Sams Teach Yourself

the Twitter API

in 24 Hours
## Contents at a Glance

<table>
<thead>
<tr>
<th>Hour</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xiii</td>
<td>Preface</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>What Is Twitter?</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Twitter Out of the Box</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>Key Issues to Consider When Developing Twitter Applications</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>Creating a Development Environment</td>
<td>33</td>
</tr>
<tr>
<td>5</td>
<td>Making Your First API Call</td>
<td>49</td>
</tr>
<tr>
<td>6</td>
<td>Building a Simple Twitter Reader</td>
<td>59</td>
</tr>
<tr>
<td>7</td>
<td>Creating a Twitter API Framework</td>
<td>73</td>
</tr>
<tr>
<td>8</td>
<td>Twitter OAuth</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>Building a Simple Twitter Client, Part I</td>
<td>95</td>
</tr>
<tr>
<td>10</td>
<td>Building a Simple Twitter Client, Part II</td>
<td>105</td>
</tr>
<tr>
<td>11</td>
<td>Expanding Our Client for More API Calls</td>
<td>113</td>
</tr>
<tr>
<td>12</td>
<td>Direct Messages</td>
<td>125</td>
</tr>
<tr>
<td>13</td>
<td>Lists</td>
<td>135</td>
</tr>
<tr>
<td>14</td>
<td>Favorites and User Methods</td>
<td>147</td>
</tr>
<tr>
<td>15</td>
<td>Search</td>
<td>161</td>
</tr>
<tr>
<td>16</td>
<td>Trends and GEO</td>
<td>177</td>
</tr>
<tr>
<td>17</td>
<td>Friendships, Notification, Block, and Account Methods</td>
<td>193</td>
</tr>
<tr>
<td>18</td>
<td>Twitter Documentation</td>
<td>205</td>
</tr>
<tr>
<td>19</td>
<td>Streaming API</td>
<td>219</td>
</tr>
<tr>
<td>20</td>
<td>FailWhale and the Future of the API</td>
<td>229</td>
</tr>
<tr>
<td>21</td>
<td>Getting Started in Twitter Android Application</td>
<td>241</td>
</tr>
<tr>
<td>22</td>
<td>Building Android Applications with Twitter</td>
<td>255</td>
</tr>
<tr>
<td>23</td>
<td>Getting Started with Twitter Using iOS</td>
<td>279</td>
</tr>
<tr>
<td>24</td>
<td>Building an iPhone and iPod Touch Application with Twitter</td>
<td>293</td>
</tr>
<tr>
<td></td>
<td>Index</td>
<td>319</td>
</tr>
</tbody>
</table>
# Table of Contents

**HOUR 1:** What Is Twitter? 1  
What Twitter Offers You ................................................................. 1  
A Brief History of Twitter—or Why 140 Characters? ......................... 2  
Summary ....................................................................................... 7  
Q&A ............................................................................................... 8

**HOUR 2:** Twitter Out of the Box 11  
What Twitter Offers You ................................................................. 11  
Registering Your Application .......................................................... 15  
The Twitter Client ......................................................................... 16  
Summary ....................................................................................... 18  
Q&A ............................................................................................... 18

**HOUR 3:** Key Issues to Consider When Developing Twitter Applications 21  
Types of Twitter Users .................................................................... 21  
Types of Twitter Applications .......................................................... 25  
Platform ......................................................................................... 30  
Summary ....................................................................................... 31  
Q&A ............................................................................................... 31

**HOUR 4:** Creating a Development Environment 33  
Background of LAMP Stacks ......................................................... 33  
Setting Up a Local Web Server ....................................................... 34  
Securing Your Web Server .............................................................. 38  
Development Tools ........................................................................ 41  
Summary ....................................................................................... 45  
Q&A ............................................................................................... 46

**HOUR 5:** Making Your First API Call 49  
Making a Simple Twitter API Call .................................................. 49  
Making a Call in PHP ....................................................................... 53
Contents

Summary .......................................................................................... 57
Q&A ................................................................................................ 58

HOUR 6: Building a Simple Twitter Reader ........................................ 59
Building Our First Twitter Client .......................................................... 59
Twitter HTTP Response Codes .............................................................. 65
Summary .......................................................................................... 69
Q&A ................................................................................................ 71

HOUR 7: Creating a Twitter API Framework ..................................... 73
Twitter API Parameters ........................................................................ 73
Creating an API Function for Twitter Function Calls ............................ 75
Summary .......................................................................................... 80
Q&A ................................................................................................ 80

HOUR 8: Twitter OAuth ..................................................................... 81
What Is a Class and Why Do We Want to Use It? ............................... 81
What Is OAuth? ................................................................................ 82
How to Register Your Application ...................................................... 82
Creating the OAuth Twitter Class ......................................................... 83
PHP Library for Working with Twitter's OAuth API ............................. 84
Setting Up the twitterOAuth Class ....................................................... 85
How to Add New Functions to Your Twitter Class Object ................... 90
How Our Class Deals with Twitter Connection Errors ....................... 92
Summary .......................................................................................... 93
Q&A ................................................................................................ 93

HOUR 9: Building a Simple Twitter Client, Part I ............................. 95
Expanding the Index File to Support Tabs .......................................... 95
Adding Support for Home Timeline .................................................... 97
Adding Support for Mentions ............................................................. 99
Adding Support for Direct Messages ................................................ 101
Summary ........................................................................................ 102
Q&A .............................................................................................. 102
### HOUR 10: Building a Simple Twitter Client, Part II

1. Updating and Adding New Files to Support Input Text Field: 105
2. Sending a Message to Twitter: 108
3. API Call for Direct Messages: 109
4. Sanitizing Messages: 110
5. Summary: 110
6. Q&A: 111

### HOUR 11: Expanding Our Client for More API Calls

1. Types of API Method Calls: 113
2. Adding Tabs to Our UI: 114
4. New Status API Calls: Retweeted: 119
5. Summary: 123
6. Q&A: 123

### HOUR 12: Direct Messages

1. Sending a Direct Message: 125
2. Adding Direct Message API Support: 127
3. Adding More Direct Message API Support: 131
4. The Destroy API Method: 132
5. Summary: 133
6. Q&A: 133

### HOUR 13: Lists

1. What Is a List?: 135
2. Implementing the List API into Our Application: 137
3. Three Types of List Methods: 142
4. Summary: 144
5. Q&A: 144

### HOUR 14: Favorites and User Methods

1. Favorites API Methods: 147
2. User API Methods: 153
3. Summary: 158
4. Q&A: 159
### HOUR 15: Search

- History of Twitter Search API .............................................................. 161
- Twitter’s Stance on Search ................................................................. 161
- The Lone Search API ........................................................................ 162
- A Quick Guide to More Information on Search from the Twitter Docs ... 170
- Summary ........................................................................................ 173
- Q&A .............................................................................................. 174

### HOUR 16: Trends and GEO

- What Is a Trending Topic? .................................................................. 177
- Supporting Trends in Our Application .................................................. 177
- Understanding the GEO Tag ............................................................... 187
- Summary ........................................................................................ 190
- Q&A .............................................................................................. 190

### HOUR 17: Friendships, Notification, Block, and Account Methods

- Friendships Methods .......................................................................... 193
- Notification Methods ........................................................................ 197
- Block Methods .................................................................................. 198
- Account Methods .............................................................................. 199
- Summary ........................................................................................ 202
- Q&A .............................................................................................. 202

### HOUR 18: Twitter Documentation

- The Twitter Dev Website .................................................................... 205
- Dev.twitter.com/doc .......................................................................... 211
- Twitter Resource Page Overview ........................................................ 212
- Summary ........................................................................................ 216
- Q&A .............................................................................................. 216

### HOUR 19: Streaming API

- The Three Types of Streaming APIs ................................................... 219
- Streaming Methods .......................................................................... 222
- Summary ........................................................................................ 226
- Q&A .............................................................................................. 226
HOUR 20: FailWhale and the Future of the API ........................................ 229
   What Is Spotting the FailWhale? .................................................. 229
   Review of the Application We Just Built .................................... 231
   Where Is the Twitter API Going? .................................................. 236
   Summary ...................................................................................... 237
   Q&A ............................................................................................. 238

