom:summary"/>
     </div>
     <br style="clear:both"/>
     <gm:pager/>
  </gm:template>

  <script type="text/javascript">
    //<![CDATA[
     function update_feed() {
```

```
        var tags = document.forms[0].tags.value;

      // let's get the bounds of the map
        var flickrMap = google.mashups.getObjectById('flickrMap');

        var bounds = flickrMap.getBounds();
        var lat0 = bounds.getSouthWest().lat();
        var lon0 = bounds.getSouthWest().lng();
        var lat1 = bounds.getNorthEast().lat();
        var lon1 = bounds.getNorthEast().lng();

        update_feed0 (tags,lat0,lon0,lat1,lon1);

      } // update_feed

    function update_feed0(tags,lat0,lon0,lat1,lon1) {

      var flickrList = google.mashups.getObjectById('flickrList');
      var flickrMap = google.mashups.getObjectById('flickrMap');

      var url =

'http://pipes.yahoo.com/pipes/pipe.run?_id=YG9eZGW03BGukZGJTqoASA&_render=~CC
C
rss&api_key=e81ef8102a5160154ef4662adcc9046b&extras=geo&min_upload_date=82048320
0&~CCC
per_page=100'
          + '&tags=' + escape (tags) + "&lat0=" + lat0 + "&lon0=" + lon0
          + "&lat1=" + lat1 + "&lon1=" + lon1;

      // clear the old overlays (I'm doing this to get rid of the boundary)

      flickrMap.getMap().clearOverlays();

      document.getElementById('current_tags').innerHTML =
          "<a href='" + url + "'>Feed Link</a>";

      //a lert('flickrList' + flickrList);
      // alert('url' + url);

      flickrList.setData(url);
      flickrList.setPage(0);  // reset the pager

      // now draw a bounding box

      border = new GPolygon([
      new GLatLng(lat0, lon0),
      new GLatLng(lat1, lon0),
      new GLatLng(lat1, lon1),
      new GLatLng(lat0,lon1),
      new GLatLng(lat0,lon0)
      ], "#ff0000", 2);

      flickrMap.getMap().addOverlay(border);
```

```
  } // update_feed0

    function init_data() {

        var lat0=37.817785166068;
        var lat1=37.926190569376
        var lon0=-122.34375;
        var lon1=-122.17208862305;

        update_feed0("",lat0,lon0,lat1,lon1);

    } // init_data

  // figure what is the currently selected entry and copy that over
    function copy_selected() {

      var flickrList = google.mashups.getObjectById('flickrList');
      var entry = flickrList.getSelectedEntry();
      if (entry) {
        var savedEntries = google.mashups.getObjectById('savedEntries');
        savedEntries.getData().addEntry(entry);
      }

    } // copy_selected

    function delete_selected() {

      var savedEntries = google.mashups.getObjectById('savedEntries');
      var entry = savedEntries.getSelectedEntry();
      if (entry) {
        savedEntries.getData().removeEntry(entry);
      }

