
NSSet Class Reference

Data Management: Data Types & Collections



2009-08-18



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.

OpenGL is a registered trademark of Silicon Graphics, 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

NSSet Class Reference 5

Overview	5
Adopted Protocols	6
Tasks	7
Creating a Set	7
Initializing a Set	7
Counting Entries	8
Accessing Set Members	8
Comparing Sets	8
Creating a Sorted Array	9
Key-Value Observing	9
Describing a Set	9
Class Methods	9
set	9
initWithArray:	10
initWithObject:	10
initWithObjects:	11
initWithObjects:count:	12
initWithSet:	13
Instance Methods	13
addObserver:forKeyPath:options:context:	13
allObjects	14
anyObject	15
containsObject:	15
count	15
description	16
descriptionWithLocale:	16
enumerateObjectsUsingBlock:	17
enumerateObjectsWithOptions:usingBlock:	17
filteredSetUsingPredicate:	18
initWithArray:	18
initWithObjects:	19
initWithObjects:count:	20
initWithSet:	20
initWithSet:copyItems:	21
intersectsSet:	21
isEqualToSet:	22
isSubsetOfSet:	22
makeObjectsPerformSelector:	23
makeObjectsPerformSelector:withObject:	23
member:	24

- objectEnumerator 24
- objectsPassingTest: 25
- objectsWithOptions:passingTest: 26
- removeObserver:forKeyPath: 26
- setByAddingObject: 27
- setByAddingObjectsFromArray: 27
- setByAddingObjectsFromSet: 28
- setValue:forKey: 28
- sortedArrayUsingDescriptors: 29
- valueForKey: 29

Document Revision History 31

Index 33

NSSet Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSMutableCopying NSFastEnumeration NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Collections Programming Topics for Cocoa
Declared in	NSKeyValueCoding.h NSKeyValueObserving.h NSPredicate.h NSSet.h NSSortDescriptor.h
Related sample code	AnimatedTableView CoreRecipes QuickLookSketch Sketch+Accessibility Sketch-112

Overview

The `NSSet`, `NSMutableSet`, and `NSCountedSet` classes declare the programmatic interface to an object that manages a set of objects. `NSSet` provides support for the mathematical concept of a set. A set, both in its mathematical sense and in the implementation of `NSSet`, is an unordered collection of distinct elements. The `NSMutableSet` (a subclass of `NSSet`) and `NSCountedSet` (a subclass of `NSMutableSet`) classes are provided for sets whose contents may be altered.

`NSSet` and `NSMutableSet` are part of a class cluster, so sets are not actual instances of `NSSet` or `NSMutableSet`. Rather, the instances belong to one of their private subclasses. (For convenience, we use the term *set* to refer to any one of these instances without specifying its exact class membership.) Although a set's class is private, its interface is public, as declared by the abstract superclasses `NSSet` and `NSMutableSet`. Note that `NSCountedSet` is not part of the class cluster; it is a concrete subclass of `NSMutableSet`.

`NSSet` declares the programmatic interface for static sets of objects. You establish a static set's entries when it's created, and thereafter the entries can't be modified. `NSMutableSet`, on the other hand, declares a programmatic interface for dynamic sets of objects. A dynamic—or mutable—set allows the addition and deletion of entries at any time, automatically allocating memory as needed.

You can use sets as an alternative to arrays when the order of elements isn't important and performance in testing whether an object is contained in the set is a consideration—while arrays are ordered, testing for membership is slower than with sets.

Objects in a set must respond to the `NSObject` protocol methods `hash` and `isEqual:`—see the `NSObject` protocol for more information.

Note that if mutable objects are stored in a set, either the `hash` method of the objects shouldn't depend on the internal state of the mutable objects or the mutable objects shouldn't be modified while they're in the set (note that it can be difficult to know whether or not a given object is in a collection).

Objects added to a set are not copied; rather, an object receives a `retain` message before it's added to a set.

Typically, you create a temporary set by sending one of the `set...` methods to the `NSSet` class object. These methods return an `NSSet` object containing the elements (if any) you pass in as arguments. The `set` (page 9) method is a “convenience” method to create an empty mutable set.

The set classes adopt the `NSCopying` and `NSMutableCopying` protocols, making it convenient to convert a set of one type to the other.