HOUR 21: Getting Started in Twitter Android Application .................. 241
   Introducing Android ....................................................................... 241
   Creating the Hello Android Project ............................................. 243
   Summary ...................................................................................... 251
   Q&A ............................................................................................. 252

HOUR 22: Building Android Applications with Twitter ..................... 255
   Using Twitter OAuth in Android .................................................. 255
   Importing Packages ...................................................................... 261
   Summary ...................................................................................... 276
   Q&A ............................................................................................. 276

HOUR 23: Getting Started with Twitter Using iOS .............................. 279
   Introducing iOS ............................................................................. 279
   Creating a Hello World Application ............................................. 280
   Summary ...................................................................................... 289
   Q&A ............................................................................................. 290

HOUR 24: Building an iPhone and iPod Touch Application with Twitter 293
   Introducing Twitter xAuth ............................................................. 293
   Benefits of Using Twitter xAuth .................................................. 294
   Selecting Twitter Objective-C Libraries ...................................... 294
   Loading xAuth Token .................................................................... 302
   Posting Tweet ............................................................................... 304
   Adding MGTwitterEngine Delegate Methods .............................. 305
   Creating Objects in Interface Builder ......................................... 308
   Summary ...................................................................................... 315
   Q&A ............................................................................................. 316

INDEX ............................................................................................. 319
About the Author

Dr. Christopher Peri received his Doctorate from the University of California, Berkeley, in Architecture. His focus was on Collaboration in Virtual Environments delving into methods that facilitate designers and engineers to improve communication over remote networks.

He started playing with the Twitter API very early in the API release, creating his own Twitter client called TwittFilter, which is geared more to the occasional user than someone who uses Twitter all the time. As time went on, he added more and more features and functions for his own personal use, until one day he realized he had a fairly sophisticated application and opened it up to the general public to use. He learned quite a bit about the Twitter API the hard way—by simply coding things up and seeing what happens. Although TwittFilter is still a personal project, he has already created a number of private Twitter applications, robots, and smaller projects like NewsSnacker.com, which is open to the public.
About the Contributing Author

Bess Ho is a UI Engineer in mobile, tablet, TV, and web with a strong background in data analytic and consumer behavior. She received her Master Degree from the University of California, Davis in Food Science and Technology. Her focus was on Consumer Sensory Science and Engineering. She is the winner of Nokia Open Screen Project Fund and was elected as Samsung Star in the Samsung Mobile Innovator worldwide program. She served as technical editor for the book titled Building OpenSocial Apps: A Field Guide to Working with MySpace Platform (Addison Wesley, 2009). She has presented mobile technology at Stanford University, O’Reilly Web20 Expo SF, Where20 Conference, Silicon Valley China Wireless Conference, and many developer events. Currently, she is Mobile Architect (EIR) for Archimedes Ventures. She also advises many early-stage startups in UI/UXP design and mobile development in multiple platforms. She is actively teaching many mobile classes such as iOS SDK in Silicon Valley and online courses at Udemy.com. You can follow her at Twitter @Bess or Slideshare at www.slideshare.net/bess.ho. Her developer blog is at http://www.bess.co.
Acknowledgments

Christopher Peri—We would like to thank all the unknown coders on the interwebs who have contributed to not only Twitter’s success, but creating mountains of technical information and code examples that allows a lowly hobby programmer, like myself, to learn how to work with Twitter API and one day...write a book on it. A number of people have helped with this book, but I want to call out three people specifically: @chiah for creating the foundation of Hour 1, @jon_wu for Hour 8 as well as helping with debugging and general feedback on technical issues, and @LanceNanek for debugging and researching Android in Hour 22.
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

You can email or write me directly to let me know what you did or didn’t like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book’s title and author as well as your name and phone or email address. I will carefully review your comments and share them with the author and editors who worked on the book.

Email: opensource@samspublishing.com

Mail: Mark Taub
Associate Publisher
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at www.informit.com/title/9780672331107 for convenient access to any updates, downloads, or errata that might be available for this book.
Preface

This book on the Twitter API is geared to the programmer who is just a bit past beginner—who knows the basics of LAMP, including how to set up a basic server, PHP, JavaScript, HTML, and CSS. You do not have to be an expert programmer to use this book, but you should know how to look things up. In writing this book, we have tried to provide you with everything you need to get a simple Twitter client up and running. We include an hour on setting up your environment, as well as providing you with HTML and CSS codes to have something up and running. However, it's beyond the scope of this book to explain what is happening with these codes. Instead, we focus on the code surrounding the API calls, OAuth, and the returns. That does not mean that you could not use this book if you are a beginner programmer. Because we provide you with all the code and build an application up step by step, you can stop at any time and look up parts of the code you do not understand. However, if you have never coded anything before, you may find that this book moves far too fast. It may be better to get an introductory book on basic programming in PHP before reading this book.

In writing this book, we also kept in mind experienced programmers who have been asked to create a Twitter application or include Twitter support in a current application, even if they do not know much about Twitter. We believe it's important to understand what Twitter is, how it's being used, and what makes it different from other social media services. It's with this understanding that you will be able to approach your Twitter project with a more engaged understanding of what your application is trying to accomplish, which is the best way to not only satisfy product requirements, but also design future growth.

*Sams Teach Yourself Twitter API in 24 Hours* is a little different from most technical books in that the book is geared around creating a functional Twitter client, including all HTML, CSS, JavaScript, and PHP needed to create your own application. We also dedicated the last four hours of this book to getting you started with making API calls on the iPhone and Android OSes in case you want to make your own mobile Twitter application.

Unlike most books, this book was written as Twitter and the API set was going through major changes. As such, the book and the code used in the book have been edited many, many times. So much so that we expect there will be a technical oversight here and there. So be sure to check the book's website for changes and updates.
Teach Yourself the Twitter API in 24 Hours

(http://www.twitterapi24.com/). Also, as much as we tried to keep up with all the changes happening with Twitter, we fully expect some details about the various API's to evolve from the time of the last edit to the time you have this book in your hands.

We hope you enjoy this book.
Key Issues to Consider When Developing Twitter Applications

What You’ll Learn in This Hour:

- Different types of Twitter users and how they impact code design
- Different types of Twitter applications and program architecture
- Things to consider if you are not building a web-based application

Types of Twitter Users

As one would expect with an API system as open as Twitter, and the explosion of interesting applications people have developed, we have also seen the development of different types of Twitter users. Understanding these types of users and knowing which of them we are trying to reach will inform how we may want to build our Twitter application framework. As with any large user base, there are a number of ways to set up categories. In this hour, we will break down and discuss the users in the following categories or types.

The News Reader

Twitter is a great source of breaking news, whether it’s politics, business, sports, or following celebrities. Most users use searches to find what they are interested in, or they follow Twitter feeds that act like RSS readers. For example, BreakingNews is what you would guess it would be—a Twitter account publishing breaking news. Most news outlets have such accounts: CBSNews, ABC, BBC, and so on. The screenshot of NewsSnacker, an application created by the author (shown in Figure 3.1) is a good example of a Twitter application that focuses on the news.
Although making search and Twitter account API reads from Twitter does not require authentication, you can still get dinged going over the API limit because Twitter will limit calls from an IP address. So, you still need to keep in mind how often you make calls. In the case of NewsSnacker, we use a white-listed account because the user could exceed the API calls-per-hour limit since each news service is a separate call. Suppose that the user has 10 sources and refreshes every 30 minutes. That is 200 calls in an hour, which is over the current limit of 150 for non logged in users. This does not include normal calls to check for new mentions or direct messages from the user’s chosen Twitter client application. An alternative approach is to create a list of twitter news accounts and then call that list. However since NewsSnacker removes duplicate posts, a large number of returns on the list call would be required. Both approaches have their merits however; one feature of NewsSnacker is to allow a custom list of sources. This can be done by having the user log into the application and then select which of their lists they would like to call thus the second approach is being pursued in the next version of the application.

