**Consuming a RESTful Service (bit.ly) in an iPhone Application**

By [**Mugunth Kumar, Singapore**](http://www.codeproject.com/script/Membership/View.aspx?mid=1557696) | 30 Sep 2009 | [Technical Blog](http://www.codeproject.com/KB/FAQs/BlogFAQ.aspx)



**Introduction**

Of late, many programmers who develop for Windows/Linux have moved to iPhone because of its runaway success. Among the most successful apps on the Apple AppStore, first comes games. Next comes apps like Tweetie and others those consume a Web Service (REST or otherwise). Games are pretty difficult to get started especially if you aren’t a game programmer. However, developing iPhone applications that consume a Web Service is quite easy. In this article, I’ll illustrate how to write an iPhone application to consume a Web Service. I’ll take bit.ly’s REST service as an example, and at the end, we will be developing a nice “bit.ly” wrapper in Objective-C. For those who are new to REST, it stands for Representational State Transfer. Head on to Wikipedia [here](http://en.wikipedia.org/wiki/Representational_State_Transfer) for more details. The post itself is split into two parts, one focusing XML and the other JSON. At the end of the second part, you can download the source code attached.

**Bit.ly documentation**

Bit.ly, as you all know, is a URL shortening service that became popular all of a sudden when Twitter started using bit.ly as its default URL shortening service over tinyurl.com. The REST documentation of bit.ly is available from Google Code [here](http://code.google.com/p/bitly-api/wiki/ApiDocumentation).

**Some basics before we dirty our hands with real code**

**Authentication types**

bit.ly (and many such services) use a kind of HTTP authentication called Basic Authentication. In basic authentication, the username and password that are typed into the client are formatted concatenated with a **:** and the resultant string is converted to "base64" encoding. This string is passed along with the HTTP Header usually in every API call that does some function which you have to normally log on to see. There is one more type of authentication, oAuth, which stands for Open Authentication. As of now, bit.ly doesn’t use oAuth. Explaining oAuth itself needs a separate article and is outside the scope of this article.

**HTTP Request types**

When you access a RESTful service, you either request data or post data. In most cases, your HTTP Request type is either a “GET” or a “POST”. There are two more types of requests, and they are, “PUT” and “DELETE”. When to use what is up to the designer of the RESTful service. You, as a consumer, just follow the documentation. In our case, bit.ly will be using only “GET” methods.

**The real code**

You shouldn’t be shocked to know that accessing a RESTful service for data is just a matter of 10 lines of code.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

NSString \*longURL = @"http://mugunthkumar.com"

NSString \*baseURLString =

@"http://api.bit.ly/shorten?version=2.0.1&" +

@"longUrl=%@&login=bitlyapidemo&apiKey=yourapikeyhere";

NSString \*urlString = [[NSString alloc] initWithFormat:@"%@%@",

baseURLString, longURL];

NSURL \*url = [[NSURL alloc] initWithString:urlString];

NSString \*result = [[NSString alloc] initWithContentsOfURL:url];yourapikeyhere

To see the result:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

NSLog(result);

Result:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

[url release];

[urlString release];

[result release];

**Parsing the result**

The result which we “NSLogged” here could be either XML or JSON. In our case, bit.ly by default sends JSON formatted data, which is by far the most commonly used. JSON is lighter than XML for transferring the same amount of data. There is a lot of debate going on about which is good for data transfer. I personally prefer JSON. Of late, JSON is picking up over XML with the advent of JSON parsers. Objective C has a very robust JSON parser called [json-framework](http://code.google.com/p/json-framework/). We will look at it in the next part of this article. The other kind of data format namely XML, is returned by bit.ly, only when you pass:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

format=xml

along with the URL as an additional parameter. For parsing XML, there are two parsers. One is NSXMLParser and the other is a faster libxml2. Though libxml2 is faster, use it on your iPhone app only when you are going to parse over 10 MB of XML data. When your XML data is small (as in our case), the performance gap between NSXMLParser and libxml2 is very meager. In case of bit.ly, the shortened API returns < 1 KB of data. Most RESTful services return very little data (mostly in the order of a few hundred kilo bytes). I prefer NSXMLParser for one reason. Your code is cleaner than it would be if you use libxml2. In this article, I’ll take you through both ways of handling data.

**Parsing XML**

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

[parser initWithContentsOfURL: url];

[parser setDelegate:self];

[parser setShouldProcessNamespaces:NO];

[parser setShouldReportNamespacePrefixes:NO];

[parser setShouldResolveExternalEntities:NO];

[parser parse];

[parser release];

Parsing XML data using NSXMLParser couldn’t be simpler. You just have to create an object (parser) of type NSXMLParser, initialize it, and call parse.

**Things to note**

However, there are a couple of things to note.

1. You can initialize the parser object either with NSURL or with NSData. Both methods work equally well.
2. You **ought** to set the delegate to self. Otherwise, there is no way for the NSXMLParser to notify you of events.
3. The next three lines are optional. You usually set it to YES if you want to get notified of those events as well.
4. The call to [parser parse] is blocking. This means, it will not return until the complete content is parsed. So, you can (and should) release any associated memory at the next line.