`NSSet` provides methods for querying the elements of the set. `allObjects` (page 14) returns an array containing the objects in a set. `anyObject` (page 15) returns some object in the set. `count` (page 15) returns the number of objects currently in the set. `member:` (page 24) returns the object in the set that is equal to a specified object. Additionally, `intersectsSet:` (page 21) tests for set intersection, `isEqualToSet:` (page 22) tests for set equality, and `isSubsetOfSet:` (page 22) tests for one set being a subset of another.

The `objectEnumerator` (page 24) method provides for traversing elements of the set one by one. For better performance on Mac OS X v10.5 and later, you can also use the Objective-C fast enumeration feature (see Fast Enumeration).

`NSSet`'s `makeObjectsPerformSelector:` (page 23) and `makeObjectsPerformSelector:withObject:` (page 23) methods provides for sending messages to individual objects in the set.

`NSSet` is “toll-free bridged” with its Core Foundation counterpart, *CFSet Reference*. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSSet *` parameter, you can pass a `CFSetRef`, and in a function where you see a `CFSetRef` parameter, you can pass an `NSSet` instance (you cast one type to the other to suppress compiler warnings). See *Interchangeable Data Types* for more information on toll-free bridging.

Adopted Protocols

`NSCoding`

```
encodeWithCoder:
initWithCoder:
```

NSCopying`copyWithZone:`**NSMutableCopying**`mutableCopyWithZone:`

Tasks

Creating a Set

- + `set` (page 9)
Creates and returns an empty set.
- + `setWithArray:` (page 10)
Creates and returns a set containing a uniqued collection of those objects contained in a given array.
- + `setWithObject:` (page 10)
Creates and returns a set that contains a single given object.
- + `setWithObjects:` (page 11)
Creates and returns a set containing the objects in a given argument list.
- + `setWithObjects:count:` (page 12)
Creates and returns a set containing a specified number of objects from a given C array of objects.
- + `setWithSet:` (page 13)
Creates and returns a set containing the objects from another set.
- `setByAddingObject:` (page 27)
Returns a new set formed by adding a given object to the collection defined by the receiver.
- `setByAddingObjectsFromSet:` (page 28)
Returns a new set formed by adding the objects in a given set to the collection defined by the receiver.
- `setByAddingObjectsFromArray:` (page 27)
Returns a new set formed by adding the objects in a given array to the collection defined by the receiver.

Initializing a Set

- `initWithArray:` (page 18)
Initializes a newly allocated set with the objects that are contained in a given array.
- `initWithObjects:` (page 19)
Initializes a newly allocated set with members taken from the specified list of objects.
- `initWithObjects:count:` (page 20)
Initializes a newly allocated set with a specified number of objects from a given C array of objects.
- `initWithSet:` (page 20)
Initializes a newly allocated set and adds to it objects from another given set.
- `initWithSet:copyItems:` (page 21)
Initializes a newly allocated set and adds to it members of another given set.

Counting Entries

- [count](#) (page 15)
Returns the number of members in the receiver.

Accessing Set Members

- [allObjects](#) (page 14)
Returns an array containing the receiver's members, or an empty array if the receiver has no members.
- [anyObject](#) (page 15)
Returns one of the objects in the receiver, or `nil` if the receiver contains no objects.
- [containsObject:](#) (page 15)
Returns a Boolean value that indicates whether a given object is present in the receiver.
- [filteredSetUsingPredicate:](#) (page 18)
Evaluates a given predicate against each object in the receiver and returns a new set containing the objects for which the predicate returns true.
- [makeObjectsPerformSelector:](#) (page 23)
Sends to each object in the receiver a message specified by a given selector.
- [makeObjectsPerformSelector:withObject:](#) (page 23)
Sends to each object in the receiver a message specified by a given selector.
- [member:](#) (page 24)
Determines whether the receiver contains an object equal to a given object, and returns that object if it is present.
- [objectEnumerator](#) (page 24)
Returns an enumerator object that lets you access each object in the receiver.
- [enumerateObjectsUsingBlock:](#) (page 17)
Executes a given Block using each object in the receiver.
- [enumerateObjectsWithOptions:usingBlock:](#) (page 17)
Executes a given Block using each object in the receiver, using the specified enumeration options.
- [objectsPassingTest:](#) (page 25)
Returns a set of object that pass a test in a given Block.
- [objectsWithOptions:passingTest:](#) (page 26)
Returns a set of object that pass a test in a given Block, using the specified enumeration options.

