Easy NSOperations

@raul_mpad

I WISH I HAD...

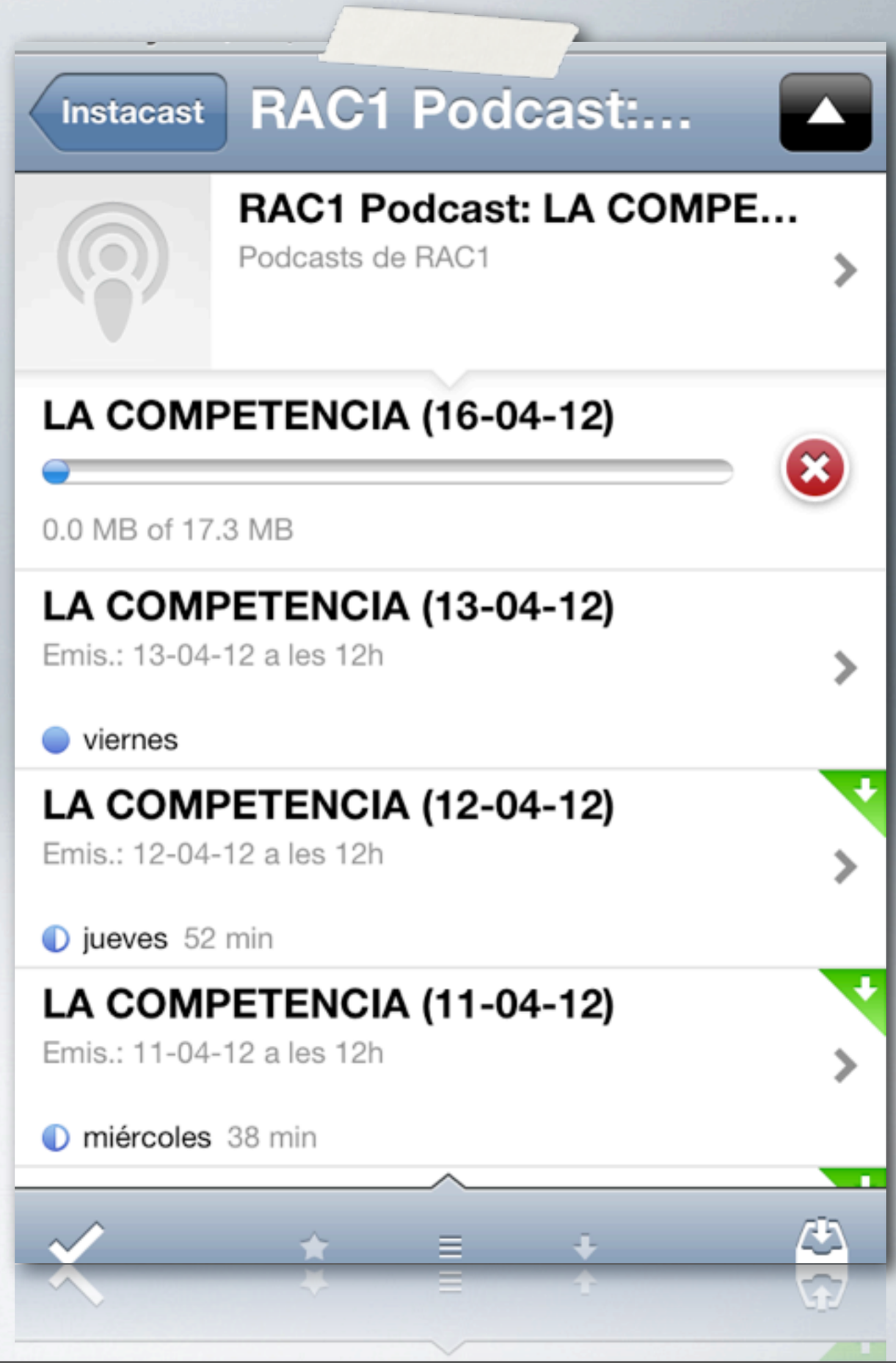
...better copying than pioneering..

I NEED...

Download content...

Store it...

Simultaneously...



So, Where do I start?

1. **Google**, of course!!
2. **StackOverflow**, am I a real developer or what??
3. **Apple** reference, the one and only...
4. **NSCodersBCN**, new kid on the block syndrome.

NSOperations & NSOperationQueue

"The NSOperation class is an abstract class you use to encapsulate the code and data associated with a single task"

"The NSOperationQueue class regulates the execution of a set of NSOperation objects"

So, what do I need?

- **NSOperation** to encapsulate and manage the download process.
- **NSOperationQueue** to attach multiple and simultaneous NSOperation objects
- **NSURLConnection** to establish http connections to the content
- Some **xib** representation of the process

Subclassing NSOperation

```
@interface RMPDownloadContentOperation : NSOperation {

  // Cell to uplooad status
  NSIndexPath *_indexPath;

  // Current status
  BOOL _finished;
  BOOL _executing;

  // Connection
  NSURLConnection *_conn;
  NSURL *_url;

  // To save to disk
  NSString *_filePath;
  NSOutputStream *_stream;

  // Progress values
  float _progressValue;
  long _downloadedContentLength;
  long _expectedContentLength;

  NSError * _error;
}
```

- @interface RMPDownloadContentOperation
  - progressValue
  - expectedContentValue
  - ✓ downloadedContentLength
  - indexPath
  - conn
  - url
  - filePath
  - stream
  - error
  - -initWithUrl:saveToFilePath:updatingCellAtRow:

- ✓ @implementation RMPDownloadContentOperation
  - progressValue
  - expectedContentValue
  - downloadedContentLength
  - indexPath
  - conn
  - url
  - filePath
  - stream
  - error
  - -initWithUrl:saveToFilePath:updatingCellAtRow:
  - -start
  - -done
  - -cancelled

  Delegate Methods for NSURLConnection
  - -connection:didReceiveResponse:
  - -connection:didFailWithError:
  - -connection:didReceiveData:
  - -connectionDidFinishLoading:
  - -connection:willCacheResponse:

Overriding NSOperation

- **start** method begins the execution of the operation.

```
- (void)start {
  // Ensure this operation is not being restarted and that it has not been cancelled
  if (![NSThread isMainThread]) {
    [self performSelectorOnMainThread:@selector(start)
                              withObject:nil waitUntilDone:NO];
    return;
  }

  if( _finished || [self isCancelled] ) {
//    [self done];
    return;
  }

  // KVO isExecuting
  [self willChangeValueForKey:@"isExecuting"];
  _executing = YES;
  [self didChangeValueForKey:@"isExecuting"];

  // Create the NSURLConnection
  self.conn = [[NSURLConnection alloc] initWithRequest:[NSURLRequest requestWithURL:self.url
                        cachePolicy:NSURLRequestReloadIgnoringCacheData
                        timeoutInterval:30.0] delegate:self];
}
```

- **done & cancelled** methods are handmade... and called by...

NSURLConnectionDelegate Protocol

```
@implementation RMPDownloadContentOperation
  P progressValue
  P expectedContentValue
  P downloadedContentLength
  P indexPath
  P conn
  P url
  P filePath
  P stream
  P error
  M -initWithUrl:saveToFilePath:updatingCellAtRow:
  M -start
  M -done
  M -cancelled
  Delegate Methods for NSURLConnection
✓ M -connection:didReceiveResponse:
  M -connection:didFailWithError:
  M -connection:didReceiveData:
  M -connectionDidFinishLoading:
  M -connection:willCacheResponse:
```

1. Open the **output stream** to save the content of the Response. We save the **content-length** of the Response for later calculations

2. If any error, report and mark as **done** the NSOperation object.

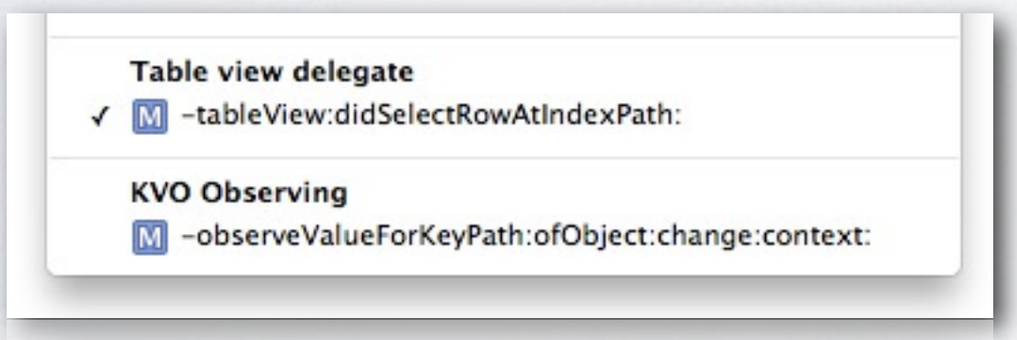
3. **Each time** we receive content from the connection we save it through the **stream** and update the **progress indicator** to the user

4. Whenever the download is done we update the NSOperation status with KVO.

Warning!! Protocol has **change**!!

NSOperation
KVO Pattern implementation

XxxViewController	NSOperation subclass
Add Observers & Observe	Notify changes




```
// KVO isFinished!
[self willChangeValueForKey:@"isExecuting"];
[self willChangeValueForKey:@"isFinished"];
_executing = NO;
_finished  = YES;
[self didChangeValueForKey:@"isFinished"];
[self didChangeValueForKey:@"isExecuting"];
```

Using NSOperationQueue

- Alloc and init, as usual...

```
// Create operation queue
self.operationQueue = [NSOperationQueue new];
// set maximum operations possible
[self.operationQueue setMaxConcurrentOperationCount:5];
```

- Queue NSOperations

```
RMPDownloadContentOperation *op = [[RMPDownloadContentOperation
alloc] initWithUrl:[self.urls objectAtIndex:0]

saveToFilePath: filePath

updatingCellAtRow:indexPath];
[self.operationQueue addOperation:op];
```

Progress indication

- User should know...

- And we must implement..

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
  static NSString *DefaultCellIdentifier = @"DefaultCell";
  static NSString *DownCellIdentifier = @"DownCell";

  if ([self.states objectAtIndex:[indexPath row]] == @"0") {
    RMPCustomCell *customCell = [tableView dequeueReusableCellWithIdentifier:DefaultCellIdentifier];
    customCell.lblTitle.text = [self.courses objectAtIndex:[indexPath row]];
    return customCell;
  } else {
    UITableViewCell *downCell = [tableView dequeueReusableCellWithIdentifier:DownCellIdentifier];
    return downCell;
  }
}
```

Demo

GOOGLIGRAPHY

- MUST HAVE: http://eng.pulse.me/concurrent-downloads-using-nsoperationqueues/
- NSOperations & GCD: http://stackoverflow.com/questions/4344884/what-tasks-are-more-suitable-to-nsoperation-than-gcd

Thank You!

Any easy question?