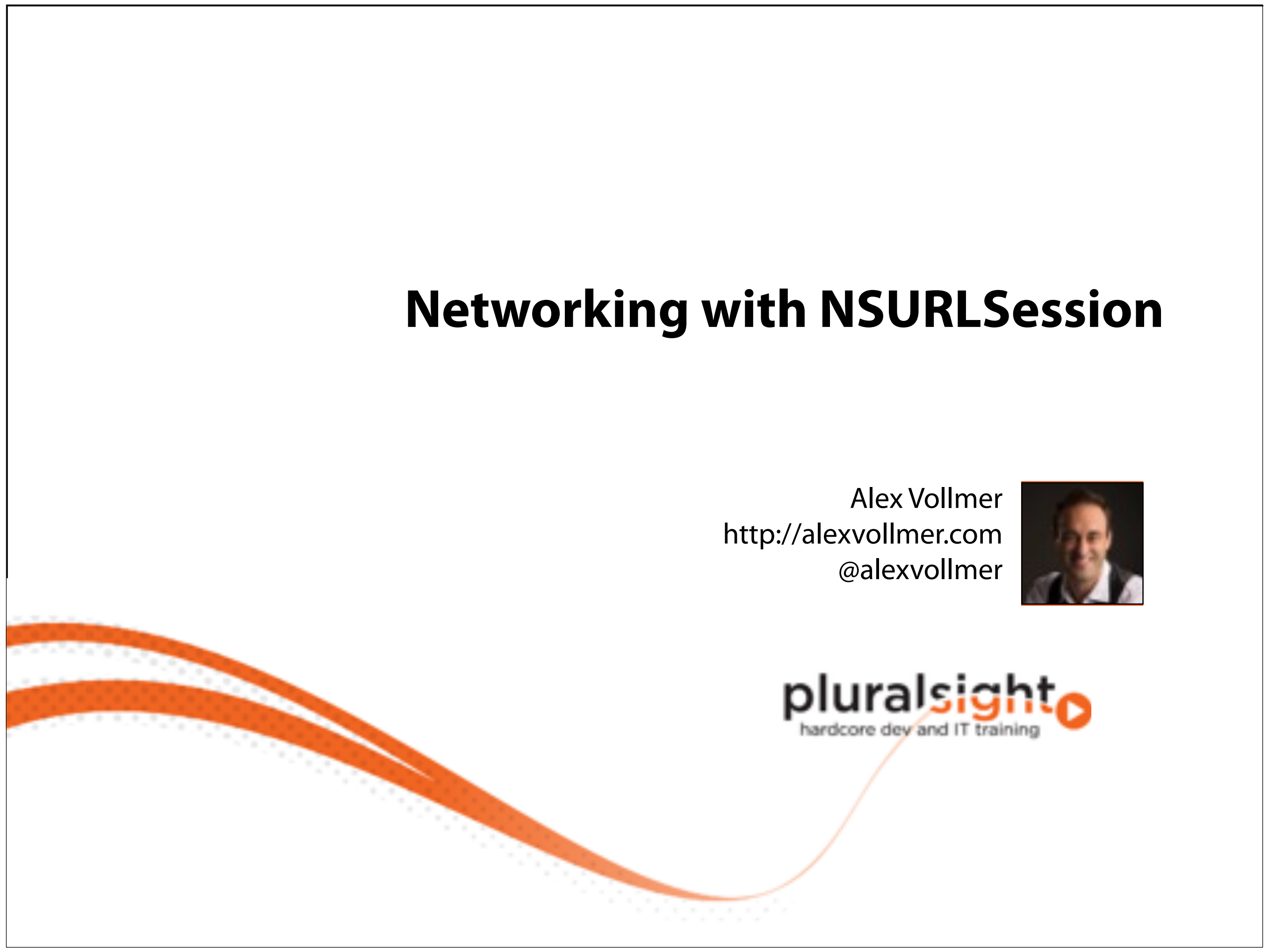


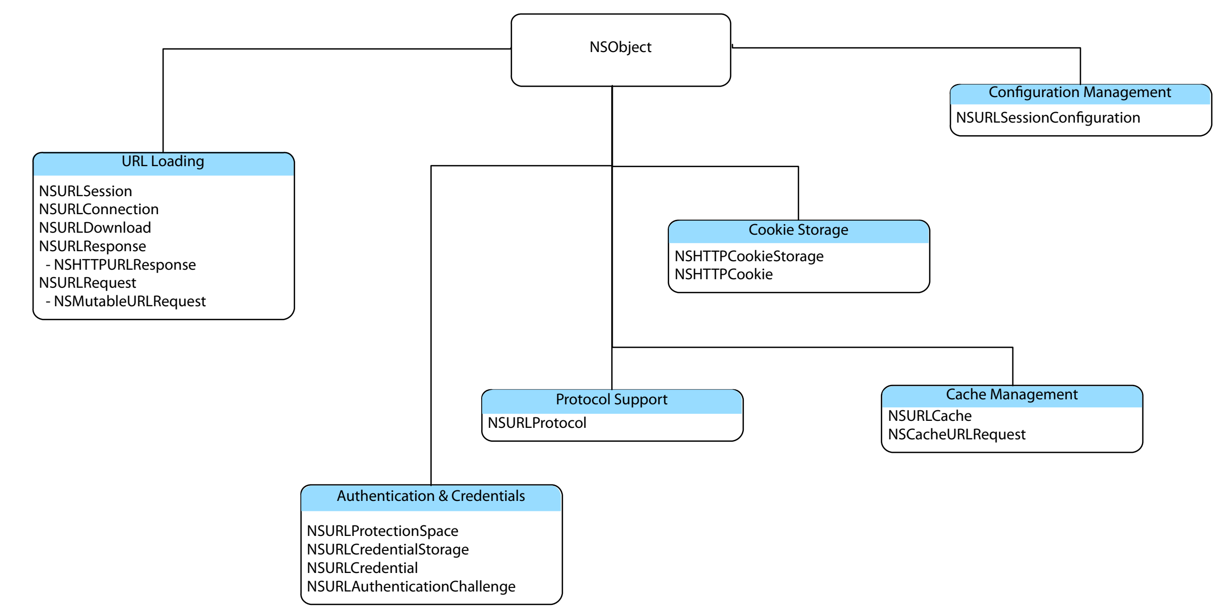
Networking with NSURLSession

Alex Vollmer
