CHAPTER 5

# Doing the Monster Mash: A Mashup

A long time ago, some god-like developer came up with the concept of an Application Programming Interface (API). In short, an API is nothing but a known (to those that might use it) interface to a program or system. The developer came up with this idea, and everyone saw it, and saw that it was a Good Thing™. But, in the immortal words of Dr. Leonard H. McCoy, "... engineers, they love to change things." We couldn't just stick with the term API. No, we just *had* to come up with something new, and that something is the term *mashup*. Since it's a term that is all the rage these days, and also something that often involves JavaScript to a large degree, it's most definitely an appropriate topic for this book. So, in this chapter, I'll introduce the concept of the mashup, and then we'll put that concept to use in a handy little application.

## What's a Mashup?

A mashup, as it has come to be known, is basically a web site or application that takes content from multiple sources, most usually via some sort of public programmatic interface, and integrates it all into a new experience—that is, a new application. If this sounds a bit like the promise of web services to you, you aren't too far off. In fact, web services are sometimes involved in mashups, although that setup typically involves a server infrastructure and some server-side code, *vis-à-vis*. You won't typically call on *true* web services (SOAP, UDDI, WSDL, and all that the term *web services* typically denotes) from a JavaScript client, and almost certainly not from a web browser (not without plug-ins or similar technology, generally).

No, in recent times, the term mashup has generally come to mean browser-based JavaScript clients aggregating content through public APIs from various companies and vendors to form new applications. These APIs are often referred to as web services, and even though they may not truly be web services in the sense of using the full technology stack, they fulfill the same basic goal as those types of web services. They provide services and function over a network—specifically, the Web—so calling them web services isn't really too far-fetched!

Many companies are getting into the API business, including companies you've certainly heard of: Google, Yahoo, Amazon, and eBay, just to name a few. Google and Yahoo have really led the charge, and Yahoo, in particular, originated a neat trick that will be central to the application we'll build in this chapter.

Mashups are also a part of what people often mean when they use the term Web 2.0. Web 2.0 means different things to different people, but sharing resources is usually part of what people mean by it, so mashups certainly fit right in.

One of the other things that is often lumped under Web 2.0 is effects. Take a look at a site like Digg, for instance. I was going to insert a screenshot here, but truthfully, it wouldn't get the point across because it has to be seen live. So please visit `http://www.digg.com`, if you aren't already a frequent visitor (and you should be, by the way!) and just look around a bit. As you do, take note of the various effects. For instance, assuming you aren't signed in, try to click the Digg It button next to an article. Notice how the text is faded and you get a little pop-up over it telling you about signing up? Now, if you create an account, sign in, and try clicking that button again, you'll see the Digg count fade out, then fade back in with the new value. These are all examples of the kinds of UI effects that most consider a part of Web 2.0.

# Monster Mash(up) Requirements and Goals

Now, with all of that about mashups in mind, let's discuss what this chapter's application will do and what it will demonstrate.

- The basic function of the application is to be able to enter a ZIP code and to get from that a list of hotels.
- For each hotel, we should be able to click its name and see some extended information.
- In addition to extended information, we would also like to see a map of the area.
- We should be able to zoom in and out on the map.
- The UI will use various effects provided by the well-known script.aculo.us library.
- We will utilize some APIs from Google and Yahoo to get the hotel information and maps.
- This application should be purely browser-based and not require any server component to function.

All of this will result in an application that is fully buzzword-compliant and very much fits the most common definition of Web 2.0 applications. Let's start by taking a look at the Yahoo and Google APIs.

# The Yahoo APIs

Yahoo did something very cool a short while ago, and it is this one cool thing that makes the application in this chapter possible. Before we can discuss that though, we have to discuss what was going on before the coolness occurred.

For a while now, many companies have been exposing public APIs for people to use, Yahoo among them. For instance, you could perform a Yahoo search remotely, or you could get a Yahoo map from your own application, and so on. These APIs—these "web services," if you will—usually used XML as their data-transport mechanism. You would post some XML to a given URL, and you would get an XML response back. It was (and still is) as simple as that. These types of services don't require all the web services like SOAP, UDDI, WSDL, and the like.