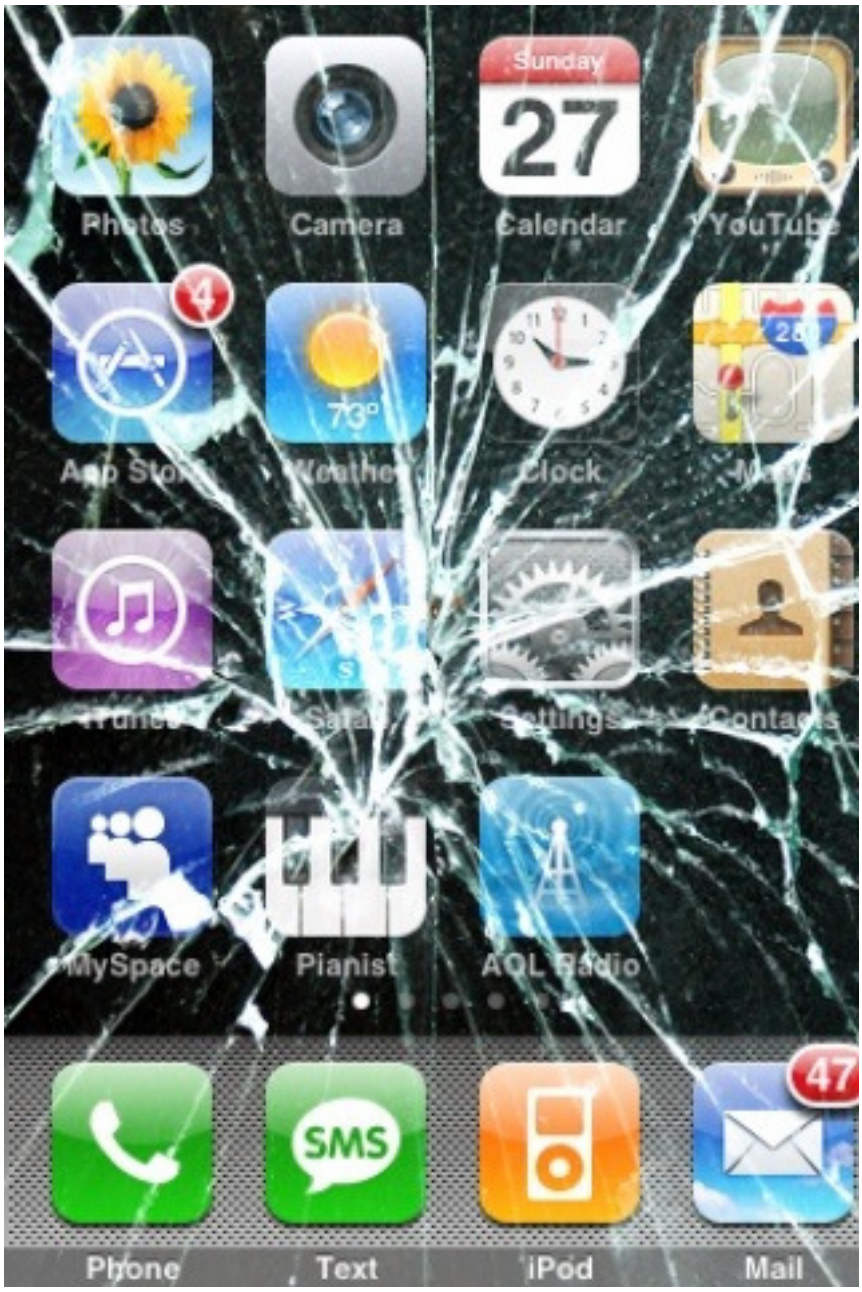
initWithContentsOfURL: Methods Considered Harmful

Adrian Kosmaczewski

2010-05-28

As I promised on Twitter[1], here's a small discussion about the problems brought by the "initWithContentsOfURL:" family of methods.

[1] http://twitter.com/akosma/status/14715706200



A quick search in the Xcode documentation browser brings in an interesting list of classes including this initializer (with or without additional parameters):

- NSArray
- NSManagedObjectModel
- NSData
- NSDictionary
- NSXMLParser
- NSMappingModel

- NSString
- AVAudioPlayer

Don't get me wrong, the title of this post is a bit misleading; it's not that this method is always problematic, particularly when the URL passed as parameter has been built using the fileURLWithPath: family of initializers. In this case, the URL will point to a local resource, and the worst thing that can happen in that case is that the filename is wrong, or that the file does not exist for some reason (NSFileManager to the rescue!).