http://alexvollmer.com
@alexvollmer

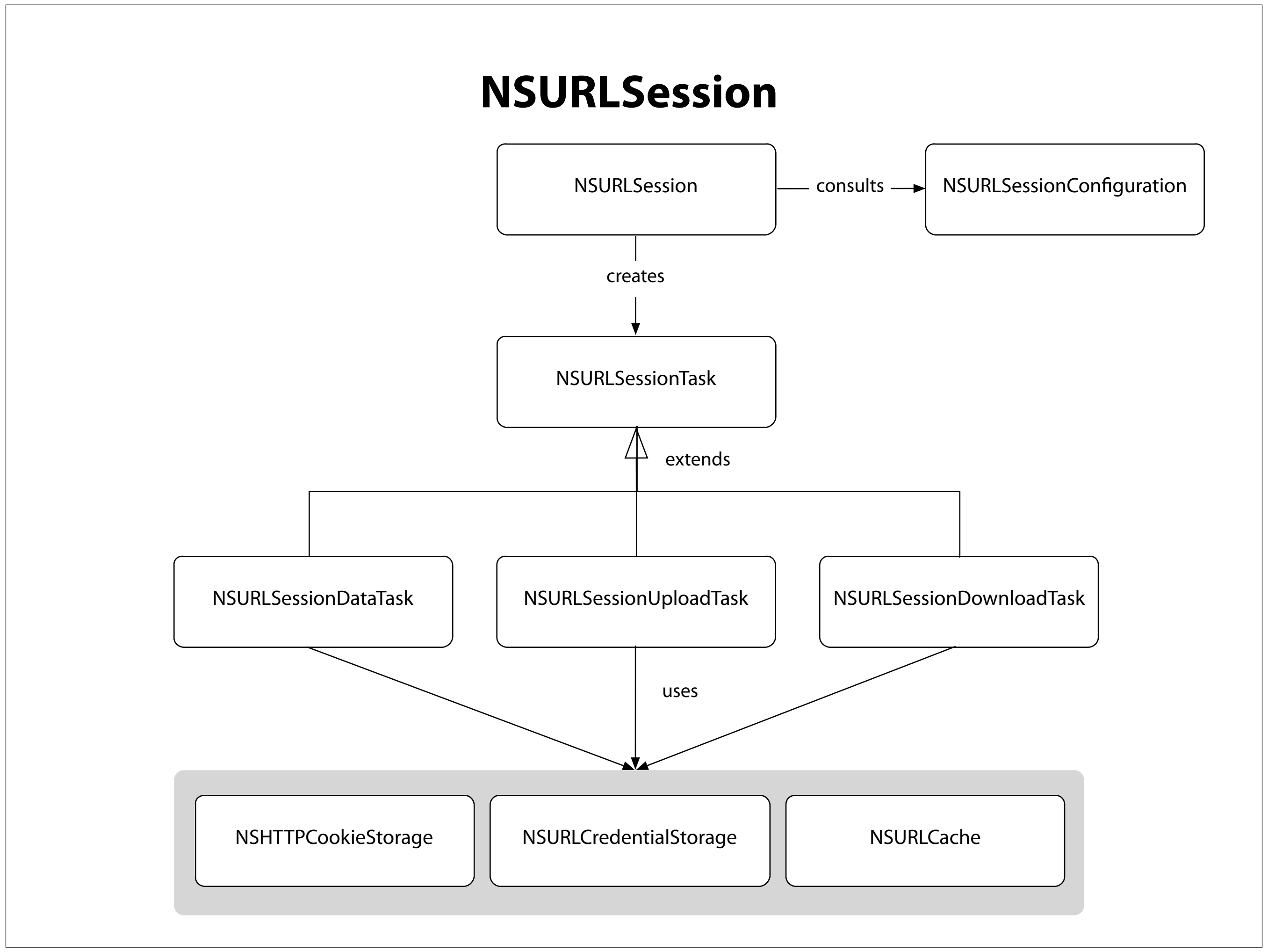
pluralsight
hardcore dev and IT training

