Now that we have covered what we will be learning throughout the course of this book, it’s time to get our hands dirty by jumping in and creating our first mobile web applications. Both of these applications will be very simple applications that interact with the Twitter API. An API, or an application programming interface, is an interface used to interact with a specific application based off a set of rules or parameters given to the application by the user. More often than not, one would use an API to gather and parse data from the application’s infrastructure without ever needing to directly connect to the application’s database to fetch that information.

The first application we will build is the application “Who’s That Tweet?”, which will parse through a small predefined list of verified Twitter users and display a random tweet on the page, as well as a list of four possible individuals that might have created that tweet.

The second small web application we will build is called “I Love Ham”, which will also be borrowing Twitter’s fire hose of data to create a quick and fun form of entertaining for mobile users on the go. “I Love Ham” will focus on very basic game mechanics. The user is presented with two predefined rhyming Twitter searches to choose from. If they choose the Twitter search that receives the most results back, then they are the winners. Both of these mobile web games will rely heavy on HTML5, JavaScript, and a fantastic piece of technology called JSONP.

In this chapter, we’ll get you up and running with a development environment on your own personal computer (think of it as your own mini-Internet, which will connect to Twitter on the real Internet, but mostly live just on your computer) and discuss the Who’s That Tweet? application. In the next chapter, we’ll continue by discussing the I Love Ham application. Let’s get started!
CHAPTER 2: Twitter Applications: Who's That Tweet?

JSONP

You might remember us talking about JSON in chapter one. JSON is a wonderful piece of technology and chances are if you have worked in the web development field in the past few years, then you have probably run across it once or twice. If you've surfed the web in the past day, you've also probably run across a website using it! If you have ever worked with the APIs of Flickr, Twitter, or Gowalla, then chances are you are also very familiar with JSON. For those that skipped over chapter one completely, JSON is a human readable data interchange technology that is made to be lightweight and is also an open standard. As a technology, JSON is fairly young with its usage being traced back originally to a little after the turn of the century with the JSON.org (http://www.json.org) website launching in 2002.

If you are familiar using JavaScript and creating objects, then JSON should look relatively familiar to you. However, JSON is to not be confused with a JavaScript object as they are not the same. As much as I love JSON, there is one tiny little annoying thing that most developers are not aware of until they spend hours ripping out their own hair prematurely making themselves go bald and that is a really nasty, little, pesky thing called cross-domain scripting.

To understand what cross-domain scripting is and why there are hardcoded rules in place in your browser to prevent you from doing it, you have to first understand what cross-site scripting (XSS) is. XSS is a type of vulnerability in web sites and applications that allows an attacker or intruder to basically inject their own scripts and content into the site or system. An intruder might use an XSS vulnerability to upload a script to your server that would allow them to simulate a Bash shell (a common UNIX shell or command-line interface) for your server in their very own browser, or worse yet, possibly use that vulnerability to steal thousands of usernames and passwords from your database.

Due to these concerns, your browser will not load JSON data from a domain outside of your site’s server in most cases. In order to grab and use that data, it would have to be grabbed on the backend server side before the page is loaded and then served to the user. This doesn’t seem like that big of a deal until you start building applications that have thousands upon thousands of users. Then parsing that external data can end up impacting the performance of your server. To get around this performance issue and delegate some of those external data parsing requirements to the end user, we will use JSONP or “JSON with padding”, which will allow our application to request the JSON data we need because it will be wrapped (or padded) within a JavaScript object.

Setting Up Your Development Environment

Before we can get started coding, we will need to set up a development environment. For the sake of simplicity, I will assume that we are using a Windows–based operating systems so I will only go through the steps of setting up a basic WAMP (Windows, Apache, MySQL, and PHP) server for us to test our code on. I know not everyone uses