

---

# NSPredicate Class Reference

[Cocoa > Data Management](#)



2006-08-16



Apple Inc.  
© 2006 Apple Computer, 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, Leopard, Logic, Mac, and Mac OS 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

---

## **NSPredicate Class Reference 5**

---

Overview	5
Tasks	6
Constructors	6
Evaluating a Predicate	6
Getting Format Information	6
Class Methods	7
predicateWithFormat:	7
predicateWithFormat:argumentArray:	7
predicateWithFormat:arguments:	8
predicateWithValue:	8
Instance Methods	9
evaluateWithObject:	9
evaluateWithObject:substitutionVariables:	9
predicateFormat	10
predicateWithSubstitutionVariables:	10

---

## **Document Revision History 11**

---

## **Index 13**

---



# NSPredicate Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSCoding NSCopying NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/Foundation.framework
<b>Availability</b>	Available in Mac OS X v10.4 and later.
<b>Companion guide</b>	Predicate Programming Guide
<b>Declared in</b>	NSPredicate.h
<b>Related sample code</b>	CoreRecipes DerivedProperty iSpend PredicateEditorSample SimpleCalendar

## Overview

The `NSPredicate` class is used to define logical conditions used to constrain a search either for a fetch or for in-memory filtering.

You use predicates to represent logical conditions, used for describing objects in persistent stores and in-memory filtering of objects. Although it is common to create predicates directly from instances of `NSComparisonPredicate`, `NSCompoundPredicate`, and `NSExpression`, you often create predicates from a format string which is parsed by the class methods on `NSPredicate`. Examples of predicate format strings include:

- Simple comparisons, such as `grade == "7"` or `firstName like "Shaffiq"`
- Case/diacritic insensitive lookups, such as `name contains[cd] "itroen"`
- Logical operations, such as `(firstName like "Mark") OR (lastName like "Adderley")`
- With Mac OS X version 10.5 and later, you can create “between” predicates such as `date between {$YESTERDAY, $TOMORROW}`.

You can create predicates for relationships, such as:

- `group.name like "work*"`

- ALL children.age > 12
- ANY children.age > 12

You can create predicates for operations, such as `@sum.items.price < 1000`. For a complete syntax reference, refer to the *Predicate Programming Guide*.

You can also create predicates that include variables, so that the predicate can be pre-defined before substituting concrete values at runtime. On Mac OS X v10.4, for predicates that use variables, evaluation is a two step process (see [predicateWithSubstitutionVariables:](#) (page 10) and [evaluateWithObject:](#) (page 9)). Mac OS X v10.5 introduces a new method, [evaluateWithObject:substitutionVariables:](#) (page 9), which combines these steps.

## Tasks

### Constructors

- + [predicateWithFormat:](#) (page 7)  
Creates and returns a new predicate formed by creating a new string with a given format and parsing the result.
- + [predicateWithFormat:argumentArray:](#) (page 7)  
Creates and returns a new predicate by substituting the values in a given array into a format string and parsing the result.
- + [predicateWithFormat:arguments:](#) (page 8)  
Creates and returns a new predicate by substituting the values in an argument list into a format string and parsing the result.
- [predicateWithSubstitutionVariables:](#) (page 10)  
Returns a copy of the receiver with the receiver's variables substituted by values specified in a given substitution variables dictionary.
- + [predicateWithValue:](#) (page 8)  
Creates and returns a predicate that always evaluates to a given value.

### Evaluating a Predicate

- [evaluateWithObject:](#) (page 9)  
Returns a Boolean value that indicates whether a given object matches the conditions specified by the receiver.
- [evaluateWithObject:substitutionVariables:](#) (page 9)  
Returns a Boolean value that indicates whether a given object matches the conditions specified by the receiver after substituting in the values in a given variables dictionary.

### Getting Format Information

- [predicateFormat](#) (page 10)  
Returns the receiver's format string.

## Class Methods

### **predicateWithFormat:**

Creates and returns a new predicate formed by creating a new string with a given format and parsing the result.

```
+ (NSPredicate *)predicateWithFormat:(NSString *)format, ...
```

#### **Parameters**

*format*

The format string for the new predicate.

...

A comma-separated list of arguments to substitute into *format*.

#### **Return Value**

A new predicate formed by creating a new string with *format* and parsing the result.

#### **Discussion**

For details of the format of the format string and of limitations on variable substitution, see Predicate Format String Syntax.

#### **Availability**

Available in Mac OS X v10.4 and later.

