

---

# Core Data Snippets

[Cocoa > Data Management](#)



2009-03-04



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, and Cocoa 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

---

## **Introduction 5**

Organization of This Document 5

---

## **Accessing the Core Data Stack 7**

Getting a Managed Object Context 7  
Creating a New Managed Object Context 7  
Getting the Managed Object Model 8  
Adding a Persistent Store 8

---

## **Fetching Managed Objects 11**

Basic Fetch 11  
Fetch with Sorting 11  
Fetch with a Predicate 12  
Fetch with a Predicate Template 12  
Fetch with Sorting and a Predicate 13

---

## **Creating and Deleting Managed Objects 15**

Creating a Managed Object 15  
Saving a Managed Object 15  
Deleting a Managed Object 15

---

## **Document Revision History 17**

---



# Introduction

---

This document contains prototypes for commonly-used snippets of code that you're likely to use in a program that uses Core Data. In some cases (particularly in cases where a code snippet might be only one or two lines), the document provides guidance rather than explicit code.

This document does not provide an in-depth explanation of the code snippets. You're expected to be familiar with the Core Data framework and understand how to use the snippets in context. To learn more about Core Data, read *Core Data Programming Guide*.

## Organization of This Document

The document contains the following articles:

- [“Accessing the Core Data Stack”](#) (page 7)
- [“Fetching Managed Objects”](#) (page 11)
- [“Creating and Deleting Managed Objects”](#) (page 15)



# Accessing the Core Data Stack

---

This article contains snippets for creating and accessing parts of the Core Data stack.

How you access the parts of the Core Data stack may depend in part on the application architecture. Broadly-speaking, there are two basic application architectures for programs that use Core Data:

- Single-coordinator applications.

These applications typically have a single Core Data stack (as defined by a single persistent store coordinator). They generally use a single persistent store for the whole application.

- Document-based applications.

These use the Application Kit's `NSPersistentDocument` class. There is typically a persistent store coordinator and a single persistent store associated with each document.

This article uses the terms “single-coordinator application” and “document-based application” to differentiate between these architectures.

## Getting a Managed Object Context

In an single-coordinator applications, you can get the application's context directly from the application delegate.

In document-based applications, you can get the context directly from the document instance.

Sometimes, though, it's easier or more appropriate to retrieve the context from somewhere other than application or the document. Several objects you might use in a Core Data-based application keep a reference to a managed object context. A managed object itself has a reference to its own context, as do the various controller objects that support Core Data such as array and object controllers (`NSArrayController` and `NSObjectController`).

Retrieving the context from one of these objects has the advantage that if you re-architect your application, for example to make use of multiple contexts, your code is likely to remain valid. For example, if you have a managed object, and you want to create a new managed object that will be related to it, you can ask original object for its managed object context and create the new object using that. This will ensure that the new object you create is in the same context as the original.

## Creating a New Managed Object Context

You sometimes need to create a new managed object context to contain a disjoint set of edits that you might want to discard without perturbing the main context (for example, if you're presenting a modal view to add and edit a new object).

To create a new managed object context, you need a persistent store coordinator.

```
NSPersistentStoreCoordinator *psc = <#Get the coordinator#>;
NSManagedObjectContext *newContext = [[NSManagedObjectContext alloc] init];
[newContext setPersistentStoreCoordinator:psc];
```

In a single-coordinator application, you can get the coordinator directly from the application delegate

In a document-based application, you get the persistent store coordinator from the document instance's managed object context.

If you already have a reference to a context, you can ask it for its persistent store coordinator. This way you can be sure that the new context is using the same coordinator as the existing one (assuming this is your intent):

```
NSManagedObjectContext *context = <#Get the context#>;
NSPersistentStoreCoordinator *psc = [context persistentStoreCoordinator];
NSManagedObjectContext *newContext = [[NSManagedObjectContext alloc] init];
[newContext setPersistentStoreCoordinator:psc];
```

## Getting the Managed Object Model

You sometimes need to access a managed object model to get information about a particular entity.

Applications typically have just a single model (although it may have more than one configuration). In a single-coordinator application, you typically get the model directly from the application delegate. In a document-based application, you get the model directly from the document.

If you have access to a managed object context—directly or indirectly (see [“Getting a Managed Object Context”](#) (page 7))—you can get the model from the context's persistent store coordinator.

**Creating a managed object:** When you create a new managed object, you need to specify its entity. Typically, however, you don't actually need to access the entity or model directly—see [“Creating and Deleting Managed Objects”](#) (page 15).

