NSURLSession

CocoaHeads Lyon - 15 mai 2014

Mustapha Tarek Ben Lechhab
@nsdeveloppeur
http://www.octiplex.com

Qui ?

- Ingénieur en Développement Logiciel à Octiplex
- Voir nos dernières (et prochaines) apps sur http://www.appstore.com/octiplex

Plan

- Historique des API réseau sur iOS
- Vue d’ensemble de NSURLSession
- 3 cas concrets
- Quelques points importants

Historique APIs

- Depuis iOS 2.0 : `NSURLConnection` & `CFNetwork`
- `ASIHTTPRequest` jusque 2011
- AFNetworking 1.0 depuis 2011
- Depuis fin 2013 : **`NSURLSession`** & AFNetworking 2.0

NSURLConnection ?

- Pas vraiment commode pour les cas d’utilisations les plus communs

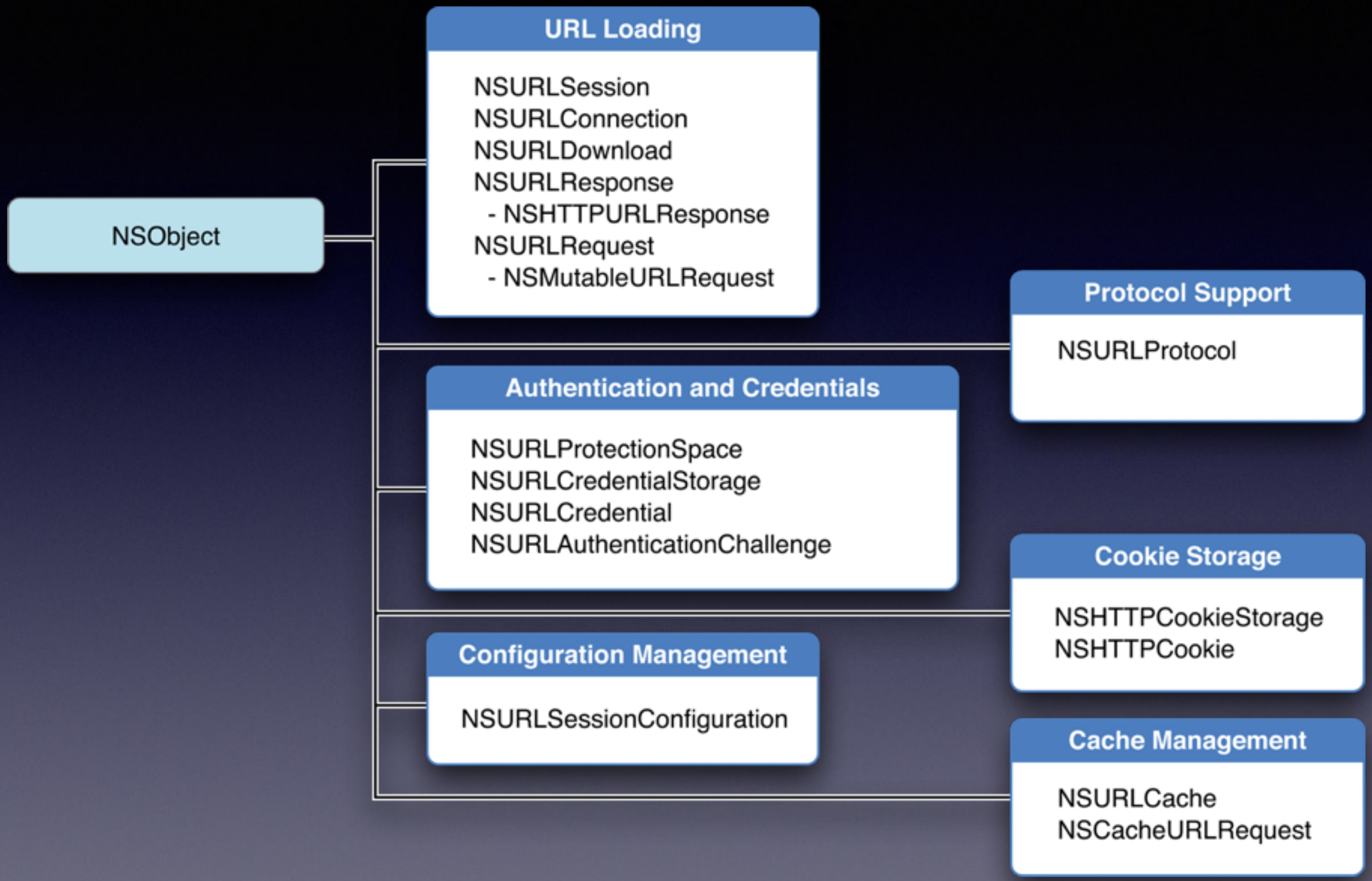
- Beaucoup de code générique pour gérer les headers, le body et la réponse des requêtes

- Éléments communs non réutilisables/ dissociables de manière simple (cache, timeout etc.)

NSURLSession

- Famille de classes autour de `NSURLSession`
- Ressemble fortement à AFNetworking d'un point de vue API
- Beaucoup d'éléments restent les mêmes : `NSURL, NSURLRequest, NSURLResponse`...

