NSGarbageCollector Class Reference

Cocoa > Objective-C Language

2008-10-15

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

Simultaneously published in the United States and Canada.

Contents

NSGarbageCollector Class Reference 5

Overview 5
Tasks 6
- Shared Instance 6
- Collection State 6
- Triggering Collection 6
- Manipulating External References 6
- Accessing an Unscanned Memory Zone 6

Class Methods 7
- defaultCollector 7

Instance Methods 7
- collectExhaustively 7
- collectIfNeeded 7
- disable 8
- disableCollectorForPointer: 8
- enable 9
- enableCollectorForPointer: 9
- isCollecting 10
- isEnabled 10
- zone 10

Document Revision History 13

Index 15

NSGarbageCollector 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	Garbage Collection Programming Guide
Declared in	NSGarbageCollector.h

Overview

`NSGarbageCollector` provides a convenient interface to the garbage collection system.

Cocoa's garbage collector is a conservative generational garbage collector. It uses "write-barriers" to detect cross generational stores of pointers so that "young" objects can be collected quickly.

You enable garbage collection (GC) by using the `-fobjc-gc compiler` option. This switch causes the generation of the write-barrier assignment primitives. You must use this option on your main application file *and all others used by the application*, including frameworks and bundles. Bundles are ignored if they are not GC-capable.

The collector determines what is garbage by recursively examining all nodes starting with globals, possible nodes referenced from the thread stacks, and all nodes marked as having "external" references. Nodes not reached by this search are deemed garbage. Weak references to garbage nodes are then cleared.

Garbage nodes that are objects are sent (in an arbitrary order) a `finalize` message, and after all `finalize` messages have been sent their memory is recovered. It is a runtime error (referred to as "resurrection") to store a object being finalized into one that is not. For more details, see Implementing a finalize Method in *Garbage Collection Programming Guide*.

You can request collection from any thread (see `collectIfNeeded` (page 7) and `collectExhaustively` (page 7)).

Tasks

Shared Instance

`+ defaultCollector` (page 7)
Returns the default garbage collector.

Collection State

`- disable` (page 8)
Temporarily disables collections.

`- enable` (page 9)
Enables collection after collection has been disabled.

`- isEnabled` (page 10)
Returns a Boolean value that indicates whether garbage collection is currently enabled for the current process.

`- isCollecting` (page 10)
Returns a Boolean value that indicates whether a collection is currently in progress.

Triggering Collection

`- collectExhaustively` (page 7)
Tells the receiver to collect iteratively.

`- collectIfNeeded` (page 7)
Tells the receiver to collect if memory consumption thresholds have been exceeded.

Manipulating External References

`- disableCollectorForPointer:` (page 8)
Specifies that a given pointer will not be collected.

`- enableCollectorForPointer:` (page 9)
Specifies that a given pointer may be collected.

Accessing an Unscanned Memory Zone

`- zone` (page 10)
Returns a zone of unscanned memory.

Class Methods

defaultCollector

Returns the default garbage collector.

```
+ (id)defaultCollector
```

Return Value
The default garbage collector for the current process. Returns `nil` if the current process is not running with garbage collection.

Discussion
There is at most one garbage collector for Cocoa within a single process.

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

Declared In
`NSGarbageCollector.h`

Instance Methods

collectExhaustively

Tells the receiver to collect iteratively.

```
- (void)collectExhaustively
```

Discussion
You use this method to indicate to the collector that it should perform an exhaustive collection. Collection is subject to interruption on user input.

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

Declared In
`NSGarbageCollector.h`

collectIfNeeded

Tells the receiver to collect if memory consumption thresholds have been exceeded.

```
- (void)collectIfNeeded
```

Discussion
You use this method to indicate to the collector that there is an opportunity to perform a collection. Collection is subject to interruption on user input.

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

Declared In
`NSGarbageCollector.h`

disable

Temporarily disables collections.

```
- (void)disable
```

Discussion
Invocations of this method can be nested. To reenable collection, you must send the collector an `enable` (page 9) message once for each invocation of this method.