**Chatters**

Twitter does allow for people to have conversations; it’s called a *direct message*. However, many people like to hold their conversations in public and a big attraction for these people is conversation threading. This is a very complicated proposition, so much so that new APIs are being created to deal with this situation. We will cover
Types of Twitter Users

Types of Twitter Users

retweeting in later hours, but this could cause quite an impact on your code’s structure because of older reply techniques that use the letters RT for conversations instead of recent API methods that support replies formally. So, supporting Twitter conversation is a decision you will want to make early in your product’s design.

Power Users and PR Managers

Power Users and PR Managers

Although you will have a drag-out fight between the two because one is personal messaging and the other is more professional, the impact on product design is not that much different. PR (public relations) managers, power users, and anyone who consumes or monitors a lot of Twitter information will put special requirements on you as a product developer. Like the limits to the number of API calls mentioned in the section discussing the user group news readers, the issues with the power users and PR managers group will be the same, with the added requirements of being able to sort and search the stream of messages that come in. They may also need to send messages on a schedule or from people using the same account. There is usually no simple way around this issue other than to start thinking of a well laid-out database up front. You may want to also explore having your server make the API calls and relay the information to your Twitter application in the form of automatic processes or bots. Furthermore, set up your architecture to deal with a wide variety of API calls. We will cover this later in the hour. PR managers will want more than just searching the Twitter stream; they will want to make sense of it and make sense of who is on that stream and their influence. The API has just expanded to handle retweets, but not all Twitter clients will be updated to work with this API. As such, you still need to pay attention to RT (the current convention for a retweet) and hashtags. Plan for this up front. Also, plan to keep some of the user information in your database; you will want to use it for user profile and relationship analysis. Although the number of power users, compared to typical Twitter users, is quite low, having a power user using (and advocating) your application is highly desirable, and although every power user you talk to will have a different list of features and functions, there are some things you must be able to support—for example, dynamic search. Just providing a call and return to the search API is not good enough anymore. The current and future power users of Twitter are going to demand just as much power and feedback as they get using Google search. For example, power users would want links with the tweets that are returned to be followed and analyzed in some manner. Perhaps you should show a thumbnail of the site, or display the title and the first 50 words of the link. Be sensitive to nonstandard protocols, such as searching stock quotes using the $ sign in front of the stock market ID. For example, $aapl for Apple. Power users are going to demand speed and customization and will fully expect that your application understand the nonstandard features (social conventions) of Twitter.
Microbloggers

Microbloggers will want to take the time to craft each tweet carefully. Pay attention to the ease of creating a message—that is, allowing them to save as drafts, sending to multiple Twitter accounts, spell checking (yes, spell checking), and although this is not easy, a quick look up of the other tweeters or access to a list of tweeters. Especially for PR users, you may want to have a look at simple web-based CRM products to give you ideas. A new API to Twitter is the capability to store lists of tweeters. This is useful to all power users as well as microbloggers.

High-Frequency Users (TwitterHolics)

The current rules of the API system allow only 350 calls per hour if you are logged in, 150 if not. This may seem like a lot, but based on what features you are providing to your users, this can go very quickly. It’s not unlikely that you could have five API calls per user action if you need to make follow-up calls. If they are high-frequency users, they may find themselves approaching the 350-call limit pretty quickly. Although there are calls that do not require credentials, you could still run up against this limit because Twitter does count the number of calls from an IP. As such, be sure you monitor the number of calls the user has left and deal with it accordingly. The good news is that an API call exists for checking how many API calls the user has left which does not count against your API limit. However, calling it over and over again too often (every 5 seconds, for example) could trigger other traffic limit controls.

New Users

This is less an API architecture question than a GUI issue. Although GUI design is not addressed directly in this book, consider using clear terms and common metaphors (like an email system, for example) for the layout and functionality of your application. Do not assume that your users will understand various social conventions in Twitter, so explain it up front and design your functions’ intent clearly using tool tips for icons for example. If you are making an application that reflects some aspects of the Twitter.com site, be sure to follow the conventions Twitter uses.

Bots

Bots (programs that perform automated tasks), including creating spam or setting up phishing attacks, will always be an issue. A sophisticated Twitter application will be aware of some of these bots and try to protect users. You may, however, need to
create your own bots (for good, not evil). For example, you might take a RSS feed and republish it to Twitter after passing it through a business rules filter which is something the main Author of this book does. Because a bot is nothing more than “rules” you have for dealing with reading or creating Twitter messages or lists, you will find creating automated processes very easy with the Twitter API.

**Types of Twitter Applications**

Normally, when I’m about to start writing a Twitter application, I already know what I want it to do. Thus, based on the features and functions I have in mind, I already know what platform and category of users I’m targeting. Because we cannot know what you, the reader, have in mind, we will try to set up a basic framework for thinking about the various things you can do with Twitter as we go through this book. Part of Twitter’s success is its simplicity and wide-open API. As such, people have developed powerful, sophisticated applications, mashups, and simple widgets that run in other apps or on web pages. However, the approach you will take building a full-on application is different from building a simple mashup or widget.

A mashup is a web page or application that takes two or more data sources and combines them into a new service. Typically, mashups create a functionality not envisioned by the creators of the original sources. Twitter is a very popular mashup source.

Building a feature-rich Twitter application takes some planning. Although we will walk you through various examples of how to build apps around specific APIs, we want to bring focus, too. There is an overall approach you should determine before you write line one.

**Widget**

Let’s talk about architecture around a simple widget. Suppose our simple widget is going to display the results of a search or the latest tweets from a user. This is the easiest to build. All we have to think about is four steps: make an API call to Twitter, parse the return, format it, and display it. That’s it. We diagrammed this simple architecture in Figure 3.2.
All API systems work this way, but what’s great about Twitter is that the results are already of value. Quite often, blogging sites (mostly personal) have this type of widget. I have a widget like this on my blog (see Figure 3.3).

**Mashup**

Because a mashup can be the combination of anything, and that’s kind of the point of mashups, we are going to think about our architecture a bit differently. Although technically, a mashup can be just two sources of information or very complex number and relationship of sources, we are going to stick with the spirit of what is considered a mashup by just thinking about mixing two data sources. For example, we can take our Twitter search feed and weather data and display tweets from places that are raining versus tweets from where it’s sunny. In this case, we need to store our returns from Twitter somewhere while we get weather data. Then we need to perform some business logic on those returns.
Business logic is a nontechnical term generally used to describe the functional algorithms that handle information exchange between a database and a user interface. It is distinguished from input/output data validation and product logic. From Wikipedia, the free encyclopedia

In this case, we need to hold our returns in an array so that when we get the weather data, we can reorganize our data. Because tweets are small, discrete messages, it makes sense to create a multidimensional array object that we can easily explore. So now, we will add one more layer to our diagram. As you can see in Figure 3.4, we are using arrays to store our parsed return so that we can apply some rules (business logic) to create a more valuable dataset.

Twitter Application

I would expect that only a small percentage of readers of this book are intending to build a full-featured Twitter client, but if you are, you want to approach building your application like any other application. Think about your calls to Twitter almost like calls to a database where you provide a set of parameters with your call and get a filtered response that can then be analyzed or applied to a set of rules. It is also well worth your time to set up your Twitter calls in a separate class to deal with errors and changes to the API. You should also set up another class to deal with converting your Twitter calls into multidimensional arrays and/or storing them in a database. The reason for this is that Twitter is still changing. Even during the writing of this book, we had to make adjustments to the book’s index as new methods were introduced and other calls were deprecated. By keeping these two processes in standalone classes, you’re going to save yourself some headaches down the road. If you are
planning on building a full-scale Twitter app, we recommend bookmarking the website for this book and the Twitter API site. Really! It changes and grows that much.