Comparing Sets

- [isSubsetOfSet:](#) (page 22)
Returns a Boolean value that indicates whether every object in the receiver is also present in another given set.
- [intersectsSet:](#) (page 21)
Returns a Boolean value that indicates whether at least one object in the receiver is also present in another given set.
- [isEqualToSet:](#) (page 22)
Compares the receiver to another set.

- [valueForKey:](#) (page 29)
Return a set containing the results of invoking `valueForKey:` on each of the receiver's members.
- [setValue:forKey:](#) (page 28)
Invokes `setValue:forKey:` on each of the receiver's members.

Creating a Sorted Array

- [sortedArrayUsingDescriptors:](#) (page 29)
Returns an array of the receiver's content sorted as specified by a given array of sort descriptors.

Key-Value Observing

- [addObserver:forKeyPath:options:context:](#) (page 13)
Raises an exception.
- [removeObserver:forKeyPath:](#) (page 26)
Raises an exception.

Describing a Set

- [description](#) (page 16)
Returns a string that represents the contents of the receiver, formatted as a property list.
- [descriptionWithLocale:](#) (page 16)
Returns a string that represents the contents of the receiver, formatted as a property list.

Class Methods

set

Creates and returns an empty set.

```
+ (id)set
```

Return Value

A new empty set.

Discussion

This method is declared primarily for the use of mutable subclasses of `NSSet`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [setWithArray:](#) (page 10)
- + [setWithObject:](#) (page 10)
- + [setWithObjects:](#) (page 11)

- [setByAddingObject:](#) (page 27)
- [setByAddingObjectsFromSet:](#) (page 28)
- [setByAddingObjectsFromArray:](#) (page 27)

Related Sample Code

Core Data HTML Store
 CoreRecipes
 CustomAtomicStoreSubclass
 GLUT
 Sketch-112

Declared In

NSSet.h

initWithArray:

Creates and returns a set containing a uniqued collection of those objects contained in a given array.

```
+ (id) initWithArray:(NSArray *)anArray
```

Parameters

anArray

An array containing the objects to add to the new set. If the same object appears more than once in *anArray*, it is added only once to the returned set. Each object receives a `retain` message as it is added to the set.

Return Value

A new set containing a uniqued collection of those objects contained in *anArray*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [set](#) (page 9)
- + [initWithObject:](#) (page 10)
- + [initWithObjects:](#) (page 11)
- [setByAddingObject:](#) (page 27)
- [setByAddingObjectsFromSet:](#) (page 28)
- [setByAddingObjectsFromArray:](#) (page 27)

Related Sample Code

CoreRecipes
 NewsReader

Declared In

NSSet.h

initWithObject:

Creates and returns a set that contains a single given object.

```
+ (id)setWithObject:(id)anObject
```

Parameters

anObject

The object to add to the new set. *anObject* receives a `retain` message after being added to the set.

Return Value

A new set that contains a single member, *anObject*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [set](#) (page 9)
- + [setWithArray:](#) (page 10)
- + [setWithObjects:](#) (page 11)
- [setByAddingObject:](#) (page 27)
- [setByAddingObjectsFromSet:](#) (page 28)
- [setByAddingObjectsFromArray:](#) (page 27)

Related Sample Code

AnimatedTableView
 Core Data HTML Store
 SimpleTemperatureConverter
 Sketch+Accessibility

Declared In

NSSet.h

setWithObjects:

Creates and returns a set containing the objects in a given argument list.

```
+ (id)setWithObjects:(id)anObject ...
```

Parameters

anObject

The first object to add to the new set.

anObject, ...

A comma-separated list of objects, ending with `nil`, to add to the new set. If the same object appears more than once in the list of objects, it is added only once to the returned set. Each object receives a `retain` message as it is added to the set.

Return Value

A new set containing the objects in the argument list.

Discussion

As an example, the following code excerpt creates a set containing three different types of elements (assuming `aPath` exists):

```
NSSet *mySet;
NSData *someData = [NSData dataWithContentsOfFile:aPath];
```

```
NSNumber *aValue = [NSNumber numberWithInt:5];
NSString *aString = @"a string";
```

```
mySet = [NSSet initWithObjects:someData, aValue, aString, nil];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [set](#) (page 9)
- + [initWithArray:](#) (page 10)
- + [initWithObject:](#) (page 10)
- [setByAddingObject:](#) (page 27)
- [setByAddingObjectsFromSet:](#) (page 28)
- [setByAddingObjectsFromArray:](#) (page 27)

Related Sample Code

iSpend
iSpendPlugin
QTCoreVideo201
QuickLookSketch
Sketch+Accessibility

Declared In

NSSet.h

initWithObjects:count:

Creates and returns a set containing a specified number of objects from a given C array of objects.