#### **Related Sample Code**

CoreRecipes

iSpend

QTMetadataEditor

SimpleCalendar

SpotlightFortunes

#### **Declared In**

NSPredicate.h

### **predicateWithFormat:argumentArray:**

Creates and returns a new predicate by substituting the values in a given array into a format string and parsing the result.

```
+ (NSPredicate *)predicateWithFormat:(NSString *)predicateFormat
argumentArray:(NSArray *)arguments
```

#### **Parameters**

*predicateFormat*

The format string for the new predicate.

*arguments*

The arguments to substitute into *predicateFormat*. Values are substituted into *predicateFormat* in the order they appear in the array.

**Return Value**

A new predicate by substituting the values in *arguments* into *predicateFormat*, and parsing the result.

**Discussion**

For details of the format of the format string and of limitations on variable substitution, see Predicate Format String Syntax.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSPredicate.h

**predicateWithFormat:arguments:**

Creates and returns a new predicate by substituting the values in an argument list into a format string and parsing the result.

```
+ (NSPredicate *)predicateWithFormat:(NSString *)format arguments:(va_list)argList
```

**Parameters**

*format*

The format string for the new predicate.

*argList*

The arguments to substitute into *predicateFormat*. Values are substituted into *predicateFormat* in the order they appear in the argument list.

**Return Value**

A new predicate by substituting the values in *argList* into *predicateFormat* and parsing the result.

**Discussion**

For details of the format of the format string and of limitations on variable substitution, see Predicate Format String Syntax.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSPredicate.h

**predicateWithValue:**

Creates and returns a predicate that always evaluates to a given value.

```
+ (NSPredicate *)predicateWithValue:(BOOL)value
```

**Parameters**

*value*

The value to which the new predicate should evaluate.

**Return Value**

A predicate that always evaluates to *value*.

**Availability**

Available in Mac OS X v10.4 and later.

**Related Sample Code**

CoreRecipes

**Declared In**

NSPredicate.h

## Instance Methods

**evaluateWithObject:**

Returns a Boolean value that indicates whether a given object matches the conditions specified by the receiver.

```
- (BOOL)evaluateWithObject:(id)object
```

**Parameters**

*object*

The object against which to evaluate the receiver.

**Return Value**

YES if *object* matches the conditions specified by the receiver, otherwise NO.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSPredicate.h

**evaluateWithObject:substitutionVariables:**

Returns a Boolean value that indicates whether a given object matches the conditions specified by the receiver after substituting in the values in a given variables dictionary.

```
- (BOOL)evaluateWithObject:(id)object
    substitutionVariables:(NSDictionary *)variables
```

**Parameters**

*object*

The object against which to evaluate the receiver.

*variables*

The substitution variables dictionary. The dictionary must contain key-value pairs for all variables in the receiver.

**Return Value**

YES if *object* matches the conditions specified by the receiver after substituting in the values in *variables* for any replacement tokens, otherwise NO.

**Discussion**

This method returns the same result as the two step process of first invoking [predicateWithSubstitutionVariables:](#) (page 10) on the receiver and then invoking [evaluateWithObject:](#) (page 9) on the returned predicate. This method is optimized for situations which require repeatedly evaluating a predicate with substitution variables with different variable substitutions.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

NSPredicate.h

**predicateFormat**

Returns the receiver's format string.

```
- (NSString *)predicateFormat
```

**Return Value**

The receiver's format string.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSPredicate.h

**predicateWithSubstitutionVariables:**

Returns a copy of the receiver with the receiver's variables substituted by values specified in a given substitution variables dictionary.

```
- (NSPredicate *)predicateWithSubstitutionVariables:(NSDictionary *)variables
```

**Parameters**

*variables*

The substitution variables dictionary. The dictionary must contain key-value pairs for all variables in the receiver.

**Return Value**

A copy of the receiver with the receiver's variables substituted by values specified in *variables*.

**Discussion**

The receiver itself is not modified by this method, so you can reuse it for any number of substitutions.

**Availability**

Available in Mac OS X v10.4 and later.

**Related Sample Code**

DerivedProperty

**Declared In**

NSPredicate.h

# Document Revision History

---

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

Date	Notes
2006-08-16	Corrected minor typographical error.
Leopard WWDC	Abandoned in favour of 2.1.
2006-06-28	Added NSObject to the list of adopted protocols; corrected minor formatting errors.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## E

---

evaluateWithObject: **instance method 9**  
evaluateWithObject:substitutionVariables:  
    **instance method 9**

## P

---

predicateFormat **instance method 10**  
predicateWithFormat: **class method 7**  
predicateWithFormat:argumentArray: **class method 7**  
predicateWithFormat:arguments: **class method 8**  
predicateWithSubstitutionVariables: **instance method 10**  
predicateWithValue: **class method 8**