Best Practices for Progress Reporting

Session 232

Vince Spader Cocoa Frameworks Engineer
Agenda

Introduction
Composition
Cancellation, pausing, and resuming
User interface
Best practices
Introduction
Introduction

NSProgress represents the completion of some work
Introduction
Introduction
Introduction
Introduction

Makes it easy to report progress in your app across various components
Introduction

Makes it easy to report progress in your app across various components

Cocoa APIs are reporting their progress via NSProgress

• NSBundleResourceRequest
• UIDocument
• NSData
Introduction

Makes it easy to report progress in your app across various components

Cocoa APIs are reporting their progress via NSProgress

- NSBundleResourceRequest
- UIDocument
- NSData

Helps with localization
Introduction

var totalUnitCount: Int64
var completedUnitCount: Int64
var fractionCompleted: Double { get }
Units
Units

Bytes
Files
Photos
Percentage points
Fraction of work
Anything
var indeterminate: Bool { get }

Returns true if totalUnitCount < 0 or completedUnitCount < 0
Localization

```swift
var localizedDescription: String!
var localizedAdditionalDescription: String!
```
let progress = NSProgress()
progress.totalUnitCount = 5,312,764
progress.completedUnitCount = 419,240
let progress = NSProgress()
progress.totalUnitCount = 5_312_764
progress.completedUnitCount = 419_240
progress.kind = NSProgressKindFile

localizedDescription: Processing files...
localizedAdditionalDescription: 419 KB of 5.3 MB
Localization

userInfo

var userInfo: [NSObject : AnyObject] { get }
func setUserInfoObject(AnyObject?, forKey: String)
let progress = NSProgress()
progress.totalUnitCount = 5_312_764
progress.completedUnitCount = 419_240
progress.setUserInfoObject(97, forKey: NSProgressEstimatedTimeRemainingKey)

localizedDescription: 7% completed
localizedAdditionalDescription: 419,240 of 5,312,764 — About 1 minute, 37 seconds remaining
Localization

NSProgressKindFile

```swift
let progress = NSProgress()
progress.totalUnitCount = 5_312_764
progress.completedUnitCount = 419_240
progress.kind = NSProgressKindFile
progress.setUserInfoObject(NSProgressFileOperationKindDownloading,
    forKey: NSProgressFileOperationKindKey)
```

localizedDescription: **Downloading files...**

localizedAdditionalDescription: **419 KB of 5.3 MB**
let progress = NSProgress()
progress.totalUnitCount = 5_312_764
progress.completedUnitCount = 419_240
progress.kind = NSProgressKindFile
progress.setUserInfoObject(NSProgressFileOperationKindDownloading,
                          forKey: NSProgressFileOperationKindKey)
progress.setUserInfoObject(url, forKey: NSProgressFileURLKey)

localizedDescription: **Downloading “Photos.zip”...**
localizedAdditionalDescription: **419 KB of 5.3 MB**
let progress = NSProgress()
progress.totalUnitCount = 5_312_764
progress.completedUnitCount = 419_240
progress.kind = NSProgressKindFile
progress.setUserInfoObject(NSProgressFileOperationKindDownloading, forKey: NSProgressFileOperationKindKey)
progress.setUserInfoObject(7, forKey: NSProgressFileCompletedCountKey)
progress.setUserInfoObject(9, forKey: NSProgressFileTotalCountKey)
let progress = NSProgress()
progress.totalUnitCount = 5_312_764
progress.completedUnitCount = 419_240
progress.kind = NSProgressKindFile
progress.setUserInfoObject(NSProgressFileOperationKindDownloading, forKey: NSProgressFileOperationKindKey)
progress.setUserInfoObject(50443, forKey: NSProgressThroughputKey)

localizedDescription: Downloading files...
localizedAdditionalDescription: 419 KB of 5.3 MB (50 KB/sec)
Responsibilities

For creators

If you create a progress, you are responsible for updating it:

- totalUnitCount
- kind
- userInfo
- completedUnitCount
Responsibilities

For clients

If you receive a progress, do not update it

totalUnitCount { get }
completedUnitCount { get }
fractionCompleted { get }
localizedDescription { get }
localizedAdditionalDescription { get }
protocol NSProgressReporting : NSObjectProtocol {
    var progress: NSProgress { get }
}

NSProgressReporting
protocol NSProgressReporting : NSObjectProtocol {
    var progress: NSProgress { get }
}

NSProgressReporting
Demo

Introduction
Composition
Composition

What you’re tracking might not be a single operation’s progress
Composition

What you’re tracking might not be a single operation’s progress

Download
Composition

What you're tracking might not be a single operation’s progress

Download    Verify
What you’re tracking might not be a single operation’s progress
Composition

What you’re tracking might not be a single operation’s progress

Download  Verify  Decompress

But, the user only sees one progress bar
What you're tracking might not be a single operation's progress

But, the user only sees one progress bar
Composition

Download

Verify

Decompress
Composition

Overall

Download
Verify
Decompress
Composition

Portions of a parent’s totalUnitCount can be assigned to a child progress object, referred to as pendingUnitCount

- This is in terms of the parent’s units, not the child’s
- The parent’s pendingUnitCount is assigned to the child
Composition

When a child finishes, the parent’s completedUnitCount is incremented by the pendingUnitCount

- Do not update the completedUnitCount manually
- Assign everything to children
Composition
Composition

Import
Composition

Import
2 photos
Composition

Import
2 photos
Composition

Import 2 photos

1 of 2 photos

Photo 1

1 of 2 photos

Photo 2
Composition

1 of 2 photos

Photo 1
2 steps

Import
2 photos

1 of 2 photos

Photo 2
2 steps
Composition

Import 2 photos

1 of 2 photos

Photo 1
2 steps

1 of 2 photos

Photo 2
2 steps
Composition

1 of 2 photos

Import
2 photos

1 of 2 photos

Photo 1
2 steps

Download

1 of 2 steps

Filter

1 of 2 steps

Download

1 of 2 steps

Filter

1 of 2 steps

Photo 2
2 steps
Composition

Import 2 photos

1 of 2 photos

Photo 1
2 steps

Download 512 kilobytes

Photo 2
2 steps

Download 34 megabytes

Filter 7 stages

Filter 4 stages
Composition

Import
0 of 2 photos
0%

1 of 2 photos

Photo 1
0 of 2 steps
0%

Download
0 of 512 kilobytes
0%

1 of 2 steps

Filter
0 of 7 stages
0%

Photo 2
0 of 2 steps
0%

Download
0 of 34 megabytes
0%

1 of 2 steps

Filter
0 of 4 stages
0%
Composition

1 of 2 photos
   Photo 1
      1 of 2 steps
         Download
            512 of 512 kilobytes
               100%
   Photo 2
      1 of 2 steps
         Filter
            7 of 7 stages
               100%

1 of 2 photos
   1 of 2 steps
      Download
            34 of 34 megabytes
               100%
   1 of 2 steps
      Filter
            0 of 4 stages
               0%

Import
   1 of 2 photos
      1 of 2 steps
         100%
      75%
Composition

Import
2 of 2 photos
100%

Photo 1
2 of 2 steps
100%

Download
512 of 512 kilobytes
100%

Filter
7 of 7 stages
100%

Photo 2
2 of 2 steps
100%

Download
34 of 34 megabytes
100%

Filter
4 of 4 stages
100%
Composition

Weighting

1 of 2 steps
50%

Download
512 of 512 kilobytes
100%

1 of 2 steps

Filter
0 of 7 stages
0%
Composition

Weighting

1 of 2 steps

Download
512 of 512 kilobytes
100%

Photo
2 of 2 steps
100%

1 of 2 steps

Filter
7 of 7 stages
100%
Composition

Weighting

1 of 2 steps

Download

Photo
2 steps

1 of 2 steps

Filter
Composition

Weighting

Photo

Download → Filter
Composition
Weighting

Photo
10 “steps”

9 of 10 “steps”

Download

1 of 10 “steps”