```
+ (id)initWithObjects:(id *)objects count:(NSUInteger)count
```

Parameters

objects

A C array of objects to add to the new set. If the same object appears more than once in *objects*, it is added only once to the returned set. Each object receives a `retain` message as it is added to the set.

count

The number of objects from *objects* to add to the new set.

Return Value

A new set containing *count* objects from the list of objects specified by *objects*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [set](#) (page 9)
- + [initWithArray:](#) (page 10)
- + [initWithObject:](#) (page 10)
- + [initWithObjects:](#) (page 11)

- [setByAddingObject:](#) (page 27)
- [setByAddingObjectsFromSet:](#) (page 28)
- [setByAddingObjectsFromArray:](#) (page 27)

Declared In

NSSet.h

initWithSet:

Creates and returns a set containing the objects from another set.

```
+ (id)initWithSet:(NSSet *)aSet
```

Parameters*aSet*

A set containing the objects to add to the new set. Each object receives a `retain` message as it is added to the new set.

Return Value

A new set containing the objects from *aSet*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [set](#) (page 9)
- + [initWithArray:](#) (page 10)
- + [initWithObject:](#) (page 10)
- + [initWithObjects:](#) (page 11)
- [setByAddingObject:](#) (page 27)
- [setByAddingObjectsFromSet:](#) (page 28)
- [setByAddingObjectsFromArray:](#) (page 27)

Declared In

NSSet.h

Instance Methods

addObserver:forKeyPath:options:context:

Raises an exception.

```
- (void)addObserver:(NSObject *)observer forKeyPath:(NSString *)keyPath
options:(NSKeyValueObservingOptions)options context:(void *)context
```

Parameters*observer*

The object to register for KVO notifications. The observer must implement the key-value observing method `observeValueForKeyPath:ofObject:change:context:.`

keyPath

The key path, relative to the receiver, of the property to observe. This value must not be `nil`.

options

A combination of the `NSKeyValueObservingOptions` values that specifies what is included in observation notifications. For possible values, see `NSKeyValueObservingOptions`.

context

Arbitrary data that is passed to *observer* in `observeValueForKeyPath:ofObject:change:context:.`

Special Considerations

`NSSet` objects are not observable, so this method raises an exception when invoked on an `NSSet` object. Instead of observing a set, observe the unordered to-many relationship for which the set is the collection of related objects.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [removeObserver:forKeyPath:](#) (page 26)

Declared In

`NSKeyValueObserving.h`

allObjects

Returns an array containing the receiver's members, or an empty array if the receiver has no members.

- (NSArray *)allObjects

Return Value

An array containing the receiver's members, or an empty array if the receiver has no members. The order of the objects in the array isn't defined.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [anyObject](#) (page 15)
- [objectEnumerator](#) (page 24)

Related Sample Code

Core Data HTML Store
CoreRecipes
LightTable
Sketch-112

Declared In

`NSSet.h`

anyObject

Returns one of the objects in the receiver, or `nil` if the receiver contains no objects.

- (id)anyObject

Return Value

One of the objects in the receiver, or `nil` if the receiver contains no objects. The object returned is chosen at the receiver's convenience—the selection is not guaranteed to be random.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [allObjects](#) (page 14)
- [objectEnumerator](#) (page 24)

Related Sample Code

Core Data HTML Store

Declared In

NSSet.h

containsObject:

Returns a Boolean value that indicates whether a given object is present in the receiver.

- (BOOL)containsObject:(id)anObject

Parameters

anObject

The object for which to test membership of the receiver.

Return Value

YES if *anObject* is present in the receiver, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [member:](#) (page 24)

Related Sample Code

OpenGL Filter Basics Cocoa

Declared In

NSSet.h

count

Returns the number of members in the receiver.

- (NSUInteger)count

Return Value

The number of members in the receiver.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CoreRecipes

GeekGameBoard

LightTable

Declared In

NSSet.h

description

Returns a string that represents the contents of the receiver, formatted as a property list.

- (NSString *)description

Return Value

A string that represents the contents of the receiver, formatted as a property list.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [descriptionWithLocale:](#) (page 16)

Declared In

NSSet.h

descriptionWithLocale:

Returns a string that represents the contents of the receiver, formatted as a property list.

- (NSString *)descriptionWithLocale:(id)locale

Parameters

locale

In Mac OS X v10.4 and earlier, this must be a dictionary that specifies options used for formatting each of the receiver's members. In Mac OS X v10.5 and later, you can use an `NSLocale` object. If you do not want the receiver's members to be formatted, specify `nil`.

Return Value

A string that represents the contents of the receiver, formatted as a property list.

Discussion

This method sends each of the receiver's members `descriptionWithLocale:` with *locale* passed as the sole parameter. If the receiver's members do not respond to `descriptionWithLocale:`, this method sends `description` instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 16)

Declared In

NSSet.h

enumerateObjectsUsingBlock:

Executes a given Block using each object in the receiver.