Also, somewhat like a database, you can store information in Twitter. For example, a much-overlooked feature is favorites. This API call allows you to save tweets. This can be quite useful as a means of understanding which tweeters and types of tweets a user tends to favor. New to the API list is lists. This is a list of tweeters a user creates. Again, it’s a powerful bit of information that can be quite useful in understanding users’ preferences. What is more interesting, though, is using these two API calls as storage devices if your user is under your control—a corporate account, for example. Because the user of that account does not interact with the account personally, you can use these API calls to store tweets and lists that can have greater meaning than originally intended. For example, suppose you have a corporate account for company X. We can store in the favorite list all tweets that match a certain rule, like any tweet that has an unfavorable term in the tweet. Now you have a list of tweets that public relations can examine using other than the application you developed. Also, remember you have access to user bio, location, and other elements. Again, because of how open Twitter is with its API, you can use these fields for anything—for example, including the updating of the Twitter background based on the latest message from the company, or perhaps updating the location field if you’re a mobile food van, or changing the profile image based on the time of day or your mood. Instant database functionality ... of sorts! Now this does not mean you should not have a database if you intended on storing anything beyond the simple examples provided here. Also, it is not recommended to abuse this open access by placing unrelated data in these fields. Most applications will follow a simple structure, as illustrated in Figure 3.5.
Pure Chat
This class of Twitter application is concerned with creating tweets, reading incoming tweets, searching Twitter, retweeting, setting/getting favorites, and displaying simple user account information. Everything can be done as a standalone command, meaning you do not need to store information outside of Twitter. Each command has only one or two API calls. The current Twitter.com main web page is this type of application. Since we do not need to keep track of a state or store data, we can create this application using nothing more than a simple collection of PHP calls. For this class of application, we want to think about our application as a series of standalone pages. It would be a good idea to use cookies on the user’s computer in case you need to store last-seen dates or other simple pieces of information.

Structured Display
Very common with Twitter applications are the capabilities to save groups, perform more advanced searches, display only new information and some threaded conversations, and so on. Although some of these structured displays can be somewhat complex, the approach you would take as a programmer is not that much different. Many of these structured displays can be achieved without storing information on the server but by using API calls and cookies instead. Consider the following example: Suppose we want to display a column of unread tweets, tweets from our “top 10” friends, three or four saved searches, and your current favorites lists. All of these can be achieved by passing variables within the existing API calls. You will actually work far harder at the UI than the backend coding. For this class of application, we want to set up our code as a series of calls that are more or less self-contained. This will make dealing with the GUI less troublesome as redesign is requested or required.

Twitter Statistics
Collecting statistics from Twitter data provides great promise for research, improved discover and communications. However, this class of Twitter application is a bit harder. TwittFilter, another application created by this author is in this class as shown in Figure 3.6.

This class of Twitter application depends on creating new information through analyzing the return or returns from past Twitter calls, and storing or modifying this information on the server, typically in a database. This, however, is where we find Twitter to be the most interesting; because Twitter has such a large user base, you can gather enough data to infer information that does not have a direct user correlation. Did someone say mashup? For example, a very popular and now API-supported feature called “Trends” in Twitter is nothing more than a constant search
Because we need to store information as well as grab details for analysis, we need to think about how we structure our program differently. For this class of application, we want to think of Twitter as more of a database source. Setting up our arrays that allow for ease of use within formulas, as well as pulling and pushing into databases, will be a great benefit as our analytics become more and more complex. However, if you are not white listed, you will run into the API call limit quite quickly. It’s recommended that if you plan to do statics that require large sample sets or recursive calls, that you explore the streaming API.

Platform

Now that we know the class of application we want to develop, we need to think about the delivery platform. If you are going to develop for UI-hosted apps, such as native mobile apps or Adobe Air, you may again want to modify how to approach your coding. Typically, when creating an app for the iPhone or other mobile platform, many of our UI elements are going to be handled on the device. You also will want to minimize the amount of traffic going back and forth as much as possible. Therefore, you should design your application around the output, which could be XML, JSON, or some custom bitcode. In this case, the organizational structure you choose will dictate how you structure your backend code. Because we have the luxury of storing information on our target platform, we can focus on speed and ease of architecture. Even though our backend code is not responsible for the presentation...
layer (display), you still need to follow the basic tenants of good programming design by keeping the business logic separate from the API calls; don’t fall into the trap of making each call from your application as a separate instance, as you may with the Pure Chat approach. You never know when your application starts to take on more features than you planned.

**Summary**

In this hour, you were introduced to various types of Twitter users. Depending on your product’s target market, you may need to think about how you will approach the design architecture of your product.

This was not intended to be an exhaustive list, nor an absolute one. One could easily break this list into smaller pieces or roll it up into more general categories; instead, it’s to provide a framework to think about the application you intend to build. We broke this up into two sections because we want to make a distinction between the type of use and type of application, however, do not think you can explore one without the other. When designing any application, you should always start with the user. What is the value proposition you are offering users in order for them to use your application? Once you understand that, you can then move to the type of application you want to create. So, we started out with an exploration of types of Twitter users, and then types of Twitter applications. We ended this hour with a short conversation about platforms. If you are developing for anything other than the desktop, you are most likely already aware of these points, but we included them for less-experienced developers as good to know.

Now, hold on to your hats because in the following hours, we are going to start building code!

**Q&A**

**Q.** Should I apply for a white-list account before I start coding?

**A.** No. White listed accounts are currently not available. However, you will find that having 350 calls per hour is plenty as you learn how to develop your program.

**Q.** I plan to make a simple Twitter application now, but I may expand it later. Should I bother setting up a separate twitterAPI class?

**A.** Yes. If you have any plans, even just thoughts of doing something beyond a few different types of API calls, set up a separate class for your API calls. In addition to new APIs, current API calls can change.
Workshop

Quiz

1. What is meant by thinking about Twitter as a type of database?

2. I check my Twitter account only a few days a week on my iPhone. What kind of Twitter user am I?

3. Is it illegal to create bots?

4. What is the easiest type of Twitter application to create?

Quiz Answers

1. This is a two-part answer: 1) Although Twitter exposes everything, you still can only get detail data on users one at a time although this is changing. Thus, thinking about accessing user statistics as if you were accessing a database is a useful way to think about what you can do with Twitter. 2) If you have control over the Twitter account(s), you can use the fields in Twitter to store information instead of on your database.

2. You are a news reader. Even if you are reading only your timeline (people you follow), you are more of a consumer of information than a creator.

3. No—and not all bots are bad. However, the good folks at Twitter do actively look for automated processes that abuse the system.

4. A pure chat widget.

Exercises

1. Describe your typical target user and then determine the class of application you feel is appropriate for your user.

2. If you plan to create an automated process, write down each step and then count the number of times you will need to call Twitter to get information. What happens if the user hits refresh 10 times in 10 minutes? Will you go over the 150-API call limit if they are not logged in?
Index

Symbols

# (hashtag), 2, 8
#pragma mark, 301
$ (dollar sign), 23
@ (at symbol), 258
+ (plus symbol), 258
200 OK response code, 242
304 Not Modified response code, 230
400 Bad Request response code, 230
401 Unauthorized response code, 230
403 Forbidden response code, 230
404 Not Found response code, 230
406 Not Acceptable response code, 230
420 Enhance Your Calm response code, 230, 235
500 Internal Server Error response code, 230
502 Bad Gateway response code, 230
503 Service Unavailable response code, 230

