
NSInvocation Class Reference

Data Management: Event Handling



2009-07-04



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, Mac OS, and Objective-C are trademarks of Apple Inc., registered in the United States and other countries.

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

NSInvocation Class Reference 5

Overview	5
Adopted Protocols	6
Tasks	6
Creating NSInvocation Objects	6
Configuring an Invocation Object	6
Dispatching an Invocation	7
Getting the Method Signature	7
Class Methods	7
invocationWithMethodSignature:	7
Instance Methods	7
argumentsRetained	7
getArgumentAtIndex:	8
getReturnValue:	9
invoke	9
invokeWithTarget:	10
methodSignature	10
retainArguments	11
selector	11
setArgumentAtIndex:	11
setReturnValue:	12
setSelector:	13
setTarget:	13
target	14

Document Revision History 15

Index 17

NSInvocation 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	Distributed Objects Programming Topics
Declared in	NSInvocation.h
Related sample code	CubePuzzle DeskPictAppDockMenu ForwardInvocation

Overview

An `NSInvocation` is an Objective-C message rendered static, that is, it is an action turned into an object. `NSInvocation` objects are used to store and forward messages between objects and between applications, primarily by `NSTimer` objects and the distributed objects system.

An `NSInvocation` object contains all the elements of an Objective-C message: a target, a selector, arguments, and the return value. Each of these elements can be set directly, and the return value is set automatically when the `NSInvocation` object is dispatched.

An `NSInvocation` object can be repeatedly dispatched to different targets; its arguments can be modified between dispatch for varying results; even its selector can be changed to another with the same method signature (argument and return types). This flexibility makes `NSInvocation` useful for repeating messages with many arguments and variations; rather than retyping a slightly different expression for each message, you modify the `NSInvocation` object as needed each time before dispatching it to a new target.

`NSInvocation` does not support invocations of methods with either variable numbers of arguments or union arguments. You should use the [invocationWithMethodSignature:](#) (page 7) class method to create `NSInvocation` objects; you should not create these objects using `alloc` and `init`.

This class does not retain the arguments for the contained invocation by default. If those objects might disappear between the time you create your instance of `NSInvocation` and the time you use it, you should explicitly retain the objects yourself or invoke the `retainArguments` method to have the invocation object retain them itself.

Note: `NSInvocation` conforms to the `NSCoding` protocol, but only supports coding by an `NSPortCoder`. `NSInvocation` does not support archiving.

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Creating NSInvocation Objects

- + `invocationWithMethodSignature:` (page 7)
Returns an `NSInvocation` object able to construct messages using a given method signature.

Configuring an Invocation Object

- `setSelector:` (page 13)
Sets the receiver's selector.
- `selector` (page 11)
Returns the receiver's selector, or 0 if it hasn't been set.
- `setTarget:` (page 13)
Sets the receiver's target.
- `target` (page 14)
Returns the receiver's target, or `nil` if the receiver has no target.
- `setArgumentAtIndex:` (page 11)
Sets an argument of the receiver.
- `getArgumentAtIndex:` (page 8)
Returns by indirection the receiver's argument at a specified index.
- `argumentsRetained` (page 7)
Returns `YES` if the receiver has retained its arguments, `NO` otherwise.
- `retainArguments` (page 11)
If the receiver hasn't already done so, retains the target and all object arguments of the receiver and copies all of its C-string arguments.
- `setReturnValue:` (page 12)
Sets the receiver's return value.
- `getReturnValue:` (page 9)
Gets the receiver's return value.

Dispatching an Invocation

- [invoke](#) (page 9)
Sends the receiver's message (with arguments) to its target and sets the return value.
- [invokeWithTarget:](#) (page 10)
Sets the receiver's target, sends the receiver's message (with arguments) to that target, and sets the return value.

Getting the Method Signature

- [methodSignature](#) (page 10)
Returns the receiver's method signature.

Class Methods

invocationWithMethodSignature:

Returns an `NSInvocation` object able to construct messages using a given method signature.

```
+ (NSInvocation *)invocationWithMethodSignature:(NSMethodSignature *)signature
```

Parameters

signature

An object encapsulating a method signature.

