
NSAutoreleasePool Class Reference





Apple Inc.
© 2009 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

iPhone is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR

CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSAutoreleasePool Class Reference 5

Overview	5
Threads	6
Garbage Collection	6
Tasks	6
Managing a Pool	6
Adding an Object to a Pool	7
Class Methods	7
addObject:	7
Instance Methods	8
addObject:	8
autorelease	8
drain	8
release	9
retain	9

Document Revision History 11

Index 13

NSAutoreleasePool Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Memory Management Programming Guide for Cocoa
Declared in	NSAutoreleasePool.h
Related sample code	CocoaSpeechSynthesisExample OpenCL NBody Simulation Example SpellingChecker CarbonCocoa Bundled SpellingChecker-CarbonCocoa SpellingChecker-CocoaCarbon

Overview

The `NSAutoreleasePool` class is used to support Cocoa's reference-counted memory management system. An autorelease pool stores objects that are sent a `release` message when the pool itself is drained.

In a reference-counted environment (as opposed to one which uses garbage collection), an `NSAutoreleasePool` object contains objects that have received an `autorelease` message and when drained it sends a `release` message to each of those objects. Thus, sending `autorelease` instead of `release` to an object extends the lifetime of that object at least until the pool itself is drained (it may be longer if the object is subsequently retained). An object can be put into the same pool several times, in which case it receives a `release` message for each time it was put into the pool.

In a reference counted environment, Cocoa expects there to be an autorelease pool always available. If a pool is not available, autoreleased objects do not get released and you leak memory. In this situation, your program will typically log suitable warning messages.

The Application Kit creates an autorelease pool on the main thread at the beginning of every cycle of the event loop, and drains it at the end, thereby releasing any autoreleased objects generated while processing an event. If you use the Application Kit, you therefore typically don't have to create your own pools. If your application creates a lot of temporary autoreleased objects within the event loop, however, it may be beneficial to create "local" autorelease pools to help to minimize the peak memory footprint.

You create an `NSAutoreleasePool` object with the usual `alloc` and `init` messages and dispose of it with `drain` (page 8) (or `release` (page 9)—to understand the difference, see "[Garbage Collection](#)" (page 6)). Since you cannot retain an autorelease pool (or autorelease it—see `retain` (page 9) and

[autorelease](#) (page 8)), draining a pool ultimately has the effect of deallocating it. You should always drain an autorelease pool in the same context (invocation of a method or function, or body of a loop) that it was created. See Autorelease Pools for more details.

Each thread (including the main thread) maintains its own stack of `NSAutoreleasePool` objects (see “Threads” (page 6)). As new pools are created, they get added to the top of the stack. When pools are deallocated, they are removed from the stack. Autoreleased objects are placed into the top autorelease pool for the current thread. When a thread terminates, it automatically drains all of the autorelease pools associated with itself.

Threads

If you are making Cocoa calls outside of the Application Kit’s main thread—for example if you create a Foundation-only application or if you detach a thread—you need to create your own autorelease pool.

If your application or thread is long-lived and potentially generates a lot of autoreleased objects, you should periodically drain and create autorelease pools (like the Application Kit does on the main thread); otherwise, autoreleased objects accumulate and your memory footprint grows. If, however, your detached thread does not make Cocoa calls, you do not need to create an autorelease pool.

Note: If you are creating secondary threads using the POSIX thread APIs instead of `NSThread` objects, you cannot use Cocoa, including `NSAutoreleasePool`, unless Cocoa is in multithreading mode. Cocoa enters multithreading mode only after detaching its first `NSThread` object. To use Cocoa on secondary POSIX threads, your application must first detach at least one `NSThread` object, which can immediately exit. You can test whether Cocoa is in multithreading mode with the `NSThread` class method `isMultiThreaded`.

Garbage Collection

In a garbage-collected environment, there is no need for autorelease pools. You may, however, write a framework that is designed to work in both a garbage-collected and reference-counted environment. In this case, you can use autorelease pools to hint to the collector that collection may be appropriate. In a garbage-collected environment, sending a [drain](#) (page 8) message to a pool triggers garbage collection if necessary; [release](#) (page 9), however, is a no-op. In a reference-counted environment, [drain](#) (page 8) has the same effect as [release](#) (page 9). Typically, therefore, you should use [drain](#) (page 8) instead of [release](#) (page 9).