A

accessing other user information, 155
accessTokenReceived: method, 305-308
account methods
  account/end_session, 201
  account/update_profile, 201
  account/update_profile_background_image, 201
  account/update_profile_colors, 201
  account/update_profile_image, 201
adding to applications
  header.inc, 199-200
  parseTwitter.php, 200
  twitteroauth.php, 200
list of, 199
320

account/end_session method

account/end_session method, 201
accounts
  private accounts, 2
  rate limiting, 11, 18
  setting up, 12-15
  whitelisting, 11-12, 31
account/update_profile method, 201
account/update_profile_background_image method, 201
account/update_profile_colors method, 201
account/update_profile_image method, 201
active SDK, targeting, 285
ADT (Android Development Tools) plug-in, 242, 252
Android applications, 241-242
  ADT (Android Development Tools) plug-in, 242, 252
  Android OAuth application, 255-261
    creating, 256
    intent filters and permission, 259-260
    layout, 257-259
    Twitter Java libraries, 256
    XML resources, 261
AVD (Android Virtual Device), creating, 242
development environments, 252
Hello Android project
  AndroidManifest.xml, 248-249
  AVD (Android Virtual Device), 250
  creating, 243-244
  helloandroid.java, 245-246
  launching, 246-248
  SDK issues, 250-251
  supported API levels, 249-250
  views, 246
  importing packages, 261-275
    adding OAuth, 262-264
    authenticating application, 264-266
    responding after authentication, 266-269
    TwitterOAuth.java, 269-275
  OAuth-signpost, 251
  supported operating systems, 252
  Twitter4J library, 251
  xAuth, 275-276
Android Development Tools (ADT) plug-in, 242, 252
Android OAuth application
  importing packages
    adding OAuth, 262-264
    authenticating application, 264-266
    responding after authentication, 266-269
    TwitterOAuth.java, 269-275
  Twitter Java libraries, 256
Android SDK, downloading, 241
Android Virtual Device (AVD), 242, 250
  android:layout_marginBottom attribute, 258
  android:layout_width attribute, 258
  AndroidManifest.xml, 248-249
  @Anywhere and Tweet Button (dev.twitter.com), 215
Apache, 34
API calls
  account methods, 199
  block methods, 198
    blocks/blocking/ids, 198
    blocks/create, 198
    blocks/destroy, 198
    blocks/exists, 198
    blocks/unblocking, 198
  catching API requests, 232-233
  creating
    cURL, 53-55, 58
    in PHP, 53-57
    for Twitter function calls, 75-78
  user_timeline API, 55-57
    in XML, 49-52
DELETE calls, 136-137
  destroy, 132-133
  direct message methods, 109-110
  favorites API methods.
    See favorites
  Friendships methods
    explained, 193
    supporting in applications, 194-197
  geo/search, 187-190
GET calls, 136-137
list calls, 135-136
  List Members resources, 143
  List resources, 142
  List Subscribers resources, 143-144
How can we make this index more useful? Email us at indexes@samspublishing.com
call_trends_daily() function

call_trends_daily() function, 183

call_users() function, 157

callback attribute (search), 162, 171

callPage() function, 107-108

calls (API). See API calls

callTwitter() function, 63-64, 75

capabilities of Twitter, 1-2

catching API requests, 232-233

character limit for tweets, 8

chatters, 22-23

choosing passwords, 12-13

Chrome, 42

classes

advantages of, 82

class files, storing, 93

explained, 81-82

OAuth

adding functions to, 90-92

creating, 83

twitterOAuth

creating, 85-87

getUserTimeline() function, 90-92

oauth_index.php, 87-88

Twitter connection errors, handling, 92-93

twitteroauth.php, 88-92

when to use, 93

clients (Twitter), creating, 16-18, 27-28, 59-60

Android applications, 241-242

ADT (Android Development Tools) plug-in, 242

Android OAuth application, 255-261

AVD (Android Virtual Device), 242

development environments, 252

Hello Android project, 243-251

importing packages, 261-275

supported operating systems, 252

xAuth, 275-276

API calls. See API calls

application architecture diagram, 231-232

block methods, 198

blocks, 198

catching API requests, 232-233

direct messages

adding API support for, 101-102, 109-110, 127-130, 131-132

call_direct() function, 101

callPage() function, 107-108

deleting, 132-133

destroy API call, 132-133

direct() function, 128

friendshipExists() function, 129-130

getMessages() function, 101, 131

renderTweets() function, 128

sanitizing, 110

sendMessage() function, 107-108

testing, 126-127

UI elements, adding, 125-126

favorites

adding to applications, 148-149

createFavorite() function, 150

creating, 149

definition of, 147

destroyFavorites() function, 152

destroying, 152

favorite() function, 149

showFavorites() function, 148

Friendships methods

explained, 193

supporting in applications, 194-197

home timeline, 97-99

HTTP response codes, 65-66

index.php, 60-61, 69, 95-96

input text fields

base.js, 107-108, 110

createMessage.php, 106

index.php, 105-106

main.css, 106-107

sendMessage.php, 108

iOS. See iOS

lists

API support for, 135-136

creating, 137-141

definition of, 135

List Members resources, 143

List resources, 142

List Subscribers resources, 143-144

main.css, 61-63
development environments

Count parameter, 74
count parameter (getMentions() function), 99
create_message.php, 168-169
createFavorite() function, 150
createList() function, 139
createListItem() function, 138
createMessage.php, 106
cURL, 53-55, 58
$curl_handle variable, 53, 64
curl_setopt() function, 92
currentTimeMillis() method, 267
customizing Tweet button, 210-211

daily trends, 180-185
call_trends_daily() function, 183
header.inc, 183
parseTwitter.php, 183-184
showTrends_daily() function, 184
twitteroauth.php, 184-185
declaring
properties/methods, 300
variables, 300
DELETE calls, 136-137
deleting direct messages, 132-133
destroy API call, 132-133
destroyFavorites() function, 152
destroying favorites, 152
development environments
for Android, 252
LAMP stacks
explained, 33-34

main.php, 63, 69-70
mentions
adding support for, 99-101
call_timeline() function, 101
getMentions() function, 99-101
notifications
disabling, 197
enabling, 197
parseTwitter.php, 66-67
render.php, 67-71
retrying if Twitter is down, 233-236
Retweet button
base.js, 122
id/retweeted_by API call, 122
id/retweeted_by/ids API call, 123
main.css, 122
parseTwitter.php, 121
render.php, 121
retweets/id API call, 122
sendMessages.php, 120
twitteroauth.php, 120
Search. See Search
streaming, 219
advantages of, 226
limits on, 221
pre-launch checklist, 221-222
site streams, 220
Streaming API, 219
streaming methods, 222-226
user streams, 219-220
when to use, 220
tabs. See tabs
trending topics
call_trends() function, 178-179
definition of, 166-167
recent, daily, and weekly trends, 180-185
showTrends() function, 178-179
supporting in applications, 177-180
Tweet button, 209
adding with iframe, 210
adding with JavaScript, 210
customizing, 210-211
Twitter clients, creating
ADT (Android Development Tools) plug-in, 252
getMessagesSent() function, 132
twitterAPI.php, 64-65, 70
colors (profile), 201-203
commandLine.php
favorites support, 150
Friendships methods support, 195
list support, 138-139
commands, $_GET, 95
config.php, 85
configuring
accounts, 12-15
local web servers, 34-38
connection errors, handling, 92-93
console page (dev.twitter.com), 207
Consumer key, retrieving, 284-285
Consumer secret, retrieving, 284-285

How can we make this index more useful? Email us at indexes@samspublishing.com
development environments

popularity of, 46
XAMPP, 35-38
local web servers, configuring, 34-38
tools
Chrome, 42
Firebug, 41-42
Firefox, 41
IDEs (integrated development environments), 43-44
Internet Explorer, 42
phpMyAdmin, 42
recommended toolbox, 45
revision control systems, 44-45
text editors, 43
web server security, 38-41
MySQL, 40-41
phpMyAdmin, 41
XAMPP pages, 40
XAMPP security console, 39-40
deviation tools
Chrome, 42
Firebug, 41-42
Firefox, 41
IDEs (integrated development environments), 43-44
Internet Explorer, 42
phpMyAdmin, 42
recommended toolbox, 45
revision control systems, 44-45
text editors, 43
dev.twitter.com website, 211
@Anywhere and Tweet Button, 212
Authentication, 212
console page, 207
Ecosystem, 216
Guidelines and Terms, 213
REST API and General, 214
start page, 206
Streaming API Documentation and Search API, 215
direct() function, 128
direct messages, 125
adding API support for,
101-102, 109-110
base.js, 128
header.inc, 132
parseTwitter.php, 131-132
render.php, 127-128
sendMessage.php, 129
twitteroauth.php, 129-132
call_direct() function, 101
callPage() function, 107-108
deleting, 132-133
destroy API call, 132-133
direct() function, 128
friendshipExists() function,
129-130
getMessages() function,
101, 131
getMessagesSent() function,
132
input text fields. See input
text fields
renderTweets() function, 128
sanitizing, 110
sending message to Twitter,
108-109
sendMessage() function,
107-108
testing, 126-127
UI elements, adding
header.inc, 125-126
index.php, 125
direct messages (DMs), 2
directives
#pragma mark, 301
@interface, 300
disabling notifications, 197
Display Guidelines (dev.twitter.com), 213
DMs (direct messages), 2
documentation
apiwiki.twitter.com, 211
dev.twitter.com website, 211
@Anywhere and Tweet Button, 212
Authentication, 212
console page, 207
Ecosystem, 216
Guidelines and Terms, 213
REST API and General, 214
start page, 206
Streaming API Documentation and Search API, 215
dollar sign ($), 23
Dorsey, Jack, 2-3
downloading
Android SDK, 241
MGTwitterEngine library, 285
MGTwitterEngineDemo, 285
Oauth-signpost, 251
Twitter4J library, 251
Eclipse, 43-44, 241
Ecosystem section (dev.twitter.com), 216
editing
  index.php, 105-106
twitteroauth.php, 90-92
editors (text), 43
Egyptian revolution, tweets sent during, 5-7
enabling notifications, 197
ending sessions, 201
errors, Twitter connection errors, 92-93
exceptions
  definition of, 92
  Twitter connection errors, handling, 92