However, when using external URLs referencing resources somewhere over the network, **using this method is definitely a recipe for disaster.**

The main problem with these methods, of course, is the fact that they are **synchronous**; this means that the thread executing them (usually the UI thread) will block completely until they return, and in most applications this means that you are de-facto blocking the whole application for an unknown amount of time. This means that no buttons or UI widgets will react to input, no navigation will be possible, no touch events will be delivered or executed, nothing will happen at all until the network operation completes.

Even worse; when using initWithContentsOfURL:, there is no timeout, there is no meaningful feedback for network failures, and no way for the user to cancel the current network operation. This last factor justifies by itself not using initWithContentsOfURL: at all; you must never ship code that leads to a bad user experience. Your users will resent this and will complain!

As Joel Spolsky explained in his classic article Three Wrong Ideas From Computer Science[2],

> One example of network transparency is the famous RPC (remote procedure call), a system designed so that you can call procedures (subroutines) running on another computer on the network exactly as if they were running on the local computer. An awful lot of energy went into this. Another example, built on top of RPC, is Microsoft's Distributed COM (DCOM), in which you can access objects running on another computer as if they were on the current computer.
>
> Sounds logical, right?
>
> Wrong.
>
> There are three very major differences between accessing resources on another machine and accessing resources on the local machine:
>
> 1. Availability,
> 2. Latency, and
> 3. Reliability.

(BTW, read this article, it's really a good one, like many others in his blog)

In summary, never assume that the network is available, never assume that anything behind the network is available, and never assume that a network has

[2] http://www.joelonsoftware.com/articles/fog0000000041.html

any particular speed. This is particularly true in the case of mobile platforms, where network conditions can vary tremendously from one minute to the other!

The problem with the initWithContentsOfURL: methods is that some online tutorials use them as a quick-and-dirty way to load resources from the network (for example this one on iCodeBlog[3]), and I have myself seen production code using this method.

This is very, very bad. Using initWithContentsOfURL: with remote network resources in a tutorial like this is simply a horrendous, almost irresponsible approach towards developers new to the iPhone platform.

You must not use synchronous connections in any way in your code, unless you are 100% sure that you are running it in a non-UI thread – and even so, as Jeff LaMarche explained recently[4], you should avoid such multithreaded approaches. The NSRunLoop architecture of Cocoa and Cocoa touch allows you to bypass threading altogether when dealing with networking operations. Remember this! And read Jeff's article for more details.

This exact same problem can also be created with other APIs and libraries, for example when using NSURLConnection's sendSynchronousRequest:returningResponse:error: method (as explained today in The Apple Blog[5].)

Do not block the main UI thread. Write this on your whiteboard, on a post-it on your computer monitor, in your agenda with a reminder every day. Do not use this approach. As Cédric said[6], these methods should only be used for prototyping!

One popular example of a network library that support asynchronous connections is ASIHTTPRequest[7], which I tend to use in nearly all my projects (instead of the standard NSURLConnection classes bundled in Cocoa and Cocoa Touch.) I prefer it because it has a nicer, smaller interface, it's fast (I've benchmarked it in my iPhone Web Services Client[8] demo project on Github), and it provides handy queueing capabilities. FTW!

By the way, Ben Copsey (the creator of ASIHTTPRequest) has published today an article in his blog[9], in which he exposes the approaches used to queue network connections in his library. Designing a library for concurrent, queued network connections is not trivial, and his current approach (subclassing NSOperationQueue) might soon be deprecated for a new, NSRunLoop-y, more Cocoa-friendly, way.

In any case, I hope I have made my point: avoid synchronous operations when loading content from the network, remember what Joel said, and use a library

[3] http://icodeblog.com/2009/06/19/using-nsxmlparser-to-pull-uiimages-from-the-web/

[4] http://iphonedevelopment.blogspot.com/2010/05/downloading-images-for-table-without.html

[5] http://theappleblog.com/2010/05/28/iphone-dev-sessions-responsive-web-enabled-iphone-apps/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+TheAppleBlog+%28TheAppleBlog%29

[6] http://twitter.com/0xced/status/14717929660

[7] http://allseeing-i.com/ASIHTTPRequest/

[8] http://github.com/akosma/iPhoneWebServicesClient

[9] http://allseeing-i.com/A-possible-replacement-for-ASINetworkQueue

providing asynchronous connections. Happy coding!

PS: by the way, the title of this blog post is a reference to many other classic articles, so many that there's even a Wikipedia entry about them[10] and a great rant about them[11] by Eric Meyer!

[10] http://en.wikipedia.org/wiki/Considered_harmful
[11] http://meyerweb.com/eric/comment/chech.html