```
- (void)enumerateObjectsUsingBlock:(void (^)(id obj, BOOL *stop))block
```

Parameters

block

The Block to apply to elements in the set.

The Block takes two arguments:

obj

The element in the set.

stop

A reference to a Boolean value. The block can set the value to YES to stop further processing of the set. The *stop* argument is an out-only argument. You should only ever set this Boolean to YES within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSSet.h

enumerateObjectsWithOptions:usingBlock:

Executes a given Block using each object in the receiver, using the specified enumeration options.

```
- (void)enumerateObjectsWithOptions:(NSEnumerationOptions)opts usingBlock:(void
  (^)(id obj, BOOL *stop))block
```

Parameters

opts

A bitmask that specifies the options for the enumeration (whether it should be performed concurrently and whether it should be performed in reverse order).

block

The Block to apply to elements in the set.

The Block takes two arguments:

obj

The element in the set.

stop

A reference to a Boolean value. The block can set the value to YES to stop further processing of the set. The *stop* argument is an out-only argument. You should only ever set this Boolean to YES within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSSet.h

filteredSetUsingPredicate:

Evaluates a given predicate against each object in the receiver and returns a new set containing the objects for which the predicate returns true.

```
- (NSSet *)filteredSetUsingPredicate:(NSPredicate *)predicate
```

Parameters

predicate

A predicate.

Return Value

A new set containing the objects in the receiver for which *predicate* returns true.

Discussion

The following example illustrates the use of this method.

```
NSSet *sourceSet =
    [NSSet setWithObjects:@"One", @"Two", @"Three", @"Four", nil];
NSPredicate *predicate =
    [NSPredicate predicateWithFormat:@"SELF beginswith 'T'"];
NSSet *filteredSet =
    [sourceSet filteredSetUsingPredicate:predicate];
// filteredSet contains (Two, Three)
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSPredicate.h

initWithArray:

Initializes a newly allocated set with the objects that are contained in a given array.

- (id)initWithArray:(NSArray *)array

Parameters

array

An array of objects to add to the new set. If the same object appears more than once in *array*, it is represented only once in the returned set. Each object receives a `retain` message as it is added to the set.

Return Value

An initialized object, which might be different than the original receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithObjects:](#) (page 19)
- [initWithObjects:count:](#) (page 20)
- [initWithSet:](#) (page 20)
- [initWithSet:copyItems:](#) (page 21)
- + [setWithArray:](#) (page 10)

Declared In

NSSet.h

initWithObjects:

Initializes a newly allocated set with members taken from the specified list of objects.

- (id)initWithObjects:(id)firstObj ...

Parameters

anObject

The first object to add to the new set.

firstObj, ...

A comma-separated list of objects, ending with `nil`, to add to the new set. If the same object appears more than once in the list, it is represented only once in the returned set. Each object receives a `retain` message as it is added to the set.

Return Value

An initialized object, which might be different than the original receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithArray:](#) (page 18)
- [initWithObjects:count:](#) (page 20)
- [initWithSet:](#) (page 20)
- [initWithSet:copyItems:](#) (page 21)
- + [setWithObjects:](#) (page 11)

Declared In

NSSet.h

initWithObjects:count:

Initializes a newly allocated set with a specified number of objects from a given C array of objects.

```
- (id)initWithObjects:(id *)objects count:(NSUInteger)count
```

Parameters

objects

A C array of objects to add to the new set. If the same object appears more than once in *objects*, it is added only once to the returned set. Each object receives a `retain` message as it is added to the set.

count

The number of objects from *objects* to add to the new set.

Return Value

An initialized object, which might be different than the original receiver.