F
Facebook, compared to Twitter, 4
FailWhale, 229-230
favorite() function, 149
favorites
  adding to applications, 148
    base.js, 149
    header.inc, 148
    parseTwitter.php, 148
    twitteroauth.php, 148
  creating, 149
    base.js, 152
    commandLine.php, 150
    render.php, 149-152
twitteroauth.php, 150, 152
  definition of, 147
  destroyFavorites() function, 152
  destroying, 152
  favorite() function, 149
  showFavorites() function, 148
fields, input text fields
  base.js, 107-110
  createMessage.php, 106
  index.php, 105-106
  main.css, 106-107
  sendMessage.php, 108
files. See also specific files
  class files, storing, 93
  organizing, 72
Firebug, 41-42
Firefox, 41
follow() function, 195
followers, 2
following, 2
friendshipExists() function, 129-130, 141
Friendships methods
  explained, 193
  supporting in applications
    base.js, 195
    commandLine.php, 195
    parseTwitter.php, 196-197
    render.php, 194
    twitteroauth.php, 195
functions. See also API calls
  adding to twitterOAuth class, 90-92
    askOAuth(), 264-266
    call_timeline(), 101
call_direct(), 101, 140
call_search(), 167
call_showList(), 140
call_timeline(), 97
call_trends(), 178
call_trends_daily(), 183
call_users(), 157
callPage(), 107-108
callTwitter(), 63-64, 75
createList(), 139
createListItem(), 138
currentTimeMillis(), 267
direct(), 128
defavorite(), 149
follow(), 195
friendshipExists(), 129-130, 141
dataGet(), 266
dataHomeTimeline(), 89
dataMentions(), 233
code listing, 100
  parameters, 99-100
dataMessages(), 101, 131
dataMessagesSent(), 132
dataPublicTimeline(), 78-79
dataQueryParameter(), 266
dataRTByMe(). See also API calls
dataRTOfMe(). See also API calls
dataRTToMe(). See also API calls
dataTwitterData(), 97
dataUserRate(), 200
dataUserTimeline(), 75-78, 90-92

How can we make this index more useful? Email us at indexes@samspublishing.com
functions

htmlentities(), 54
leave(), 195
makeText(), 265, 267
OAuthRequest(), 88
onCreate(), 261
onNewIntent(), 266
parseTwitter(), 70
parseTwitterReply(), 63
parseTwitterReply(), 66
phpinfo(), 40
postMessage(), 304-305
renderLists(), 141
renderTweets(), 63, 67, 128
responseFromServer(), 108
search(), 167-168
sendMessage(), 107-108
sendSearch(), 169
setContentView(), 261
setText(), 267
setVisibility(), 268
showFollowers(), 158
showFriends(), 158
showgeo_search(), 232
showLists(), 141
showTrends(), 178-179
showTrends_current(), 184
showTrends_daily(), 184
showTrends_weekly(), 184
showUser(), 154
SimpleXMLElement(), 97
toLocaleString(), 267
updateStatus(), 108, 267

G

Geo Developer Guidelines
(dev.twitter.com), 213
GEO tag, 187-190
gcode attribute (search), 163, 165, 172
GET/search API call, 187-190
GET calls, 136-137
$GET command, 95
Get xAuth Access Token button, 288
g_public_timeline.php, 53
getData() method, 266
gHomeTimeline() function, 89
gMentions() function, 233
code listing, 100
parameters, 99-100
gMessages() function, 101, 131
gMessagesSent() function, 132
gPublicTimeline() function,
78-79
gQueryParameter() method, 266
gRTByMe() function, 119
gRTOfMe() function. See also
API calls
gRTToMe() function. See also
API calls
gTwitterData() function, 97
getUserRate() function, 200
gUsersTimeline() function,
90-92
gUserTimeline() function, 75-78
getXAuthAccessTokenForUsername:
password: method, 304
Git, 44
Global System for Mobile
Communications (GSM), 2-3

Google Chrome, 42
GSM (Global System for Mobile
Communications), 2-3
Guidelines and Terms section
(dev.twitter.com), 213

H

hashtag, 2, 8
header.inc, 114-115
account methods, 199-200
direct message support,
125-126, 132
favorites support, 148
list support, 139
recent, daily, and weekly
trends, 183
search support, 166
thumbnail viewer, creating,
156-157
trending topics support, 178
user API methods, 154

Hello Android project
AndroidManifest.xml,
248-249
AVD (Android Virtual Device),
250
creating, 243-244
helloandroid.java, 245-246
launching, 246-248
Oauth-signpost, 251
SDK issues, 250-251
supported API levels,
249-250
Twitter4J library, 251
views, 246
Hello World application (iOS), 280-283
helloandroid.java, 245-246
high-frequency users, 24
history of Twitter, 2-3
home page (Twitter), 16-18
home timeline, creating, 97-99
htmlentities() function, 54
HTTP response codes
catching, 232-236
creating, 65-66
supported codes, 230
Hypertext Coffee Pot Control Protocol, 230
$i counter, 97
ID parameter, 74
IDEs (integrated development environments), 43-44
id/retweeted_by API call, 122
id/retweeted_by/ids API call, 123
iframe, adding Tweet button with, 210
images, profile images, 201
importing libraries
to header files, 299-300
to implementation files, 301
packages, 261-275
adding OAuth, 262-264
authenticating application, 264-266
responding after authentication, 266-269
TwitterOAuth.java, 269-275
include_entities parameter (getMentions() function), 100
include_rtf parameter (getMentions() function), 100
index.php, 95-96, 105-106
creating, 60-61, 69
direct message support, 125
expanding to support tabs, 117
initializing MGTwitterEngine library, 302
initWithCoder: method, 302
input text fields
base.js, 107-110
createMessage.php, 106
index.php, 105-106
main.css, 106-107
sendMessage.php, 108
installing XAMPP, 35-38
on Linux, 37-38
on Mac OS, 37
troubleshooting installation, 38
on Windows, 35-37
integrated development environments (IDEs), 43-44
intent filters, adding to Android OAuth application, 259-260
IntentFilter objects, 259-260
Interface Builder
connecting objects in, 309-311
creating objects in, 308-315
defining object attributes in, 309-311
@interface directive, 300
Internet Explorer, 42
iOS, 279
active SDK, targeting, 285
Consumer key and Consumer secret, retrieving, 284-285
Hello World application, 280-283
memory management, 301-302
MGTwitterEngine library
Delegate methods, 305-308
downloading, 285
initializing, 302
objects, creating in Interface Builder, 308-315
#pragma mark, 301
tweets, posting, 304-305
ViewController.h, 299
declaring properties and methods, 300
declaring variable instances, 300
importing libraries to header files, 299-300
ViewController.m, 300-301
xAuth
advantages of, 294
creating xAuth application, 294-299
definition of, 293-294
explained, 284
loading xAuth tokens, 302-304
requesting, 284
Twitter application for xAuth request, 283
verifying, 286-289