In iOS 7 and Mac OS 10.9, Apple introduced NSURLSession—a new way to interact with remote resources via URL.

URL Loading System



In this module we're going to dive into NSURLSession and its related classes. Besides being the latest and greatest from Apple, NSURLSession's biggest advantage is the ability to upload and download resources even when your app isn't running.

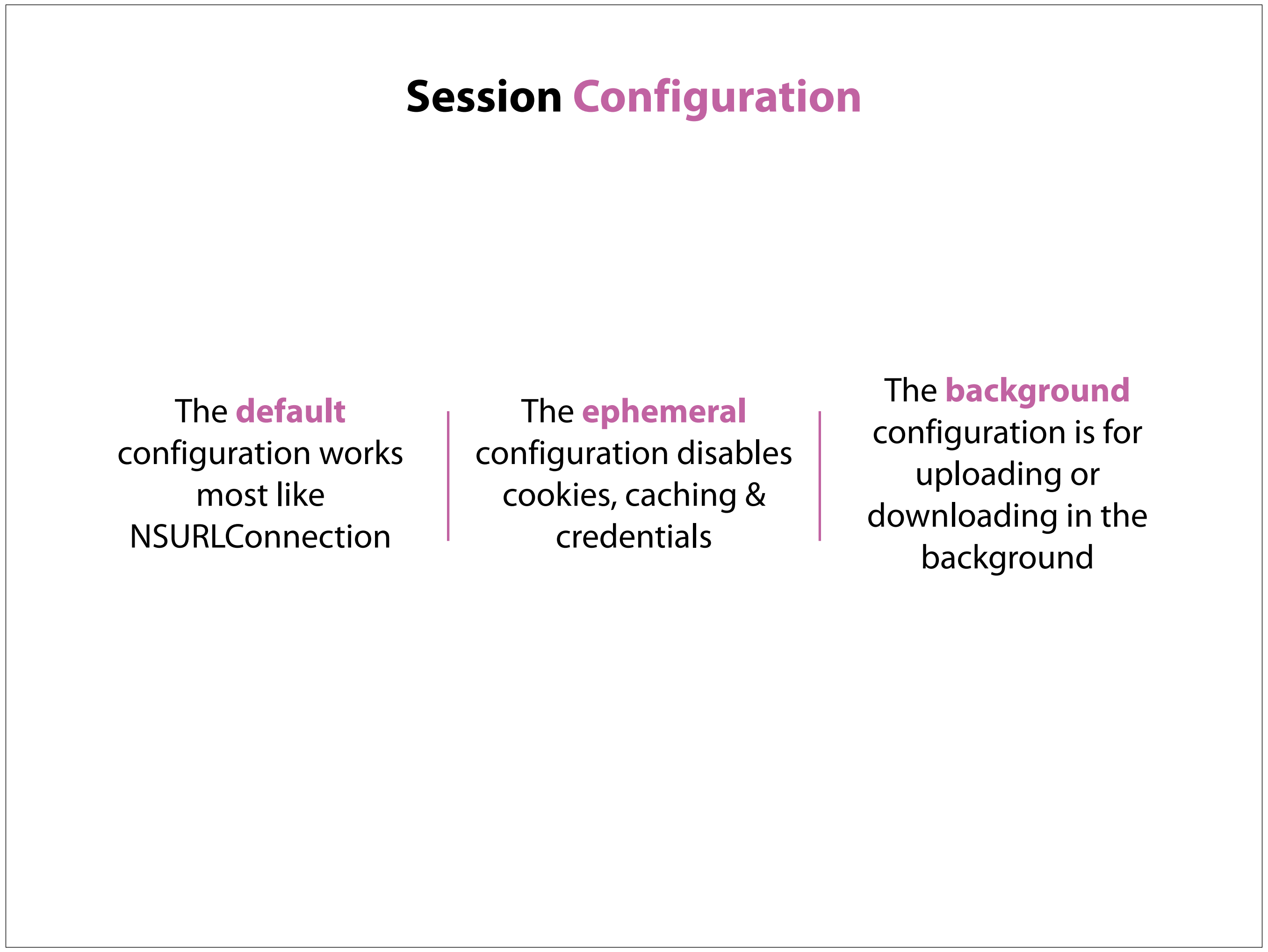


An instance of NSURLSession is essentially a factory for URL-loading tasks.

A session is configured by an instance of the NSURLSessionConfiguration class which governs things like the cookie acceptance policy, credential storage, caching and timeout policies.

A NSURLSession is used to create NSURLSessionTask instances, each of which is specific to a single URL. You don’t create NSURLSessionTasks instances directly, but one of its three subclasses: NSURLSessionDataTask, NSURLSessionUploadTask or NSURLSessionDownloadTask.

Based on the associated NSURLProtocol and the NSURLSessionConfiguration, NSURLSessionTasks instances will use the same URL-loading infrastructure as NSURLConnection for cookie management, credentials storage and caching.