Discussion

This method is the designated initializer for `NSSet`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithArray:](#) (page 18)
- [initWithObjects:](#) (page 19)
- [initWithSet:](#) (page 20)
- [initWithSet:copyItems:](#) (page 21)
- + [setWithObjects:count:](#) (page 12)

Declared In

`NSSet.h`

initWithSet:

Initializes a newly allocated set and adds to it objects from another given set.

```
- (id)initWithSet:(NSSet *)otherSet
```

Parameters

otherSet

A set containing objects to add to the receiver. Each object is retained as it is added to the receiver.

Return Value

An initialized object, which might be different than the original receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithArray:](#) (page 18)
- [initWithObjects:](#) (page 19)
- [initWithObjects:count:](#) (page 20)

- initWithSet:copyItems: (page 21)
- + setWithSet: (page 13)

Declared In

NSSet.h

initWithSet:copyItems:

Initializes a newly allocated set and adds to it members of another given set.

```
- (id)initWithSet:(NSSet *)otherSet copyItems:(BOOL)flag
```

Parameters*otherSet*

A set containing objects to add to the new set.

flag

If YES, the members of *otherSet* are copied, and the copies are added to the receiver. If NO, the members of *otherSet* are added to the receiver and retained.

Return Value

An initialized object that contains the members of *otherSet*.

This method returns an initialized object, which might be different than the original receiver.

Discussion

Note that, if *flag* is YES, `copyWithZone:` is invoked to make copies—thus, the receiver's new member objects may be immutable, even though their counterparts in *otherSet* were mutable. Also, members must conform to the `NSCopying` protocol)

Availability

Available in Mac OS X v10.0 and later.

See Also

- initWithArray: (page 18)
- initWithObjects: (page 19)
- initWithObjects:count: (page 20)
- initWithSet: (page 20)
- + setWithSet: (page 13)

Declared In

NSSet.h

intersectsSet:

Returns a Boolean value that indicates whether at least one object in the receiver is also present in another given set.

```
- (BOOL)intersectsSet:(NSSet *)otherSet
```

Parameters*otherSet*

The set with which to compare the receiver.

Return Value

YES if at least one object in the receiver is also present in *otherSet*, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [isEqualToSet:](#) (page 22)
- [isSubsetOfSet:](#) (page 22)

Declared In

NSSet.h

isEqualToSet:

Compares the receiver to another set.

```
- (BOOL)isEqualToSet:(NSSet *)otherSet
```

Parameters*otherSet*

The set with which to compare the receiver.

Return Value

YES if the contents of *otherSet* are equal to the contents of the receiver, otherwise NO.

Discussion

Two sets have equal contents if they each have the same number of members and if each member of one set is present in the other.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [intersectsSet:](#) (page 21)
- `isEqual:` (NSObject protocol)
- [isSubsetOfSet:](#) (page 22)

Declared In

NSSet.h

isSubsetOfSet:

Returns a Boolean value that indicates whether every object in the receiver is also present in another given set.

```
- (BOOL)isSubsetOfSet:(NSSet *)otherSet
```

Parameters*otherSet*

The set with which to compare the receiver.

Return Value

YES if every object in the receiver is also present in *otherSet*, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [intersectsSet:](#) (page 21)
- [isEqualToSet:](#) (page 22)

Declared In

NSSet.h

makeObjectsPerformSelector:

Sends to each object in the receiver a message specified by a given selector.

```
- (void)makeObjectsPerformSelector:(SEL)aSelector
```

Parameters*aSelector*

A selector that specifies the message to send to the members of the receiver. The method must not take any arguments. It should not have the side effect of modifying the receiver. This value must not be NULL.

Discussion

The message specified by *aSelector* is sent once to each member of the receiver. This method raises an `NSInvalidArgumentException` if *aSelector* is NULL.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [makeObjectsPerformSelector:withObject:](#) (page 23)

Declared In

NSSet.h

makeObjectsPerformSelector:withObject:

Sends to each object in the receiver a message specified by a given selector.

```
- (void)makeObjectsPerformSelector:(SEL)aSelector withObject:(id)anObject
```

Parameters*aSelector*

A selector that specifies the message to send to the receiver's members. The method must take a single argument of type `id`. The method should not, as a side effect, modify the receiver. The value must not be NULL.

anObject

The object to pass as an argument to the method specified by *aSelector*.

Discussion

The message specified by *aSelector* is sent, with *anObject* as the argument, once to each member of the receiver. This method raises an `NSInvalidArgumentException` if *aSelector* is `NULL`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [makeObjectsPerformSelector:](#) (page 23)

Declared In

`NSet.h`

member:

Determines whether the receiver contains an object equal to a given object, and returns that object if it is present.

```
- (id)member:(id)anObject
```

Parameters

anObject

The object for which to test for membership of the receiver.