    } // delete_selected

    //]]>
  </script>

</gm:page>
```

Note When writing your first apps, you might wonder why the data doesn't persist between sessions. Publish your mashup first.

At this point, you have a mashup that covers much of the GME's capabilities, and you can continue developing this mashup. Some possible further steps include the following:

* You could incorporate the GME's annotation support, which would allow users to add tags and ratings to feed entries (`http://code.google.com/gme/docs/data.html#annotations`).
* You could create a better input form to let a user enter any of the input parameters that can be fed to the Yahoo! pipe. It would be nice to have collapsible input boxes for a streamlined interface. Connecting the GME to some nice Ajax widget libraries would be helpful.
* You could minimize redundancy by packaging reusable code into modules. Currently, GME files cannot include other GME files. Once the GME lets programmers create reusable modules, you can then rewrite the mashups in this chapter to pull out common features in different tabs.

Analysis of Trade-Offs in Using GME and Yahoo! Pipes

Now that you have created mashups of Flickr and Google Maps using both specialized mashup tools (a combination of the GME and Yahoo! Pipes) and general-purpose web programming techniques (PHP and JavaScript), let's compare these two approaches. First, consider that the GME and Yahoo! Pipes provide the following to you as a developer:

* Hosting is provided; you don't need to use your own server.
* Instead of PHP and JavaScript, you program only in JavaScript. This can be considered an advantage in that you don't have to know PHP to use the GME and Yahoo! Pipes. Google and Yahoo! are doing the server-side proxying for you.
* You don't need to register for separate API keys for Google Maps to use the maps.
* You get access to a Subversion interface, to issue tracking, and to the other features of Google Code, which is used to host the GME code.
* Both the GME and Yahoo! Pipes make it easier to see what others are building; hence, they promote the sharing of tips, ideas, and code.

There are, of course, trade-offs you make by using the GME, Yahoo! Pipes, and possibly other third-party tools:

* Each tool generally presents a new framework to learn. Some are easier to learn than others, often by building on what you are likely to know from other contexts. However, there is always something new to learn.
* Sometimes the abstractions used by a given tool are not quite what you want. For instance, the central data exchange format is Atom feeds, which can be either an apt simplification or a burdensome limitation.
* Having your application hosted on the GME or Yahoo! Pipes means revealing your data and code to Google or Yahoo!

* The identity and branding of your mashup will be associated with Google or Yahoo!
* You become dependent on the infrastructure of Google and Yahoo! Lock-in could become a problem.

There are a couple of things that it would be nice to get from Yahoo! Pipes and the GME:

* Being able to host your code on your own server. If Google were one day to let you compile GME code into HTML and JavaScript that could then be modified and run independently of the GME, that would increase GME's attractiveness to many developers and users.
* GME does not have the ability to read in information other than RSS and Atom feeds. Right now, Yahoo! Pipes fills that niche well—by using pipes to read in XML and then converting it to feeds, you can then process that data in GME. However, the GME being able to process XML beyond RSS and Atom would be a useful feature.

I think a measure of success for tools such as the GME and Yahoo! Pipes is the degree to which they let you easily build applications for a specific purpose; and these are applications that you don't even mind throwing away after a single use because they were so easy to write. By this measure, the GME and Yahoo! Pipes moves you toward tools to create such apps. I think that the GME and Yahoo! Pipes makes it easier for programmers to create certain types of mashups, though it's not so clear whether they open up mashup development for a nonprogramming audience.

Other Mashup Tools

Many other tools are designed to help in creating mashups. There are so many mashup tools to look at—more than I can do justice to here. Table 11-1 lists a number of them along with a brief description and a URL for how you can learn more. Some use browser-based interfaces, while others are desktop tools. Some focus on specific aspects of creating mashups, while others aim to be a unifying framework.

Table 11-1. Other Mashup Tools

Name	Description	URL
Apatar	Open source software designed for business users and programmers to integrate data sources and formats	http://www.apatar.com/product.html
BEA AquaLogic Pages	Browser-based tools for authoring web pages and web applications	http://www.bea.com/framework.jsp?CNT=index.jsp&FP=/content/products/aqualogic/pages/
Bungee Connect	Browser-based environment for building web applications	http://www.bungeelabs.com/
Chickenfoot	Firefox extension (similar to Greasemonkey) that allows users to write scripts to "manipulate web pages and automate web browsing"	http://groups.csail.mit.edu/uid/chickenfoot/index.php

Coghead Browser-based GUI for creating and hosting business applications http://www.coghead.com/

CoScripter Firefox extension "that automates the process of recording and playing back processes" http://services.alphaworks.ibm.com/coscripter/browse/about

Dapper Browser-based GUI for producing screen-scrapers that output Atom, RSS, Google Maps, and other formats http://www.dapper.net

Data Mashups Online Service Browser-based GUI to create custom business applications, especially for mashing up data and web services http://datamashups.com/

Denodo data mashup A platform focused on creating new business services by integrating existing data http://www.denodo.com/

Extensio A platform for data extraction, integration, and delivery http://www.extensio.com/

Intel Mash Maker A experimental research project for enabling the easy creation of mashups ("mashups for the masses") http://mashmaker.intel.com/

JackBe Presto Enterprise Edition Software to let users "create, consume, and customize enterprise mashups"http://jackbe.com/resources/download.php

Marmite A research prototype of a mashup creation tool for nonprogrammers http://www.cs.cmu.edu/~jasonh/projects/marmite/

Microsoft Popfly A browser mashup tool whose drag-and-drop widgets have some similarity to Yahoo! Pipes but that puts more of an emphasis on presentation gadgets http://popfly.ms

Openkapow Desktop software for creating bots hosted by openkapow http://openkapow.com/

Potluck The mashup-making tool for SIMILE, a research project focused on the application of the semantic Web for manipulating digital assets http://simile.mit.edu/potluck

Proto Desktop mashups, especially for business intelligence applications http://www.protosw.com/

QEDWiki A browser-based interface for creating mashups http://services.alphaworks.ibm.com/qedwiki/

RSSBus Tools to "generate, manage, orchestrate, and pipeline RSS feeds" http://rssbus.com/

Serena Mashup Composer Software for creating business mashups http://www.serena.com/mashups/testdrive.html

StrikeIron SOA Express for Excel An add-on to enable easy access to web services from within Microsoft Excel http://strikeiron.com/tools/tools_soaexpress.aspx

WSO2 Mashup Server An open source platform for creating and deploying "web services mashups" http://wso2.org/projects/mashup

Summary

Several tools are available that make it easier to create mashups. In this chapter, you explored the topic primarily by using a combination of two such tools, the Google Mashup Editor and Yahoo! Pipes, to create a mashup of geotagged Flickr photos and Google Maps. By creating this mashup in a number of manageable steps, you learned

how to use the GME and incorporate Yahoo! Pipes. Moreover, by comparing the process used in this chapter to that used in the previous chapter, you get to see some of the advantages and trade-offs involved in using mashup tools instead of general-purpose web programming techniques.