How can we make this index more useful? Email us at indexes@samspublishing.com
iPhone platform

iPhone platform, 30-31. See also iOS
iPod Touch platforms. See iOS

J-K

Java libraries, 256
JavaScript, adding Tweet button with, 210
JSON, parsing, 166-167
JTwitter library, 256
Krikorian, Raffi, 185

L

LAMP stacks
explained, 33-34
popularity of, 46
XAMPP, 35
installing, 35-38
security console, 39-40
lang attribute (search), 162, 165, 171
launching Hello Android project, 246-248
layout of Android OAuth application, 257-259
leave() function, 195
libraries
cURL, 53-55, 58
importing to implementation files, 301
MGTwitterEngine, initializing, 302

M

MGTwitterEngine library
Delegate methods, 305-308
downloading, 285
Oauth-signpost, 251
Twitter Java libraries, 256
Twitter4J, 251
limits on Twitter use, 11, 18
Linux, 33
XAMPP installation, 37-38
XAMPP security console, 40
List Members resources, 143
List Subscribers resources, 143-144
list.php, 137-138
lists
API support for, 135-136
List Members resources, 143
List resources, 142
List Subscribers resources, 143-144
creating
base.js, 138
call_direct() function, 140
call_showList() function, 140
commandLine.php, 138-139
createList() function, 139
createListItem() function, 138
friendshipExists() function, 141
header.inc, 139
list.php, 137-138
parseTwitter.php, 140-141
renderLists() function, 141
showLists() function, 141
twitteroauth.php, 139, 141
definition of, 135
loading
xAuth tokens, 302-304
XML resources, 261
local web servers, configuring, 34-38
locale attribute (search), 162, 171
logic, business logic, 27
Lu, Yiying, 229

Mac OS. See also iOS
XAMPP installation, 37
XAMPP security console, 40
main.css, 106-107
creating, 61-63
Retweet button support, 122
main.php, creating, 63, 69-70
makeText() method, 265, 267
mashups, 25-27
max_id parameter (getMentions() function), 99
memory management, iOS, 301-302
mentions, adding support for, 99-101
messages
compared to statuses, 58
direct messages, 125
adding API support for, 101-102, 109-110, 127-130, 131-132
call_direct() function, 101

callPage() function, 107-108
deleting, 132-133
destroy API call, 132-133
direct() function, 128
friendshipExists() function, 129-130
getMessages() function, 101, 131
renderTweets() function, 128
sanitizing, 110
sendMessage() function, 107-108
testing whether messages can be sent, 126-127
UI elements, adding, 125-126
getMessagesSent() function, 132
sending message to Twitter, 108-109

metadata mode (Search), 173
methods. See specific methods

MGTwitterEngine library
Delegate methods, 305-308
downloading, 285
initializing, 302

MGTwitterEngineDemo, 285
MGTwitterEngineDemoViewController.h, 280-283

microbloggers, 24
mobile platforms, 30-31
Mubarek, Muhammed Hosni Sayed, 5-7

multiple parameters, 80
MySQL, 34, 40-41

N

NAT (Network Address Translation), 38

navs.cc, 115-117

Netbeans, 44

Network Address Translation (NAT), 38

new users, 24

news readers, 21-22

NewsSnacker, 21-22

Notepad++, 43

notification methods, 197
notifications

disabling, 197

enabling, 197

notifications/follow method, 164

notifications/leave method, 197

twitterOAuth class

adding functions to, 90-92

creating, 85-87

gUserTimeline() function, 90-92

oauth_index.php, 87-88
twitteroauth.php, 88-92

OAuth class, creating, 83

oauth_index.php, 87-88

OAuthRequest() function, 88

Oauth-signpost, 251

objects

creating in Interface Builder, 308-315
definition of, 81-82

Odeo, 2-3

onCreate() function, 261

onNewIntent() method, 266

organizing files, 72

O

OAuth, 255-261

adding, 262-264

Android OAuth application creating, 256

intent filters and permission, 259-260

layout, 257-259

Twitter Java libraries, 256

XML resources, 261
definition of, 82

flow overview, 84

OAuth class, creating, 83

Twitter connection errors, handling, 92-93

P

packages, importing, 261-275

adding OAuth, 262-264

authenticating application, 264-266

responding after authentication, 266-269

TwitterOAuth.java, 269-275

page attribute (search), 163-165, 171

Page parameter, 74, 100

parameters

explained, 73-75

for getMentions() function, 99-100

How can we make this index more useful? Email us at indexes@samspublishing.com
parseTwitter() function

parseTwitter() function, 70
parseTwitter.php, 118

account methods, 200
creating, 66-67
direct message support, 132
favorites support, 148
Friendships methods support, 196-197
list support, 140-141
parsing JSON, 166-167
recent, daily, and weekly

trends, 183-184
Retweet button support, 121
search support, 166-167
thumbnail viewer, creating, 157
trending topics support, 178-179
user API methods, 154

parseTwitterReply() function, 63, 66
parseTwitterReply.php, 70
parsing JSON, 166-167
passwords, choosing, 12-13
Perl, 33
permissions, adding to Android
OAuth application, 259-260
PHP, 33-34, 53-57
phpinfo() function, 40
phpMyAdmin, 41-42
platforms, 30-31
plus symbol (+), 258
POST calls, 136-137
posting tweets (iOS), 304-305
postMessage() method, 304-305
power users, 23
PR managers, 23
private accounts, 2

$profile_image_url variable, 66
profiles
profile colors, 201-203
profile images, 201

properties, declaring, 300
protocols
Hypertext Coffee Pot Control
Protocol, 230
NAT (Network Address
Translation), 38
SMS (Short Message
System), 2-3
public relations managers, 23
pure chat applications, 27-28
Python, 33

Q-R
q attribute (search), 171
rate limiting, 11, 18
readers. See clients (Twitter)
recent trends, 180-185
header.inc, 183
parseTwitter.php, 183-184
twitteroauth.php, 184-185
recommended toolbox, 45
refreshing search results, 173
registering applications, 15-16,
82-83
renderLists() function, 141
render.php
creating, 67-68, 70-71
direct message support,
127-128
favorites support, 149-152
Friendship methods support,
194
Retweet button support, 121
renderTweets() function, 63,
67, 128
@reply, 2, 8
requestFailed: method, 305-308
requesting Twitter xAuth, 284
requests (Search), 164-165
requestSucceeded: method,
305-308
responding after authentication,
266-269
response codes (HTTP). See HTTP
response codes
responsefromServer() function,
108
REST API and General section
(dev.twitter.com), 214
result_type attribute (search),
163, 172
retrieving Consumer key and
Consumer secret, 284-285
retrying if Twitter is down,
233-236
retweet API call, 119-123
Retweet button
base.js, 122
id/retweeted_by API call, 122
id/retweeted_by/ids API call,
123
main.css, 122
parseTwitter.php, 121
render.php, 121
retweets/id API call, 122
sendMessages.php, 120
twitteroauth.php, 120
statuses/firehose method

statuses/firehose method, 224-225
statuses/followers API method, 156
statuses/friends API method, 156
statuses/links method, 225
statusesReceived: method, 305-308
statuses/re_tweet method, 225
statuses/sample method, 225-226
Stenberg, Daniel, 86
streaming, 219
advantages of, 226
limits on, 221
pre-launch checklist, 221-222
site streams, 220
Streaming API, 219
streaming methods, 222-226
statuses/filter, 223-224
statuses/firehose, 224-225
statuses/links, 225
statuses/re_tweet, 225
statuses/sample, 225-226
user streams, 219-220
when to use, 220
Streaming API, 219
Streaming API Documention and Search API section (dev.twitter.com), 215
structured displays, 29
Subversion (SVN), 44
SVN (Subversion), 44