Return Value

If the receiver contains an object equal to *anObject* (as determined by `isEqual:`) then that object (typically this will be *anObject*), otherwise `nil`.

Discussion

If you override `isEqual:`, you must also override the `hash` method for the `member:` method to work on a set of objects of your class.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSet.h`

objectEnumerator

Returns an enumerator object that lets you access each object in the receiver.

```
- (NSEnumerator *)objectEnumerator
```

Return Value

An enumerator object that lets you access each object in the receiver.

Discussion

The following code fragment illustrates how you can use this method.

```
NSEnumerator *enumerator = [mySet objectEnumerator];
```

```
id value;

while ((value = [enumerator nextObject])) {
    /* code that acts on the set's values */
}
```

When this method is used with mutable subclasses of `NSSet`, your code shouldn't modify the receiver during enumeration. If you intend to modify the receiver, use the [allObjects](#) (page 14) method to create a "snapshot" of the set's members. Enumerate the snapshot, but make your modifications to the original set.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `nextObject` (`NSEnumerator`)

Related Sample Code

CoreRecipes

GLUT

QuickLookSketch

Sketch+Accessibility

Sketch-112

Declared In

`NSSet.h`

objectsPassingTest:

Returns a set of object that pass a test in a given Block.

```
- (NSSet *)objectsPassingTest:(BOOL (^)(id obj, BOOL *stop))predicate
```

Parameters

predicate

The block to apply to elements in the array.

The block takes three arguments:

obj

The element in the set.

stop

A reference to a Boolean value. The block can set the value to `YES` to stop further processing of the set. The `stop` argument is an out-only argument. You should only ever set this Boolean to `YES` within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

Return Value

An `NSSet` containing objects that pass the test.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSSet.h

objectsWithOptions:passingTest:

Returns a set of object that pass a test in a given Block, using the specified enumeration options.

```
- (NSSet *)objectsWithOptions:(NSEnumerationOptions)opts passingTest:(BOOL (^)(id obj, BOOL *stop))predicate
```

Parameters*opts*

A bitmask that specifies the options for the enumeration (whether it should be performed concurrently and whether it should be performed in reverse order).

predicate

The Block to apply to elements in the set.

The Block takes two arguments:

obj

The element in the set.

stop

A reference to a Boolean value. The block can set the value to YES to stop further processing of the set. The *stop* argument is an out-only argument. You should only ever set this Boolean to YES within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

Return Value

An NSSet containing objects that pass the test.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSSet.h

removeObserver:forKeyPath:

Raises an exception.

```
- (void)removeObserver:(NSObject *)observer forKeyPath:(NSString *)keyPath
```

Parameters*observer*

The object to remove as an observer.

keyPath

A key-path, relative to the receiver, for which *observer* is registered to receive KVO change notifications. This value must not be nil.

Special Considerations

`NSSet` objects are not observable, so this method raises an exception when invoked on an `NSSet` object. Instead of observing a set, observe the unordered to-many relationship for which the set is the collection of related objects.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [addObserver:forKeyPath:options:context:](#) (page 13)

Declared In

`NSKeyValueObserving.h`

setByAddingObject:

Returns a new set formed by adding a given object to the collection defined by the receiver.

```
- (NSSet *)setByAddingObject:(id)anObject
```

Parameters

anObject

The object to add to the collection defined by the receiver.

Return Value

A new set formed by adding *anObject* to the collection defined by the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- + [set](#) (page 9)
- + [setWithArray:](#) (page 10)
- + [setWithObject:](#) (page 10)
- + [setWithObjects:](#) (page 11)
- [setByAddingObjectsFromSet:](#) (page 28)
- [setByAddingObjectsFromArray:](#) (page 27)

Declared In

`NSSet.h`

setByAddingObjectsFromArray:

Returns a new set formed by adding the objects in a given array to the collection defined by the receiver.

```
- (NSSet *)setByAddingObjectsFromArray:(NSArray *)other
```

Parameters

other

The array of objects to add to the collection defined by the receiver.