Session **Configuration**

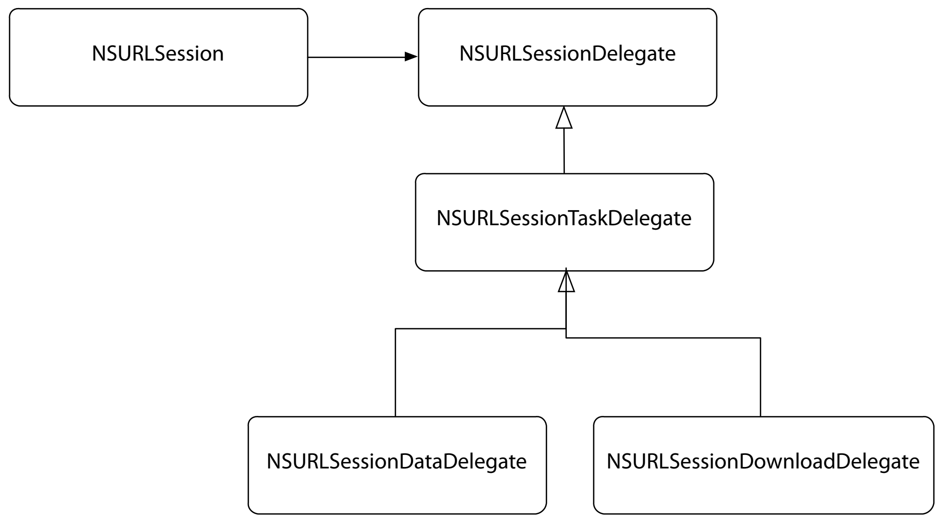
The **default** configuration works most like NSURLConnection

The **ephemeral** configuration disables cookies, caching & credentials

The **background** configuration is for uploading or downloading in the background

Session configurations come in three flavors: default, ephemeral and background.

NSURLSession Delegates



An NSURLSession instance can have a delegate assigned to it.

The basic delegate protocol, NSURLSessionDelegate, is called when the session is invalid or when receiving credential challenges for certain authentication types.

The NSURLSessionTaskDelegate protocol extends the NSURLSessionDelegate protocol and is used to notify the delegate of generic task-specific events.

The NSURLSessionDataDelegate protocol extends that protocol and notifies the delegate of task-related events specific to retrieving data.

The NSURLSessionDownloadDelegate protocol also extends the NSURLSessionTaskDelegate protocol for download-specific events.