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

See Also
- `enable` (page 9)

Declared In
`NSGarbageCollector.h`

disableCollectorForPointer:

Specifies that a given pointer will not be collected.

```
- (void)disableCollectorForPointer:(void *)ptr
```

Parameters

ptr

A pointer to the memory that should not be collected.

Discussion
You use this method to ensure that memory at a given address will not be collected. You can use this, for example, to create new root objects:

```
NSMutableDictionary *globalDictionary;
globalDictionary = [NSMutableDictionary dictionary];
[[NSGarbageCollector defaultCollector]
    disableCollectorForPointer:globalDictionary];
```

The new dictionary will not be collectable and will persist for the lifetime of the application unless it is subsequently passed as the argument to `enableCollectorForPointer:` (page 9). For more about root objects and scanned memory, see *Garbage Collection Programming Guide*.

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

See Also
- `enableCollectorForPointer:` (page 9)

Declared In
NSGarbageCollector.h

enable

Enables collection after collection has been disabled.

```
- (void)enable
```

Discussion
This method balances a single invocation of `disable` (page 8). To reenable collection, this method must be invoked as many times as was `disable` (page 8).

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

See Also
- `disable` (page 8)
- `isEnabled` (page 10)

Declared In
NSGarbageCollector.h

enableCollectorForPointer:

Specifies that a given pointer may be collected.

```
- (void)enableCollectorForPointer:(void *)ptr
```

Parameters
ptr
A pointer to the memory that may be collected.

Discussion
You use this method to make memory that was previously marked as uncollectable. For example, given the address of the global dictionary created in `disableCollectorForPointer:` (page 8), you could make the dictionary collectable as follows:

```
[[NSGarbageCollector defaultCollector]
    enableCollectorForPointer:globalDictionary];
```

For more about root objects and scanned memory, see *Garbage Collection Programming Guide*.

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

See Also
- `disableCollectorForPointer:` (page 8)

Declared In
NSGarbageCollector.h

isCollecting

Returns a Boolean value that indicates whether a collection is currently in progress.

```
- (BOOL)isCollecting
```

Return Value
`YES` if a collection is currently in progress, otherwise `NO`.

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

Declared In
`NSGarbageCollector.h`

isEnabled

Returns a Boolean value that indicates whether garbage collection is currently enabled for the current process.

```
- (BOOL)isEnabled
```

Return Value
`YES` if garbage collection is enabled for the current process, otherwise `NO`.

Discussion
This method returns `NO` if garbage collection is on, but has been temporarily suspended (using `disable` (page 8)).

To check whether the current process is using garbage collection check the result of `[NSGarbageCollector defaultCollector]`. If `defaultCollector` (page 7) is `nil`, then garbage collection is permanently off. If `defaultCollector` (page 7) is not `nil`, then the current process is using garbage collection—you can then use `isEnabled` to determine whether or not the collector is actually allowed to run right now.

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

See Also
- `disable` (page 8)
- `enable` (page 9)

Declared In
`NSGarbageCollector.h`

zone

Returns a zone of unscanned memory.

```
- (NSZone *)zone
```

Return Value
A memory zone of memory that is not scanned.

Discussion
The collector provides a `NSZoneMalloc`-style allocation interface, primarily for compatibility with existing code that maintains zone affinity. Such memory is unscanned and you must free it using `NSZoneFree`. This is exactly equivalent to calling `NSAllocateCollectable` with the option `NSCollectorDisabledOption` (page ?).

You should typically allocate garbage-collected memory using `NSAllocateCollectable`.

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

Declared In
`NSGarbageCollector.h`

Document Revision History

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

Date	Notes
2008-10-15	Clarified use of -isEnabled.
2008-06-09	Updated discussion of disableCollectorForPointer:.
2006-12-19	New document that describes the Cocoa class used to interact with the garbage collection system.

Document Revision History

Index

C

`collectExhaustively` instance method 7
`collectIfNeeded` instance method 7

D

`defaultCollector` class method 7
`disable` instance method 8
`disableCollectorForPointer:` instance method 8

E

`enable` instance method 9
`enableCollectorForPointer:` instance method 9

I

`isCollecting` instance method 10
`isEnabled` instance method 10

Z

`zone` instance method 10