Discussion

The new object must have its selector set with [setSelector:](#) (page 13) and its arguments set with [setArgumentAtIndex:](#) (page 11) before it can be invoked. Do not use the `alloc/init` approach to create `NSInvocation` objects.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CubePuzzle

DeskPictAppDockMenu

Declared In

`NSInvocation.h`

Instance Methods

argumentsRetained

Returns YES if the receiver has retained its arguments, NO otherwise.

- (BOOL)argumentsRetained

Availability

Available in Mac OS X v10.0 and later.

See Also

- [retainArguments](#) (page 11)

Declared In

NSInvocation.h

getArgumentAtIndex:

Returns by indirection the receiver's argument at a specified index.

```
- (void)getArgument:(void *)buffer atIndex:(NSInteger)index
```

Parameters

buffer

An untyped buffer to hold the returned argument. See the discussion below relating to argument values that are objects.

index

An integer specifying the index of the argument to get.

Indices 0 and 1 indicate the hidden arguments *self* and *_cmd*, respectively; these values can be retrieved directly with the `target` and `selector` methods. Use indices 2 and greater for the arguments normally passed in a message.

Discussion

This method copies the argument stored at *index* into the storage pointed to by *buffer*. The size of *buffer* must be large enough to accommodate the argument value.

When the argument value is an object, pass a pointer to the variable (or memory) into which the object should be placed:

```
NSArray *anArray;
[invocation getArgument:&anArray atIndex:3];
```

This method raises `NSInvalidArgumentException` if *index* is greater than the actual number of arguments for the selector.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setArgumentAtIndex:](#) (page 11)
- `numberOfArguments` (NSMethodSignature)

Declared In

NSInvocation.h

getReturnValue:

Gets the receiver's return value.

- (void)getReturnValue:(void *)*buffer*

Parameters

buffer

An untyped buffer into which the receiver copies its return value. It should be large enough to accommodate the value. See the discussion below for more information about *buffer*.

Discussion

Use the `NSMethodSignature` method `methodReturnLength` to determine the size needed for *buffer*:

```
NSUInteger length = [[myInvocation methodSignature] methodReturnLength];
buffer = (void *)malloc(length);
[invocation getReturnValue:buffer];
```

When the return value is an object, pass a pointer to the variable (or memory) into which the object should be placed:

```
id anObject;
NSArray *anArray;
[invocation1 getReturnValue:&anObject];
[invocation2 getReturnValue:&anArray];
```

If the `NSInvocation` object has never been invoked, the result of this method is undefined.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setReturnValue:](#) (page 12)
- `methodReturnType` (`NSMethodSignature`)

Related Sample Code

CubePuzzle

Declared In

`NSInvocation.h`

invoke

Sends the receiver's message (with arguments) to its target and sets the return value.

- (void)invoke

Discussion

You must set the receiver's target, selector, and argument values before calling this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [getReturnValue:](#) (page 9)

- [setSelector:](#) (page 13)
- [setTarget:](#) (page 13)
- [setArgumentAtIndex:](#) (page 11)

Related Sample Code

CubePuzzle

Declared In

NSInvocation.h

invokeWithTarget:

Sets the receiver's target, sends the receiver's message (with arguments) to that target, and sets the return value.

```
- (void)invokeWithTarget:(id)anObject
```

Parameters*anObject*

The object to set as the receiver's target.

Discussion

You must set the receiver's selector and argument values before calling this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [getReturnValue:](#) (page 9)
- [invoke](#) (page 9)
- [setSelector:](#) (page 13)
- [setTarget:](#) (page 13)
- [setArgumentAtIndex:](#) (page 11)