tTabs, supporting, 114-117
header.inc, 114-115
index.php, 95-96, 117
navs.cc, 115-117
targeting active SDK, 285
testing direct messages, 126-127
text editors, 43
text fields. See input text fields
TextMate, 43
thumbnail viewer, creating
header.inc, 156-157
parseTwitter.php, 157
twitteroauth.php, 158
time zone, displaying, 68
toLocaleString() method, 267
trending topics
call_trends() function, 178-179
definition of, 2, 177
recent, daily, and weekly trends, 180-185
call_trends_daily() function, 183
header.inc, 183
parseTwitter.php, 183-184
showTrends_current() function, 184
showTrends_daily() function, 184
showTrends_weekly() function, 184
showTrends() function, 178-179
supporting in applications
header.inc, 178
parseTwitter.php, 178-179
twitteroauth.php, 180
Trends/available API call, 185-187
twitteroauth.php, 184-185
trends. See trending topics
Trends/available API call, 185-187
trim_user parameter (getMentions() function), 100
troubleshooting
Twitter HTTP response codes, 65-66
XAMPP installation, 38
Tweet button, 209
adding with iframe, 210
adding with JavaScript, 210
customizing, 210-211
tweets
character limit for, 8
definition of, 2
use case studies
#blamedrewscancer, 5
Egyptian revolution, 5-7
Twitter API, future of, 236-237
Twitter clients, creating, 16-18, 59-60, 199
Android applications, 241-242
ADT (Android Development Tools) plug-in, 242, 252
Android OAuth application, 255-261
AVD (Android Virtual Device), 242
development environments, 252
Hello Android project, 243-251
importing packages, 261-275
supported operating systems, 252
xAuth, 275-276
API calls. See API calls
application architecture diagram, 231-232
block methods, 198
blocks, 198
catching API requests, 232-233
direct messages
adding API support for, 101-102, 109-110, 127-130, 131-132
call_direct() function, 101
callPage() function, 107-108
deleting, 132-133
destroy API call, 132-133
direct() function, 128
friendshipExists() function, 129-130
getMessages() function, 101, 131
getMessagesSent() function, 132
renderTweets() function, 128
sanitizing, 110
sending message to Twitter, 108-109
sendMessage() function, 107-108
testing, 126-127
UI elements, adding, 125-126
favorites
adding to applications, 148-149
createFavorite() function, 150
creating, 149-152
definition of, 147
destroyFavorites() function, 152
destroying, 152
favorite() function, 149
showFavorites() function, 148
Friendships methods explained, 193
supporting in applications, 194-197
home timeline, 97-99
HTTP response codes, 65-66
index.php, 60-61, 69, 95-96
input text fields, 108
base.js, 107-108, 110
createMessage.php, 106
index.php, 105-106
main.css, 106-107
sendMessage.php, 108
iOS. See iOS
lists
API support for, 135-136
creating, 137-141
definition of, 135
List Members resources, 143
List resources, 142
List Subscribers resources, 143-144
main.css, 61-63
main.php, 63, 69-70
mentions
adding support for, 99-101
call_timeline() function, 101
getMentions() function, 99-101
notifications
disabling, 197
enabling, 197
parseTwitter.php, 66-67
render.php, 67-71
retrying if Twitter is down, 233-236
Retweet button
base.js, 122
id/retweeted_by API call, 122
id/retweeted_by/ids API call, 123
main.css, 122
parseTwitter.php, 121
render.php, 121
retweets/id API call, 122
sendMessage.php, 120
twitteroauth.php, 120
Search. See Search
streaming, 219
advantages of, 226
limits on, 221
pre-launch checklist, 221-222
site streams, 220
Streaming API, 219
streaming methods, 222-226
user streams, 219-220
when to use, 220
tabs. See tabs

How can we make this index more useful? Email us at indexes@samspublishing.com
trending topics
call_trends() function, 178-179
definition of, 166-167
recent, daily, and weekly trends, 180-185
showTrends() function, 178-179
supporting in applications, 177-180
Tweet button, 209
adding with iframe, 210
adding with JavaScript, 210
customizing, 210-211
twitterAPI.php, 64-65, 70
Twitter statistics, 29-30
Twitter xAuth. See xAuth
Twitter4J library, 251, 256
twitterAPI.php, creating, 64-65, 70
TwitterHolics, 24
twitterOAuth class
adding functions to, 90-92
creating, 85-87
getUserTimeline() function, 90-92
oauth_index.php, 87-88
Twitter connection errors, handling, 92-93
twitteroauth.php, 88-92
TwitterOAuth.Java, 269-275
twitteroauth.php, 88-92, 119
account methods, 200
direct message support, 129-132
favorites support, 148, 150, 152
Friendships methods support, 195
list support, 139, 141
recent, daily, and weekly trends, 184-185
retrying if Twitter is down, 233-236
Retweet button support, 120
search support, 167-168
thumbnail viewer, creating, 158
trending topics support, 180
user API methods, 154
$twitterResponseData variable, 64
TwittFilter, 29-30
Twurl Web Console, 206

U

UITextField object, 309
unblocking users, 198
until attribute (search), 163, 171
$update variable, 66
updateStatus() function, 109, 267
$updateTime variable, 66
updating profile images, 201
URL shortening, 215
URLs
Twitter URLs, 12
vanity URLs, 14, 18
user API methods
accessing other user information, 155
adding to applications, 154
header.inc, 154
twitteroauth.php, 154
list of, 153
statuses/followers, 156
statuses/friends, 156
thumbnail viewer, creating, 156-158
header.inc, 156-157
parseTwitter.php, 157
twitteroauth.php, 158
users/profile_images, 156
users/search, 155
users/show, 153-154
users/suggestions, 155
users/suggestions/:slug, 155-156
user streams, 219-220
User_ID parameter, 74
user_timeline API, 55-57
:user/:list_id/subscribers/:id method, 144
:user/:list_id/create_all method, 143
:user/:list_id/members method, 143
:user/:list_id/subscribers method, 144
:user/:list_id/subscribers/:id method, 144
:user/:lists method, 142
:user/ids method, 142
:user/ids/:statuses method, 142
:user/lists/memberships method, 142
:user/lists/subscriptions method, 142
users, 21
blocking, 198
bots, 24-25
chatters, 22-23
high-frequency users, 24
microbloggers, 24
new users, 24
news readers, 21-22
power users, 23
PR managers, 23
unblocking, 198
users/profile_image API method, 156
users/search API method, 155
users/suggestions API method, 155
users/suggestions/:slug API method, 155-156

V
vanity URLs, 14, 18
variables, declaring, 300. See also specific variables
verifying xAuth, 286-289
ViewController.h, 299
declaring properties and methods, 300
declaring variable instances, 300
importing libraries to header files, 299-300
ViewController.m, 300-301
viewDidLoad method, 302-304
views, Hello Android project, 246
Vim, 43

W
web interface, 237
web servers
configuring, 34-38
security, 38-41
MySQL, 40-41
phpMyAdmin, 41
XAMPP pages, 40
XAMPP security console, 39-40
websites, dev.twitter.com, 205
weekly trends, 180-185
header.inc, 183
parseTwitter.php, 183-184
showTrends_weekly() function, 184
twitteroauth.php, 184-185
Where On Earth ID (WOEID), 185-186
whitelisting, 11-12, 31
widgets, 25-26, 237
Williams, Abraham, 83
Windows
XAMPP installation, 35-37
XAMPP security console, 39-40
WOEID (Where On Earth ID), 185-186

X-Y-Z
XAMPP, 35
installing, 35-38
on Linux, 37-38
on Mac OS, 37
troubleshooting installation, 38
on Windows, 35-37
security console, 39-40
oxAuth, 275-276
advantages of, 294
creating xAuth application, 294-299
definition of, 290
explained, 284
loading xAuth tokens, 302-304
requesting Twitter xAuth, 284
Twitter application for xAuth request, 283
verifying, 286-289
ViewController.h, 299
declaring properties and methods, 300
declaring variable instances, 300
importing libraries to header files, 299-300
ViewController.m, 300-301
xauthViewController.xib, 308-315
xauthViewController.m, 300-301
Xcode, 280-283
XML
API calls, creating, 49-52
XML resources (Android OAuth application), 261

How can we make this index more useful? Email us at indexes@samspublishing.com