Filter
Composition
Weighting

- Photo
  - 10 of 10 “steps”
  - 100%

- Download
  - 512 of 512 kilobytes
  - 100%

- Filter
  - 7 of 7 stages
  - 100%
Composition
Implicit
let photoProgress = NSProgress()
let photoProgress = NSProgress()
let photoProgress = NSProgress()
photoProgress.totalUnitCount = 2
Composition

Implicit

let photoProgress = NSProgress()
photoProgress.totalUnitCount = 2
photoProgress.becomeCurrentWithPendingUnitCount(1)
let photoProgress = NSProgress()
photoProgress.totalUnitCount = 2
photoProgress.becomeCurrentWithPendingUnitCount(1)
let photoProgress = NSProgress()
photoProgress.totalUnitCount = 2
photoProgress.becomeCurrentWithPendingUnitCount(1)
let photoProgress = NSProgress()
photoProgress.totalUnitCount = 2
photoProgress.becomeCurrentWithPendingUnitCount(1)
startDownload() // NSProgress(totalUnitCount:...)

Composition
Implicit

Photo
1 of 2 steps
let photoProgress = NSProgress()
photoProgress.totalUnitCount = 2
photoProgress.becomeCurrentWithPendingUnitCount(1)
startDownload() // NSProgress(totalUnitCount:...)

Composition
Implicit

1 of 2 steps
Download

Photo
2 steps

Current Progress
let photoProgress = NSProgress()
photoProgress.totalUnitCount = 2
photoProgress.becomeCurrentWithPendingUnitCount(1)
startDownload() // NSProgress(totalUnitCount:...)
photoProgress.resignCurrent()
let photoProgress = NSProgress()
photoProgress.totalUnitCount = 2
photoProgress.becomeCurrentWithPendingUnitCount(1)
startDownload() // NSProgress(totalUnitCount:…)
photoProgress.resignCurrent()
Composition

Implicit
Composition

Implicit

If you support implicit composition

• Create with NSProgress(totalUnitCount:) immediately
• Document it
Composition

Implicit

If you support implicit composition
• Create with NSProgress(totalUnitCount:) immediately
• Document it

If no child is added
• resignCurrent will mark the pendingUnitCount as finished
• The completedUnitCount will be updated
Composition
Explicit
let filterProgress = filter.progress
Composition
Explicit

let filterProgress = filter.progress
Composition
Explicit

let filterProgress = filter.progress
let photoProgress = ...
let filterProgress = filter.progress
let photoProgress = ...
Composition
Explicit

let filterProgress = filter.progress
let photoProgress = ...
photoProgress.addChild(filterProgress, withPendingUnitCount: 1)
Composition
Explicit

let filterProgress = filter.progress
let photoProgress = ...
photoProgress.addChild(filterProgress, withPendingUnitCount: 1)
Composition
Implicit vs. explicit

Use implicit composition if
• You have a method that can’t return the NSProgress
• Releases older than OS X10.11 and iOS 9

Otherwise, use explicit composition
Demo
Composition
Cancellation, Pausing, and Resuming
var cancellable: Bool
var cancellationHandler: (() -> Void)?
var cancelled: Bool { get }
func cancel()

• Sets cancelled to true
• Invokes the cancellationHandler
• Cancellation flows down to children
• It’s permanent
Pausing and Resuming

Creators

var pausable: Bool
var pausingHandler: (() -> Void)?
var resumingHandler: (() -> Void)?
var paused: Bool { get }
Pausing and Resuming

Clients

func pause()
func resume()

- Sets paused to true/false
- Invokes the pausingHandler/resumingHandler
- Pausing/resuming flows down to children
Demo
Cancellation, pausing, and resuming
User Interface
User Interface

NSProgress properties are key value observable

• Add KVO observers to update your UI
• Not necessarily called on main thread
User Interface

Example

progress.addObserver(self, forKeyPath: "fractionCompleted",
options: [], context: &observationContext)
User Interface

Example

```swift
progress.addObserver(self, forKeyPath: "fractionCompleted",
                      options: [], context: &observationContext)
```
User Interface