Creating Data Tasks

```
/* Default task-creation (callbacks via delegate methods) */
- (NSURLSessionDataTask *)dataTaskWithRequest:(NSURLRequest *)request;

- (NSURLSessionDataTask *)dataTaskWithURL:(NSURL *)url;
```

```
/* Asynchronous convenience methods (bypasses delegate).
   These are not available for background tasks */
- (NSURLSessionDataTask *)dataTaskWithRequest:(NSURLRequest *)request
                           completionHandler:(void (^)(NSData *data,
                                                        NSURLResponse *response,
                                                        NSError *error))handler;

- (NSURLSessionDataTask *)dataTaskWithURL:(NSURL *)url
                       completionHandler:(void (^)(NSData *data,
                                                    NSURLResponse *response,
                                                    NSError *error))handler;
```

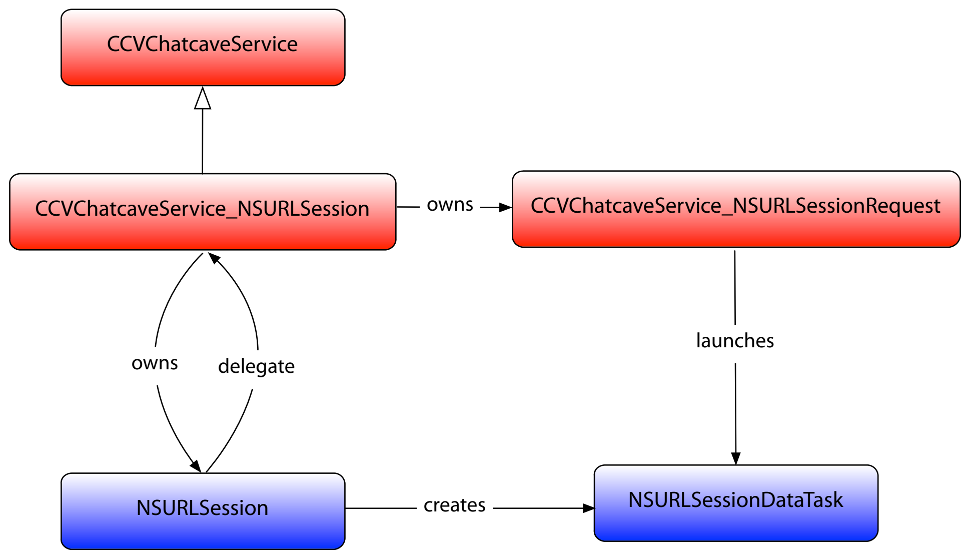
In the ChatCave app, we'll be using data-tasks to fetch data from the server. We can also use data-tasks for state-mutating HTTP operations like PUT and POST operations.

There are two styles of task-creation. The first style simply creates a task with a URL or URL request and relies on delegate callbacks.

The second style relies on Objective-C blocks to handle responses. The completion block is given an NSData instance for any body data, an NSURLResponse instance describing the envelope data and a NSError instance for any errors. If the tasks completes successfully, the error pointer will be nil.