Hiérarchie de classes 1/2



© Apple : https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/URLLoadingSystem/URLLoadingSystem.html

Hiérarchie de classes 2/2

URL Session Class Hierarchy

The `NSURLSession` API consists of the following classes (nested to show subclass relationships):

- `NSURLSession`—A session object.
- `NSURLSessionConfiguration`—A configuration object used when initializing the session.
- `NSURLSessionTask`—The base class for tasks within a session.
    - `NSURLSessionDataTask`—A task for retrieving the contents of a URL as an `NSData` object
        - `NSURLSessionUploadTask`—A task for uploading a file, then retrieving the contents of a URL as an `NSData` object
    - `NSURLSessionDownloadTask`—A task for retrieving the contents of a URL as a temporary file on disk

In addition, the `NSURLSession` API provides four protocols that define delegate methods your app can implement to provide more fine-grained control over session and task behavior.

- `NSURLSessionDelegate`—Defines delegate methods to handle session-level events
- `NSURLSessionTaskDelegate`—Defines delegate methods to handle task-level events common to all task types
- `NSURLSessionDataDelegate`—Defines delegate methods to handle task-level events specific to data and upload tasks
- `NSURLSessionDownloadDelegate`—Defines delegate methods to handle task-level events specific to download tasks

Finally, the `NSURLSession` API uses a number of classes that are also commonly used with other APIs such as `NSURLConnection` and `NSURLDownload`. Some of these shared classes include:

- `NSURL`—An object that contains a URL.
- `NSURLRequest`—Encapsulates metadata related to a URL request, including the URL, request method, and so on.
- `NSURLResponse`—Encapsulates metadata related to a server's response to a request, such as the content MIME type and length.
    - `NSHTTPURLResponse`—Adds additional metadata specific to HTTP requests, such as response headers.
- `NSCachedURLResponse`—Encapsulates an `NSURLResponse` object, along with the actual body data of the server's response, for caching purposes.

App.net

- Réseau social à la Twitter (par et pour des personnes pour qui Twitter ne convient plus)
- API REST très intéressante, beaucoup de possibilités, réponses propres
- Espace de stockage de fichiers
- http://app.net pour s'inscrire puis http://developers.app.net
  Créer une application et obtenir un user token (pour se passer d'OAuth)

Cas d’utilisation 1

- Requête GET classique, sans authentification, avec block

```
NSURL *profileEndpoint = [NSURL URLWithString:@"http://example.org"];
NSURLSessionConfiguration *defaultConf = [NSURLSessionConfiguration defaultSessionConfiguration];
NSURLSession *currentSession = [NSURLSession sessionWithConfiguration:defaultConf];

NSURLSessionDataTask *getTask =
[currentSession dataTaskWithURL:profileEndpoint
            completionHandler:^(NSData *data, NSURLResponse *response, NSError *error)
                                {
                                    NSHTTPURLResponse *httpResponse = (NSHTTPURLResponse*)response ;

                                    if (httpResponse.statusCode == 200)
                                    {
                                        // Traiter données retournées
                                    }
                                    else
                                    {
                                        // Traiter erreur
                                    }

                                }];
[getTask resume];
```

Démo

Cas d’utilisation 2

- Requête GET download, sans authentification, sans block

```
NSURLSessionConfiguration *defaultConf =
  [NSURLSessionConfiguration defaultSessionConfiguration];

NSURLSession *session = [NSURLSession sessionWithConfiguration:defaultConf
                                                      delegate:self
                                                 delegateQueue:nil];

NSURLSessionDownloadTask *downloadTask = [session downloadTaskWithURL:url];
[downloadTask resume];
```

Cas d’utilisation 2

- Méthode de delegate pour l’avancement du téléchargement

```
- (void)URLSession:(NSURLSession *)session downloadTask:(NSURLSessionDownloadTask *)downloadTask
                                            didWriteData:(int64_t)bytesWritten
                                       totalBytesWritten:(int64_t)totalBytesWritten
                               totalBytesExpectedToWrite:(int64_t)totalBytesExpectedToWrite
{
    float progress = (float)totalBytesWritten/(float)totalBytesExpectedToWrite  ;
    dispatch_async(dispatch_get_main_queue(), ^{
        [self.progressView setProgress:progress animated:YES];
    });
}
```

Cas d’utilisation 2

- Méthode de delegate pour la fin du téléchargement

```
- (void)URLSession:(NSURLSession *)session downloadTask:(NSURLSessionDownloadTask *)downloadTask
                            didFinishDownloadingToURL:(NSURL *)location
{
    NSData *imageData = [NSData dataWithContentsOfFile:location.path];

    dispatch_async(dispatch_get_main_queue(), ^{
       self.image = [UIImage imageWithData:imageData];
    });
}
```