Tasks

Managing a Pool

- [release](#) (page 9)
Releases and pops the receiver.

- [drain](#) (page 8)
In a reference-counted environment, releases and pops the receiver; in a garbage-collected environment, triggers garbage collection if the memory allocated since the last collection is greater than the current threshold.
- [autorelease](#) (page 8)
Raises an exception.
- [retain](#) (page 9)
Raises an exception.

Adding an Object to a Pool

- + [addObject:](#) (page 7)
Adds a given object to the active autorelease pool in the current thread.
- [addObject:](#) (page 8)
Adds a given object to the receiver

Class Methods

addObject:

Adds a given object to the active autorelease pool in the current thread.

```
+ (void)addObject:(id)object
```

Parameters

object

The object to add to the active autorelease pool in the current thread.

Discussion

The same object may be added several times to the active pool and, when the pool is deallocated, it will receive a `release` message for each time it was added.

Normally you don't invoke this method directly—you send `autorelease` to *object* instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [addObject:](#) (page 8)

Declared In

NSAutoreleasePool.h

Instance Methods

addObject:

Adds a given object to the receiver

```
- (void)addObject:(id)object
```

Parameters

object

The object to add to the receiver.

Discussion

The same object may be added several times to the same pool; when the pool is deallocated, the object will receive a `release` message for each time it was added.

Normally you don't invoke this method directly—you send `autorelease` to *object* instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [addObject:](#) (page 7)

Declared In

NSAutoreleasePool.h

autorelease

Raises an exception.

```
- (id)autorelease
```

Return Value

`self`.

Discussion

In a reference-counted environment, this method raises an exception.

drain

In a reference-counted environment, releases and pops the receiver; in a garbage-collected environment, triggers garbage collection if the memory allocated since the last collection is greater than the current threshold.

```
- (void)drain
```

Discussion

In a reference-counted environment, this method behaves the same as `release`. Since an autorelease pool cannot be retained (see [retain](#) (page 9)), this therefore causes the receiver to be deallocated. When an autorelease pool is deallocated, it sends a `release` message to all its autoreleased objects. If an object is added several times to the same pool, when the pool is deallocated it receives a `release` message for each time it was added.

In a garbage-collected environment, this method ultimately calls `objc_collect_if_needed`.

Special Considerations

In a garbage-collected environment, `release` is a no-op, so unless you do not want to give the collector a hint it is important to use `drain` in any code that may be compiled for a garbage-collected environment.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

Core Data HTML Store
From A View to A Movie
GLUT
PortMapper
VideoViewer

Declared In

NSAutoreleasePool.h

release

Releases and pops the receiver.

- (void)release

Discussion

In a reference-counted environment, since an autorelease pool cannot be retained (see [retain](#) (page 9)), this method causes the receiver to be deallocated. When an autorelease pool is deallocated, it sends a `release` message to all its autoreleased objects. If an object is added several times to the same pool, when the pool is deallocated it receives a `release` message for each time it was added.

In a garbage-collected environment, this method is a no-op.

Special Considerations

You should typically use [drain](#) (page 8) instead of `release`.

See Also

- [drain](#) (page 8)

retain

Raises an exception.

- (id)retain

Return Value

self.

Discussion

In a reference-counted environment, this method raises an exception.

Document Revision History

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

Date	Notes
2009-01-02	Changed "managed memory environment" to "reference-counted environment" throughout.
2008-02-08	Corrected typographical errors.
2007-10-31	Clarified the discussion of drain.
2007-02-07	Updated for Mac OS X v10.5.
2006-05-23	Made editorial corrections.
	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

addObject: class method [7](#)
addObject: instance method [8](#)
autorelease instance method [8](#)

D

drain instance method [8](#)

R

release instance method [9](#)
retain instance method [9](#)