NSOperationQueue Class Reference

Cocoa > Process Management

2008-11-19

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

NSOperationQueue Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Threading Programming Guide
Declared in	NSOperation.h
Related sample code	NSOperationSample

Overview

The `NSOperationQueue` class manages a set of `NSOperation` objects in a priority queue and regulates their execution. Operations remain in the queue until they are explicitly cancelled or finish executing. An application may create multiple operation queues, with each queue running up to its designated maximum number of operations.

A specific `NSOperation` object can be in only one operation queue at a time. Operations within a single queue coordinate their execution order using both priority levels and inter-operation object dependencies. Operation objects in different queues can coordinate their execution order using dependencies, which are not queue-specific.

Inter-operation dependencies provide an absolute execution order for operations. An operation object is not considered ready to execute until all of its dependent operations have finished executing. For operations that are ready to execute, the operation queue always executes the one with the highest priority relative to the other ready operations. For details on how to set priority levels and dependencies, see *NSOperation Class Reference*.

You should never manually start an operation while it is sitting in an operation queue. Once added, an operation stays in its queue until it finishes executing or is cancelled.

If the `isConcurrent` method of an operation returns `NO`, the operation queue automatically creates a new thread for that operation before running it. If the `isConcurrent` method returns `YES`, the operation object must create its own thread or otherwise configure its own runtime environment as part of its execution phase.

KVO-Compliant Properties

The `NSOperationQueue` class is key-value coding (KVC) and key-value observing (KVO) compliant. You can observe these properties as desired to control other parts of your application. The properties you can observe include the following:

- `operations` - read-only property
- `maxConcurrentOperationCount` - readable and writable property

For more information about key-value observing and how to attach observers to an object, see *Key-Value Observing Programming Guide*.

Threading Considerations

It is safe to use a single `NSOperationQueue` object from multiple threads without creating additional locks to synchronize access to that object.

Tasks

Managing Operations in the Queue

- `addOperation:` (page 7)
  Adds the specified operation object to the receiver.
- `operations` (page 8)
  Returns a new array containing the operations currently in the queue.
- `cancelAllOperations` (page 7)
  Cancels all queued and executing operations.
- `waitUntilAllOperationsAreFinished` (page 10)
  Blocks the current thread until all of the receiver's queued and executing operations finish executing.

Managing the Number of Running Operations

- `maxConcurrentOperationCount` (page 8)
  Returns the maximum number of concurrent operations that the receiver can execute.
- `setMaxConcurrentOperationCount:` (page 9)
  Sets the maximum number of concurrent operations that the receiver can execute.

Suspending Operations

- `setSuspended:` (page 9)
  Modifies the execution of pending operations

- `isSuspended` (page 8)
  Returns a Boolean value indicating whether the receiver is scheduling queued operations for execution.

Instance Methods

addOperation:

Adds the specified operation object to the receiver.

```
- (void)addOperation:(NSOperation *)operation
```

Parameters

operation
  The operation object to be added to the queue. In memory-managed applications, this object is retained by the operation queue.

Discussion
An operation object can be in at most one operation queue at a time and cannot be added if it is currently executing or finished. This method throws an `NSInvalidArgumentException` exception if any of these conditions is true.

Once added, the specified *operation* remains in the queue until it is executed or cancelled.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `cancel` (NSOperation)
- `isExecuting` (NSOperation)

Declared In
`NSOperation.h`

cancelAllOperations

Cancels all queued and executing operations.

```
- (void)cancelAllOperations
```

Discussion
This method sends a `cancel` message to all operations currently in the queue or executing. Queued operations are cancelled before they begin executing. If an operation is already executing, it is up to that operation to recognize the cancellation and stop what it is doing.

Availability
Available in Mac OS X v10.5 and later.

See Also
`cancel` (`NSOperation`)

Declared In
NSOperation.h

isSuspended

Returns a Boolean value indicating whether the receiver is scheduling queued operations for execution.

```
- (BOOL)isSuspended
```

Return Value
NO if operations are being scheduled for execution; otherwise, YES.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setSuspended: (page 9)

Declared In
NSOperation.h

maxConcurrentOperationCount

Returns the maximum number of concurrent operations that the receiver can execute.

```
- (NSInteger)maxConcurrentOperationCount
```

Return Value
The maximum number of concurrent operations set explicitly on the receiver using the setMaxConcurrentOperationCount: method.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setMaxConcurrentOperationCount: (page 9)

Declared In
NSOperation.h

operations

Returns a new array containing the operations currently in the queue.

```
- (NSArray *)operations
```

Return Value
A new array object containing the NSOperation objects in the order in which they were added to the queue.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSOperation.h`

setMaxConcurrentOperationCount:

Sets the maximum number of concurrent operations that the receiver can execute.

```
- (void)setMaxConcurrentOperationCount:(NSInteger)count
```

Parameters

count

The maximum number of concurrent operations. Specify the value `NSOperationQueueDefaultMaxConcurrentOperationCount` if you want the receiver to choose an appropriate value based on the number of available processors and other relevant factors.

Discussion
The specified value affects only the receiver and the operations in its queue. Other operation queue objects can also execute their maximum number of operations in parallel.

Reducing the number of concurrent operations does not affect any operations that are currently executing. If you specify the value `NSOperationQueueDefaultMaxConcurrentOperationCount` (which is recommended), the maximum number of operations can change dynamically based on system conditions.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- maxConcurrentOperationCount` (page 8)

Declared In
`NSOperation.h`

setSuspended:

Modifies the execution of pending operations

```
- (void)setSuspended:(BOOL)suspend
```

Parameters

suspend

If `YES`, the queue stops scheduling queued operations for execution. If `NO`, the queue begins scheduling operations again.

Discussion
This method suspends or restarts the execution of queued operations only. It does not have any impact on the state of currently running operations. Running operations continue to run until their natural termination or until they are explicitly cancelled.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- isSuspended` (page 8)

Declared In
`NSOperation.h`

waitUntilAllOperationsAreFinished

Blocks the current thread until all of the receiver's queued and executing operations finish executing.

```
- (void)waitUntilAllOperationsAreFinished
```

Discussion
When called, this method blocks the current thread and waits for the receiver's current and pending operations to finish executing. While the thread is blocked, the receiver continues to launch already queued operations and monitor those that are executing. During this time, the current thread cannot add operations to the queue, but other threads may. Once all of the pending operations are finished, this method returns.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSOperation.h`

Constants

Concurrent Operation Constants

Indicates the number of supported concurrent operations.

```
enum {
    NSOperationQueueDefaultMaxConcurrentOperationCount = -1
};
```

Constants
`NSOperationQueueDefaultMaxConcurrentOperationCount`
The default maximum number of operations is determined dynamically by the `NSOperationQueue` object based on current system conditions.

Available in Mac OS X v10.5 and later.

Declared in `NSOperation.h`.

Declared In
`NSOperation.h`

Document Revision History

This table describes the changes to *NSOperationQueue Class Reference*.

Date	Notes
2008-11-19	Updated the guidance related to KVO-compliant properties.
2008-10-15	Clarified ownership of operation objects when added to a queue.
2007-04-30	New document describing the methods for managing operation objects.

Document Revision History

Index