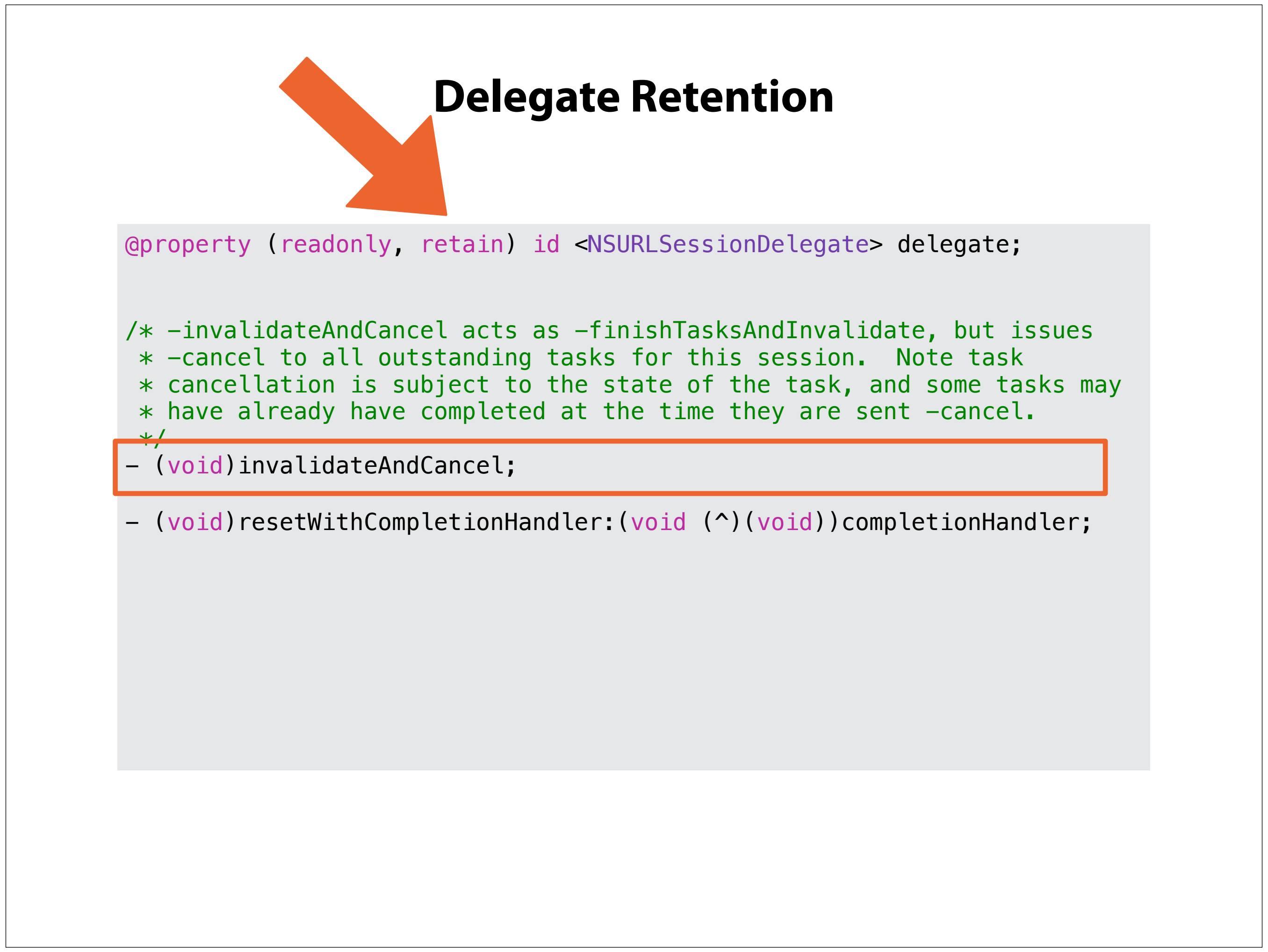
We'll use this style in the Chatcave application. One important note is that if you are implementing background downloading you *cannot* use the asynchronous convenience methods.

ChatCave Design



We'll start by subclassing CCVChatcaveService with a NSURLSession-specific implementation. It will be responsible for creating and managing a NSURLSession instance, as well as being its delegate.

We'll create another class to manage the individual tasks, which we'll call CCVChatcaveService_NSURLSessionRequest. Instances of this class will track the success and failure completion blocks from our higher level API methods in the CCVChatcaveService.



Delegate Retention

```
@property (readonly, retain) id <NSURLSessionDelegate> delegate;


/* -invalidateAndCancel acts as -finishTasksAndInvalidate, but issues
 * -cancel to all outstanding tasks for this session.  Note task
 * cancellation is subject to the state of the task, and some tasks may
 * have already have completed at the time they are sent -cancel.
 */
- (void)invalidateAndCancel;

- (void)resetWithCompletionHandler:(void (^)(void))completionHandler;
```

