Objective-C REST Client Update

Adrian Kosmaczewski

2009-01-19

I've uploaded (yet another) update to the Objective-C REST client I've blogged about previously. This time I've scanned the code with the excellent LLVM/Clang Static Analyzer and fixed a couple of memory leaks here and there. I strongly recommend to scan your own projects with this tool, it's extremely simple to use:

- Install it somewhere in your PATH;
- Set your projects to use the Debug configuration when building from the command line (you can do that in the inspector for the project, in the "Configurations" tab);

  (see Sebastien's comment below ;)
- Open Terminal.app and fire `scan-build -k -V xcodebuild` on the root of the Xcode project folder;
- If there are any problems with your code, you'll have your web browser pop up with the list of problems, their description in annotated code format, and even a link to open the file right away.

I have also fixed another problem with the code, which was preventing POST data to be properly sent to the server with the previous version. You might have encountered this problem, and here is the solution: instead of using NSString's stringByAddingPercentEscapesUsingEncoding: method, use CoreFoundation's CFURLCreateStringByAddingPercentEscapes() function. This means that this code:

```
[params appendFormat:@"%@=%@&", [key stringByAddingPercentEscapesUsingEncoding:NSUTF8StringE
[[parameters objectForKey:key] stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncodi
```

became this:

```
NSString *encodedKey = [key stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding];
CFStringRef value = (CFStringRef)[[parameters objectForKey:key] copy];
// Escape even the "reserved" characters for URLs
// as defined in http://www.ietf.org/rfc/rfc2396.txt
CFStringRef encodedValue = CFURLCreateStringByAddingPercentEscapes(kCFAllocatorDefault,
```

```
            value,
            NULL,
            (CFStringRef)@";/?:@&=+$,",
            kCFStringEncodingUTF8);
[params appendFormat:@"%@=%@&", encodedKey, encodedValue];
CFRelease(value);
CFRelease(encodedValue);
```

Now any text sequence, including any "legal" URL character (as explained in the RFC), will be encoded properly and sent to the server as required.

Cocoa only provides a thin layer over many of CoreFoundation functions and types; it does not expose completely all the functionality "below", and that's why sometimes you must dig a bit deeper and call CoreFoundation code to certain operations, like using the Address Book data, or playing sounds in your iPhone applications. The good thing, as always, is that CoreFoundation types are "toll free" bridged, which means that you can safely cast a CFStringRef into an NSString pointer without any overhead.

Chris Adamson explained this as an "Opt-in Complexity" pattern, in an article in the Inside iPhone blog last week:

> Need time zone awareness? NSTimeZone is your friend. Need to know every time zone that the device supports? Get to know `CFTimeZoneCopyKnownNames()`. Again, a niche-ier feature lives down at the Core Foundation level, and isn't wrapped by an equivalent call in Foundation, though it's easy enough to switch to procedural C and make the one-off lower-level call.