**Callbacks from NSXMLParser**

NSXMLParser has around 14 callback methods. But it’s enough if you implement just three of them. They are:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

parser:didStartElement:namespaceURI:qualifiedName:attributes:

parser:didEndElement:namespaceURI:qualifiedName:

parser:foundCharacters:

The didStartElement is called when the XMLParser encounters an opening XML element. You usually initialize temporary variables and read attributes, if any, in this function. In the case of bit.ly, you just have to read the “shortUrl” element. Just handling this element alone will do.



***ShortURL from bit.ly***

Declare the following: NSStrings, currentShortURL, actualShortURL, and currentElement. Write the following code into the didStartElement block:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

- (void)parser:(NSXMLParser \*)parser didStartElement:(NSString \*)elementName

namespaceURI:(NSString \*)namespaceURI qualifiedName:

(NSString \*)qName attributes:(NSDictionary \*)attributeDict

{

currentElement = [[elementName copy] autorelease];

if ([elementName isEqualToString:@"shortUrl"]) {

currentShortURLString = [[NSString alloc] init];

}

Now, in the foundCharacters block, write the following code:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

- (void)parser:(NSXMLParser \*)parser foundCharacters:(NSString \*)string{

if ([currentElement isEqualToString:@"shortURL"]) {

[currentShortURLString appendString:string];

}

In the didEndElement block, write:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/39119/Consuming-a-RESTful-Service-bit-ly-in-an-iPhone-Ap)

- (void)parser:(NSXMLParser \*)parser didEndElement:(NSString \*)elementName

namespaceURI:(NSString \*)namespaceURI qualifiedName:(NSString \*)qName

{

if ([elementName isEqualToString:@"shortURL"]) {

actualURLString = currentURLString;

}

Now when the parsing ends, just after the [parser release] function, actualShortURLString will contain the shortened URL.

End of story. If you couldn’t follow the code in this blog post, you can always download the source code attached (in the next part). For more detailed information on parsing using NSXMLParser, follow [this article](http://codesofa.com/blog/archive/2008/07/23/make-nsxmlparser-your-friend.html).

**Parsing JSON**

Parsing JSON is much simpler than XML. In fact, I hate to call it as "parsing" as you don't even have to write a parser!!! By using the [open source json-framework kit for iPhone](http://code.google.com/p/json-framework/) you can easily "convert" a JSON string into a NSDictionary. Now let's have a look at the code.

**JSON Parsing Code**

JSON Parsing is far much simpler than XML. Infact, the code attached below uses JSON.

Parsing the bitly information is as simple as writing these three lines of code.

SBJSON \*jsonParser = [SBJSON new];  
NSString \*jsonString = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];  
NSDictionary \*dict = (NSDictionary\*)[jsonParser objectWithString:jsonString];  
The return string is a dictionary of key value pairs.To read a value, you can call objectForKey function within NSDictionary.

In most cases, including bit.ly, JSON return strings always contain other dictionaries nested within itself. To read a value within a dictionary, just nest your calls like this.  
NSString \*shortURL = [[[dict objectForKey:@"results"]  
objectForKey:f\_longURL]  
objectForKey:@"shortUrl"];

**Source Code and Documentation**

**Source Code**

Download the source code here  
[MKBitlyHelper.zip](http://blog.mugunthkumar.com/wp-content/uploads/MKBitlyHelper.zip)

**Documentation**

With just three lines of code, you can shorten your URL using this wrapper.

Initialize the helper class with your loginname and apikey.

bitlyHelper = [[MKBitlyHelper alloc] init];  
[bitlyHelper initWithLoginName:@"yourlogin" andAPIKey:@"yourapi"];

In your application, you can either provide your application specific API or user provided API. Logging into the bit.ly api helps in tracking the click throughs and referrals. The classes doesn't provide a login or APIKey by itself.

Now, shortening or expanding URLs is as easy as calling these functions.

NSString \*shortURL = [bitlyHelper shortenURL:@"http://mugunthkumar.com"];  
NSString \*longURL = [bitlyHelper expandURL:shortURL];

**Disclaimers and other yada yada...**

Be forewarned that it may have errors. As Donald Knuth says,

Beware of bugs in the above code; I have only proved it correct, not tried it.

Feel free you use this code and re-distribute it. The source code must retain the copyrights and my attribution in any derivative works of the source code.

On your application, you might opt to attribute me in your app though it's not mandatory. I would be happy if you do so ;-)

**Related posts:**

1. [iPhone Tutorial – In-App Email](http://blog.mugunthkumar.com/coding/iphone-tutorial-in-app-email/)

**License**

This article, along with any associated source code and files, is licensed under [The Microsoft Public License (Ms-PL)](http://www.opensource.org/licenses/ms-pl.html)

**About the Author**

|  |  |
| --- | --- |
| [**Mugunth Kumar, Singapore**](http://www.codeproject.com/Members/M-Mugunth-Kumar)  Software Developer Honeywell SingaporeSingapore  Member | Working as a Software Engineer @ Honeywell Singapore I also develop iPhone Apps, Windows Apps, Web Apps in my free time. Visit, http://mugunthkumar.com for more information |

Bottom of Form