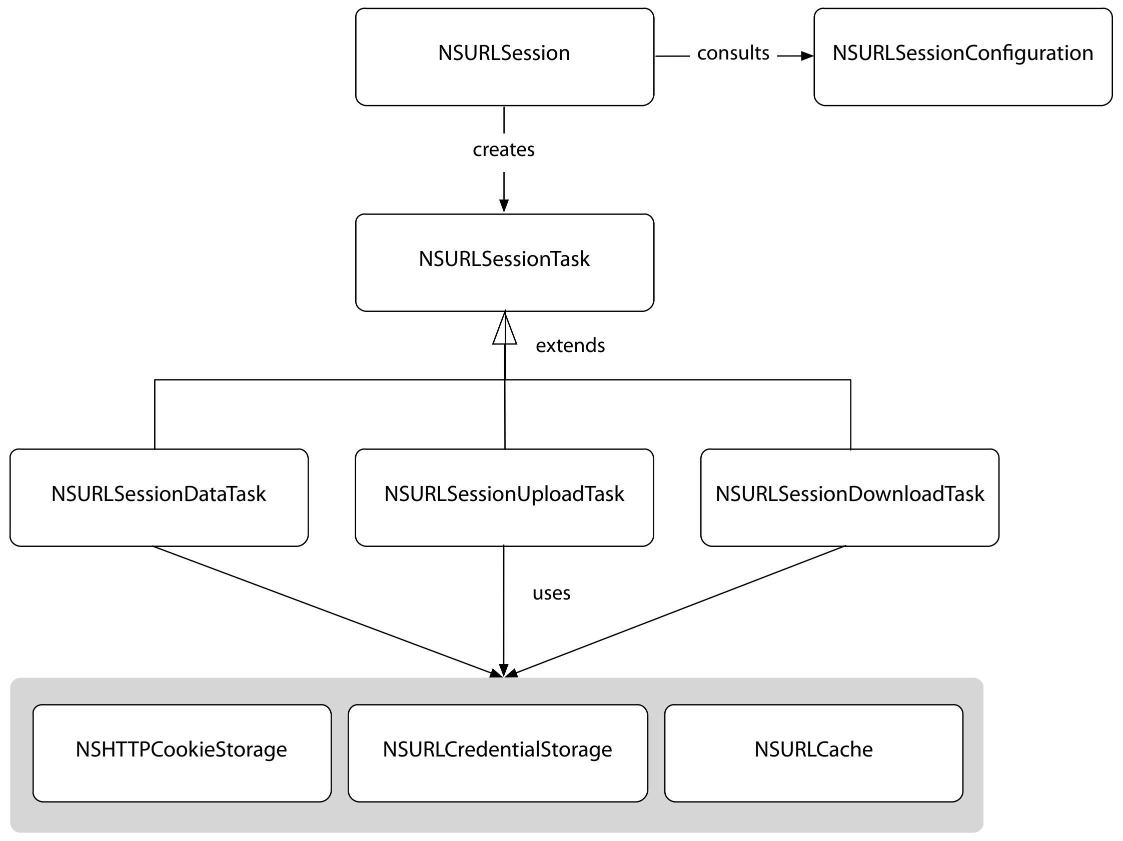
One word of caution about NSURLSession's delegate. Unlike most delegate properties in the Foundation classes, NSURLSession specifies the `retain` storage modifier for the delegate. The most likely scenario is that the object that creates and retains the NSURLSession is likely the delegate which means you are likely to have a retain cycle unless you manage things correctly.

There are two ways to invalidate the NSURLSession which will internally nil-out the delegate reference. You either need to invoke -invalidateAndCancel or -resetWithCompletionHandler:

Since the Chatcave application will only use a single NSURLSession for the lifetime of the app, we don't really need to be concerned about this. But you need to watch out for this if you are switching between instances of NSURLSession within your application.

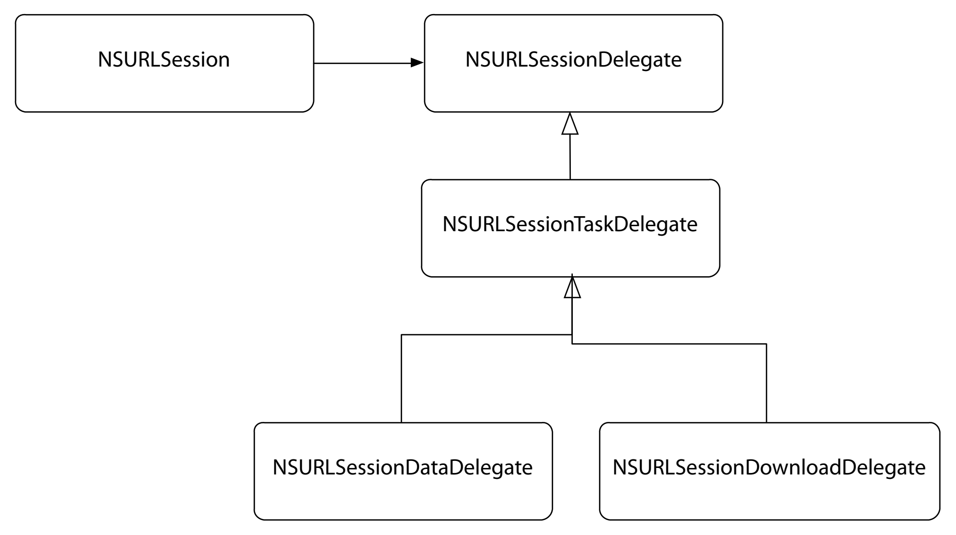
Let's dive in!