Return Value

A new set formed by adding the objects in *other* to the collection defined by the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- + [set](#) (page 9)
- + [setWithArray:](#) (page 10)
- + [setWithObject:](#) (page 10)
- + [setWithObjects:](#) (page 11)
- [setByAddingObject:](#) (page 27)
- [setByAddingObjectsFromSet:](#) (page 28)

Declared In

NSSet.h

setByAddingObjectsFromSet:

Returns a new set formed by adding the objects in a given set to the collection defined by the receiver.

```
- (NSSet *)setByAddingObjectsFromSet:(NSSet *)other
```

Parameters

other

The set of objects to add to the collection defined by the receiver.

Return Value

A new set formed by adding the objects in *other* to the collection defined by the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- + [set](#) (page 9)
- + [setWithArray:](#) (page 10)
- + [setWithObject:](#) (page 10)
- + [setWithObjects:](#) (page 11)
- [setByAddingObject:](#) (page 27)
- [setByAddingObjectsFromArray:](#) (page 27)

Declared In

NSSet.h

setValue:forKey:

Invokes `setValue:forKey:` on each of the receiver's members.

```
- (void)setValue:(id)value forKey:(NSString *)key
```

Parameters*value*The value for the property identified by *key*.*key*

The name of one of the properties of the receiver's members.

Availability

Available in Mac OS X v10.4 and later.

See Also- [valueForKey:](#) (page 29)**Related Sample Code**

CustomAtomicStoreSubclass

Declared In

NSKeyValueCoding.h

sortedArrayUsingDescriptors:

Returns an array of the receiver's content sorted as specified by a given array of sort descriptors.

- (NSArray *)sortedArrayUsingDescriptors:(NSArray *)*sortDescriptors***Parameters***sortDescriptors*

An array of NSSortDescriptor objects.

Return ValueAn NSArray containing the receiver's sorted as specified by *sortDescriptors*.**Discussion**

The first descriptor specifies the primary key path to be used in sorting the receiver's contents. Any subsequent descriptors are used to further refine sorting of objects with duplicate values. See NSSortDescriptor for additional information.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSSortDescriptor.h

valueForKey:Return a set containing the results of invoking `valueForKey:` on each of the receiver's members.- (id)valueForKey:(NSString *)*key***Parameters***key*

The name of one of the properties of the receiver's members.

Return Value

A set containing the results of invoking `valueForKey:` (with the argument *key*) on each of the receiver's members.

Discussion

The returned set might not have the same number of members as the receiver. The returned set will not contain any elements corresponding to instances of `valueForKey:` returning `nil` (note that this is in contrast with `NSArray`'s implementation, which may put `NSNull` values in the arrays it returns).

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setValueForKey:](#) (page 28)

Declared In

`NSKeyValueCoding.h`

Document Revision History

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

Date	Notes
2009-08-18	Updated for Mac OS X v 10.6. Added new enumeration methods using Blocks . Added new content testing methods. Added method that returns sorted array.
2008-10-15	Corrected typographical errors.
2008-03-11	Added NSFastEnumeration to list of adopted protocols.
2008-02-08	Corrected the definition of the member: method.
2007-12-11	Corrected typographical error.
2007-04-02	Included API introduced in Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

addObserver:forKeyPath:options:context:
instance method 13
allObjects instance method 14
anyObject instance method 15

C

containsObject: instance method 15
count instance method 15

D

description instance method 16
descriptionWithLocale: instance method 16

E

enumerateObjectsUsingBlock: instance method 17
enumerateObjectsWithOptions:usingBlock:
instance method 17

F

filteredSetUsingPredicate: instance method 18

I

initWithArray: instance method 18
initWithObjects: instance method 19
initWithObjects:count: instance method 20
initWithSet: instance method 20

initWithSet:copyItems: instance method 21
intersectsSet: instance method 21
isEqualToSet: instance method 22
isSubsetOfSet: instance method 22

M

makeObjectsPerformSelector: instance method 23
makeObjectsPerformSelector:withObject: instance
method 23
member: instance method 24

O

objectEnumerator instance method 24
objectsPassingTest: instance method 25
objectsWithOptions:passingTest: instance method
26

R

removeObserver:forKeyPath: instance method 26

S

set class method 9
setByAddingObject: instance method 27
setByAddingObjectsFromArray: instance method 27
setByAddingObjectsFromSet: instance method 28
setValue:forKey: instance method 28
setWithArray: class method 10
setWithObject: class method 10
setWithObjects: class method 11
setWithObjects:count: class method 12
setWithSet: class method 13
sortedArrayUsingDescriptors: instance method 29

V

valueForKey: [instance method](#) [29](#)