Related Sample Code

ForwardInvocation

Declared In

NSInvocation.h

methodSignature

Returns the receiver's method signature.

```
- (NSMethodSignature *)methodSignature
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSInvocation.h

retainArguments

If the receiver hasn't already done so, retains the target and all object arguments of the receiver and copies all of its C-string arguments.

- (void)retainArguments

Discussion

Before this method is invoked, [argumentsRetained](#) (page 7) returns NO; after, it returns YES.

For efficiency, newly created NSInvocations don't retain or copy their arguments, nor do they retain their targets or copy C strings. You should instruct an NSInvocation to retain its arguments if you intend to cache it, since the arguments may otherwise be released before the NSInvocation is invoked. NSTimers always instruct their NSInvocations to retain their arguments, for example, because there's usually a delay before an NSTimer fires.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSInvocation.h

selector

Returns the receiver's selector, or 0 if it hasn't been set.

- (SEL)selector

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setSelector:](#) (page 13)

Related Sample Code

ForwardInvocation

Declared In

NSInvocation.h

setArgument:atIndex:

Sets an argument of the receiver.

- (void)setArgument:(void *)*buffer* atIndex:(NSInteger)*index*

Parameters

buffer

An untyped buffer containing an argument to be assigned to the receiver. See the discussion below relating to argument values that are objects.

index

An integer specifying the index of the argument.

Indices 0 and 1 indicate the hidden arguments *self* and *_cmd*, respectively; you should set these values directly with the [setTarget:](#) (page 13) and [setSelector:](#) (page 13) methods. Use indices 2 and greater for the arguments normally passed in a message.

Discussion

This method copies the contents of *buffer* as the argument at *index*. The number of bytes copied is determined by the argument size.

When the argument value is an object, pass a pointer to the variable (or memory) from which the object should be copied:

```
NSArray *anArray;
[invocation setArgument:&anArray atIndex:3];
```

This method raises `NSInvalidArgumentException` if the value of *index* is greater than the actual number of arguments for the selector.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [getArgument:atIndex:](#) (page 8)
- `numberOfArguments` (NSMethodSignature)

Related Sample Code

CubePuzzle

DeskPictAppDockMenu

Declared In

NSInvocation.h

setReturnValue:

Sets the receiver's return value.

```
- (void)setReturnValue:(void *)buffer
```

Parameters

buffer

An untyped buffer whose contents are copied as the receiver's return value.

Discussion

This value is normally set when you send an [invoke](#) (page 9) or [invokeWithTarget:](#) (page 10) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [getReturnValue:](#) (page 9)
- `methodReturnLength` (NSMethodSignature)
- `methodReturnType` (NSMethodSignature)

Declared In

NSInvocation.h

setSelector:

Sets the receiver's selector.

- (void)setSelector:(SEL)selector

Parameters

selector

The selector to assign to the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [selector](#) (page 11)

Related Sample Code

CubePuzzle

DeskPictAppDockMenu

Declared In

NSInvocation.h

setTarget:

Sets the receiver's target.

- (void)setTarget:(id)anObject

Parameters

anObject

The object to assign to the receiver as target. The target is the receiver of the message sent by [invoke](#) (page 9).

Discussion

Availability

Available in Mac OS X v10.0 and later.

See Also

- [target](#) (page 14)

- [invokeWithTarget:](#) (page 10)

Related Sample Code

CubePuzzle

DeskPictAppDockMenu

Declared In

NSInvocation.h

target

Returns the receiver's target, or `nil` if the receiver has no target.

- (id)target

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setTarget:](#) (page 13)

Declared In

NSInvocation.h

Document Revision History

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

Date	Notes
2009-07-04	Removed obsolete NSObjCValueType.
2008-10-15	Updated code examples to use NSInteger instead of raw types.
2007-04-19	Updated for Mac OS X version 10.5 and moved recent changes in v10.4 version.
2006-11-07	Added caveat about using alloc/init to create NSInvocation objects.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

argumentsRetained [instance method 7](#)

G

getArgument:atIndex: [instance method 8](#)
getReturnValue: [instance method 9](#)

I

invocationWithMethodSignature: [class method 7](#)
invoke [instance method 9](#)
invokeWithTarget: [instance method 10](#)

M

methodSignature [instance method 10](#)

R

retainArguments [instance method 11](#)

S

selector [instance method 11](#)
setArgument:atIndex: [instance method 11](#)
setReturnValue: [instance method 12](#)
setSelector: [instance method 13](#)
setTarget: [instance method 13](#)

T

target [instance method 14](#)