Example

progress.addObserver(self, forKeyPath: "fractionCompleted",
                      options: [], context: &observationContext)
progress.addObserver(self, forKeyPath: "fractionCompleted",
   options: [], context: &observationContext)

override func observeValueForKeyPath(keyPath: ..., object: ..., ..., context: ...) {
   if context == &observationContext && keyPath == "fractionCompleted" {
      NSOperationQueue.mainQueue().addOperationWithBlock {
         let progress = (object as! NSProgress)
         progressView.progress = Float(progress.fractionCompleted)
      }
   } else {
      super.observeValueForKeyPath(...)
   }
}
User Interface

Example

```swift
override func observeValueForKeyPath(keyPath: ..., object: ..., ..., context: ...) {
    if context == &observationContext && keyPath == "fractionCompleted" {
        NSOperationQueue.mainQueue().addOperationWithBlock {
            let progress = (object as! NSProgress)
            progressView.progress = Float(progress.fractionCompleted)
        }
    } else {
        super.observeValueForKeyPath("")
    }
}
```
User Interface

Example

```swift
progress.addObserver(self, forKeyPath: "fractionCompleted",
    options: [], context: &observationContext)

override func observeValueForKeyPath(keyPath: ..., object: ..., ..., context: ...) {
    if context == &observationContext && keyPath == "fractionCompleted" {
        NSOperationQueue.mainQueue().addOperationWithBlock {
            let progress = (object as! NSProgress)
            progressView.progress = Float(progress.fractionCompleted)
        }
    } else {
        super.observeValueForKeyPath(...) 
    }
}
```
User Interface

Example

```swift
progress.addObserver(self, forKeyPath: "fractionCompleted",
                      options: [], context: &observationContext)

override func observeValueForKeyPath(keyPath: ..., object: ..., ..., context: ...) {
    if context == &observationContext && keyPath == "fractionCompleted" {
        NSOperationQueue.mainQueue().addOperationWithBlock {
            let progress = (object as! NSProgress)
            progressView.progress = Float(progress.fractionCompleted)
        }
    }
    else {
        super.observeValueForKeyPath(...)
    }
}```
Best Practices
Best Practices

Don’t use fractionCompleted to determine completion

• It’s a float

• Use completedUnitCount >= totalUnitCount instead (unless indeterminate or zero)
Best Practices

NSProgress instances cannot be reused

Make a new instance and provide an additional mechanism so clients know
Performance

Don’t update completedUnitCount in a tight loop
Don’t forget that final update to 100%
Summary

Each NSProgress object has its own units
Summary

Each NSProgress object has its own units

You can compose NSProgress objects, either implicitly or explicitly
Summary

Each NSProgress object has its own units
You can compose NSProgress objects, either implicitly or explicitly
The pendingUnitCount is in the parent’s units
Summary

Each NSProgress object has its own units
You can compose NSProgress objects, either implicitly or explicitly
The pendingUnitCount is in the parent’s units
For each progress object, you are either a creator or a client
Summary

Each NSProgress object has its own units
You can compose NSProgress objects, either implicitly or explicitly
The pendingUnitCount is in the parent’s units
For each progress object, you are either a creator or a client
Use the kind and userInfo properties to let us give a good localizedDescription
Summary

Each NSProgress object has its own units
You can compose NSProgress objects, either implicitly or explicitly
The pendingUnitCount is in the parent’s units
For each progress object, you are either a creator or a client
Use the kind and userInfo properties to let us give a good localizedDescription
NSProgress can be a conduit for cancellation, pausing, and resuming work
Summary

Each NSProgress object has its own units
You can compose NSProgress objects, either implicitly or explicitly
The pendingUnitCount is in the parent’s units
For each progress object, you are either a creator or a client
Use the kind and userInfo properties to let us give a good localizedDescription
NSProgress can be a conduit for cancellation, pausing, and resuming work
Properties are KVO observable
More Information

Documentation
NSProgress Class Reference
Progress Indicator Programming Topics
iOS Human Interface Guidelines
OS X Human Interface Guidelines

Sample Code
PhotoProgress


Technical Support
Apple Developer Forums
Developer Technical Support

General Inquiries
Paul Marcos, App Frameworks Evangelist
pmarcos@apple.com