Démo

Cas d’utilisation 3

- Requêtes POST/PUT/GET pour un upload, authentifié

```
        _endpoint = [NSURLComponents componentsWithString:kADNBaseURL];
        NSURLSessionConfiguration *defaultConf =
            [NSURLSessionConfiguration defaultSessionConfiguration];
        defaultConf.HTTPAdditionalHeaders = @{
                                              @"Authorization" :
                                              @"Bearer __USER_TOKEN__"
                                              };

        _mainSession = [NSURLSession sessionWithConfiguration:defaultConf
                                                     delegate:delegate
                                                delegateQueue:nil];
```

Configuration de l’authentification à la création de la session

Cas d’utilisation 3

- Requêtes POST/PUT/GET pour un upload, authentifié

```
self.endpoint.path = @"/files";
NSMutableURLRequest *uploadRequest = [NSMutableURLRequest requestWithURL:self.endpoint.URL];
uploadRequest.HTTPMethod = @"POST";

NSString *formType = @"type=com.octiplex.spray;name=great_photo.jpg";
uploadRequest.HTTPBody = [formType dataUsingEncoding:NSUTF8StringEncoding];

NSURLSessionDataTask *dataTask =
    [self.mainSession dataTaskWithRequest:uploadRequest
                        completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {
                            NSHTTPURLResponse *httpResponse = (NSHTTPURLResponse*)response ;
                            if (httpResponse.statusCode == 200)
                            {
                                NSDictionary *jsonResponse = [NSJSONSerialization JSONObjectWithData:data
                                                                                             options:0
                                                                                               error:NULL];

                                NSString *fileID = jsonResponse[@"data"][@"id"];
                                [self uploadData:imageData fileID:fileID completionHandler:completion];
                            }
                            else { /* Gérer erreur */}
                        }];
[dataTask resume];
```

Création du fichier par une requête POST avec un champ form

Cas d’utilisation 3

- Requêtes POST/PUT/GET pour un upload, authentifié

```
self.endpoint.path = [NSString stringWithFormat:@"/files/%@/content",fileID];
NSMutableURLRequest *uploadRequest = [NSMutableURLRequest requestWithURL:self.endpoint.URL];
uploadRequest.HTTPMethod = @"PUT";
[uploadRequest setValue:@"image/jpeg" forHTTPHeaderField:@"Content-Type"];

NSURLSessionUploadTask *dataTask =
    [self.mainSession uploadTaskWithRequest:uploadRequest
                                   fromData:imageData // fromFile:(NSURL*)fileURL
                          completionHandler:^(NSData *data, NSURLResponse *response, NSError *error)
{
                              NSHTTPURLResponse *httpResponse = (NSHTTPURLResponse*)response ;
                              if (httpResponse.statusCode == 204)
                              {
                                  [self getFileURLForID:fileID completionHandler:completion];
                              }
                              else { /* Gérer erreur */ }
                          }];
[dataTask resume];
```

Envoi du contenu du fichier (PUT)

Cas d’utilisation 3

- Requêtes POST/PUT/GET pour un upload, authentifié

```
self.endpoint.path = [NSString stringWithFormat:@"/files/%@",fileID];
    [[self.mainSession dataTaskWithURL:self.endpoint.URL
       completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {
           NSHTTPURLResponse *httpResponse = (NSHTTPURLResponse*)response;

           if (httpResponse.statusCode == 200)
           {
               NSDictionary *jsonResponse = [NSJSONSerialization JSONObjectWithData:data
                                                                             options:0
                                                                               error:NULL];


               NSURL *fileURL = [NSURL URLWithString:jsonResponse[@"data"][@"url"]];

               completion(@{@"url":fileURL},nil);
           }
           else { /* Gérer erreur */ }
       }] resume];
```

Récupération du lien du fichier en GET

Démo

Points importants

- `NSURLSession` ne se base pas sur `NSURLConnection`

- Attention à l'utilisation de

```
+ (NSURLSession *)sharedSession;
```

- Propriété `delegateQueue` : `NSOperationQueue` de callback des delegates (lors de l'instanciation)

- `NSURLSessionDownloadTask` - Callback avec un fichier téléchargé éphémère

Points importants

- Pas forcément à choisir entre block based et delegate, on peut utiliser les deux à la fois
- Pour des transferts en background, on doit utiliser

```
+ (NSURLSessionConfiguration *)backgroundSessionConfiguration:
(NSString *)identifier;
```

Géré par un processus différent.
L'`identifier` sert à récupérer la hiérarchie (et les réponses) plus tard.

Points importants

- `allowsCellularAccess` : plus pratique que Reachability

- Une instance de `NSURLSession` invalidée n'est plus utilisable

- Jetez un coup d'oeil à `NSURLComponents` (pas de doc, mais dans `NSURL.h`)

En résumé

- Excellente couche d'abstraction plus haute que NSURLConnection, répond à une grande majorité de cas d'utilisations ;
- Relativement facile de bâtir une surcouche permettant de gérer plusieurs tâches, plusieurs sessions, timeouts, retry policy, cache…

Pour aller plus loin

- Application de démo et slides (bientôt) sur https://github.com/CocoaHeadsLyon
- Blog post plus étendu (bientôt) sur http://blog.octiplex.com
- Besoins réseau plus étendus ? AFNetworking 2.0

Merci !

Questions ?