NSURLSession

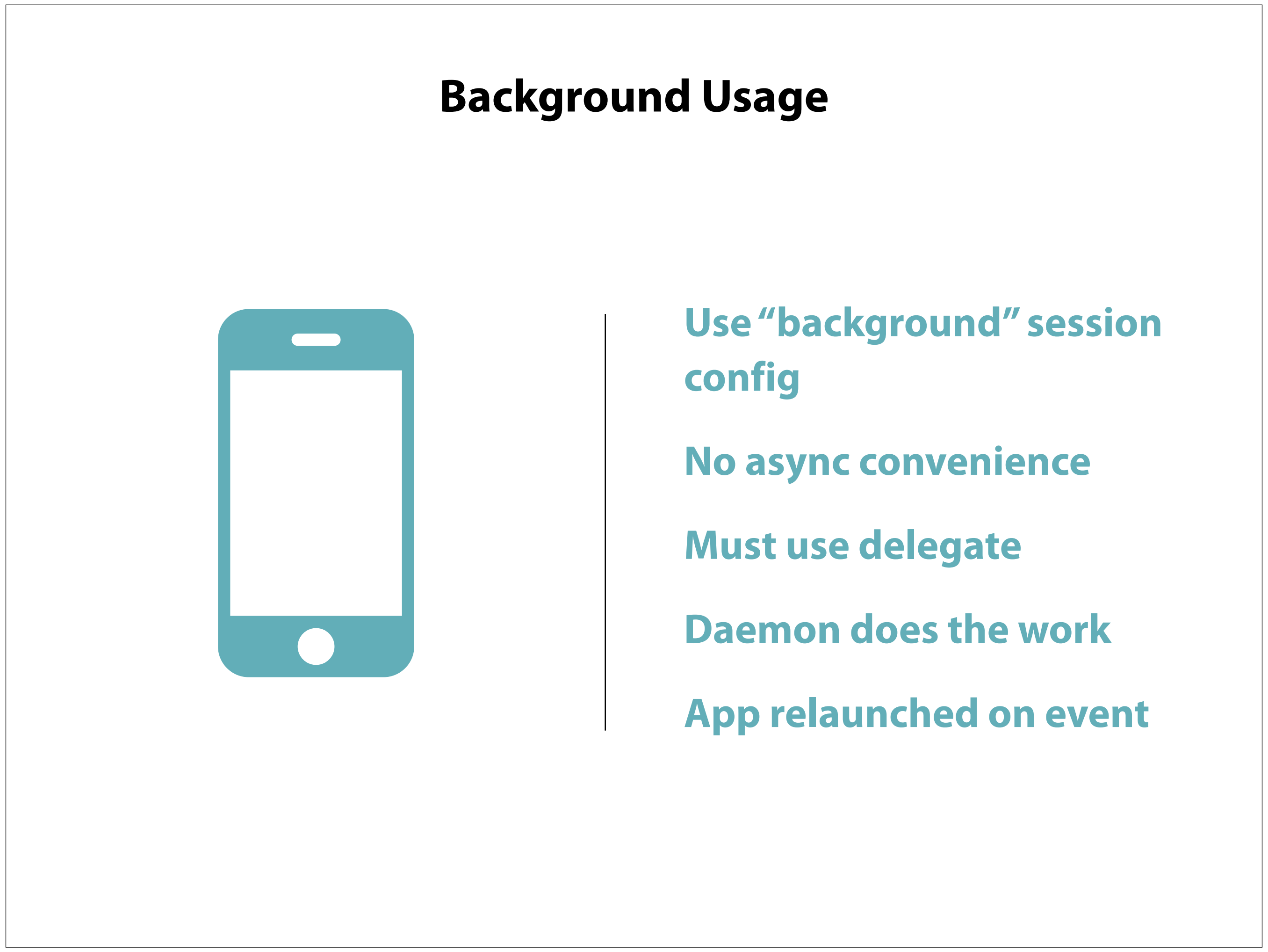


In this module we learned about NSURLSession and NSURLSessionTasks. We saw how it uses many of the same mechanisms as NSURLConnection for things like cookies, credentials and caching.

NSURLSession Delegates



We also learned how to use NSURLSessions' various delegate protocols to respond to specific events and customize behavior.



Background Usage

- Use “background” session config
- No async convenience
- Must use delegate
- Daemon does the work
- App relaunched on event

We also looked at NSURLSession’s unique ability to process in the background and how uploading and downloading can continue even when your application isn’t running.