## Adding a Persistent Store

In many applications, there is only one persistent store for each persistent store coordinator. In a single-coordinator application, the store is associated with the whole application. In a document-based application, the each document has a separate store. Sometimes, however, you might want to add another store. You add the store to the persistent store coordinator. You have to specify the store's type, location, and configuration (based on configurations present on the managed object model associated with the coordinator). You can also specify other options, such as whether an old version of the store should be migrated to a current version (see *Core Data Model Versioning and Data Migration Programming Guide*).

In a single-coordinator applications, you can get the application's coordinator directly from the application delegate.

In document-based applications, you can get the coordinator from the document's managed object context.



```
NSPersistentStoreCoordinator *psc = <#Get the coordinator#>;
NSURL *storeUrl = [NSURL fileURLWithPath:@"<#Path to store#>"];
NSString *storeType = <#Store type#>; // A store type, such as NSSQLiteStoreType
NSError *error;
if (![psc addPersistentStoreWithType:storeType configuration:nil
    URL:storeUrl options:nil error:&error]) {
    // Handle error
}
```



# Fetching Managed Objects

---

This article contains snippets for fetching managed objects.

To fetch managed objects, you minimally need a managed object context against which to execute the fetch, and an entity description to specify the entity you want. You create an instance of `NSFetchRequest` and specify its entity. You may optionally specify an array of sort orderings and/or a predicate.

How you get the managed object context depends on your application architecture—see [“Getting a Managed Object Context”](#) (page 7). Once you have the context, though, you can get the entity using `NSEntityDescription`’s convenience method, `entityForName:inManagedObjectContext:`.

## Basic Fetch

To get all the managed objects of a given entity, create a fetch request and specify just the entity:

```
NSManagedObjectContext *context = <#Get the context#>;

NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
NSEntityDescription *entity = [NSEntityDescription entityForName:@"<#Entity
name#>"
    inManagedObjectContext:context];
[fetchRequest setEntity:entity];

NSError *error;
NSArray *fetchedObjects = [context executeFetchRequest:fetchRequest error:&error];
if (fetchedObjects == nil) {
    // Handle error
}

[fetchRequest release];
```

## Fetch with Sorting

To fetch managed objects in a particular order, in addition to the components in the basic fetch (described in [“Basic Fetch”](#) (page 11)) you need to specify an array of sort orderings:

```
NSManagedObjectContext *context = <#Get the context#>;

NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
NSEntityDescription *entity = [NSEntityDescription entityForName:@"<#Entity
name#>"
    inManagedObjectContext:context];
[fetchRequest setEntity:entity];
```

```

NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc] initWithKey:@"<#Sort
key#>"
    ascending:YES];
NSArray *sortDescriptors = [[NSArray alloc] initWithObjects:sortDescriptor,
nil];
[fetchRequest setSortDescriptors:sortDescriptors];

NSError *error;
NSArray *fetchedObjects = [context executeFetchRequest:fetchRequest error:&error];
if (fetchedObjects == nil) {
    // Handle error
}

[fetchRequest release];
[sortDescriptor release];
[sortDescriptors release];

```

## Fetch with a Predicate

To fetch managed objects that meet given criteria, in addition to the components in the basic fetch (described in [“Basic Fetch”](#) (page 11)) you need to specify a predicate:

```

NSManagedObjectContext *context = <#Get the context#>;

NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
NSEntityDescription *entity = [NSEntityDescription entityForName:@"<#Entity
name#>"
    inManagedObjectContext:context];
[fetchRequest setEntity:entity];

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"<#Predicate string#>",
    <#Predicate arguments#>];
[request setPredicate:predicate];

NSError *error;
NSArray *fetchedObjects = [context executeFetchRequest:fetchRequest error:&error];
if (fetchedObjects == nil) {
    // Handle error
}

[fetchRequest release];

```

For more about predicates, see *Predicate Programming Guide*. For an alternative technique for creating the predicate that may be more efficient, see [“Fetch with a Predicate Template”](#) (page 12).

## Fetch with a Predicate Template

To fetch managed objects that meet given criteria, in addition to the components in the basic fetch (described in [“Basic Fetch”](#) (page 11)) you need to specify a predicate. `NSPredicate`’s `predicateWithFormat:` method is typically the easiest way to use a predicate (as shown in [“Fetch with a Predicate”](#) (page 12)), but it’s not the most efficient way to create the predicate itself. The predicate format string has to be parsed, arguments

substituted, and so on. For performance-critical code, particularly if a given predicate is used repeatedly, you should consider other ways to create the predicate. For a predicate that you might use frequently, the easiest first step is to create a predicate template. You might create an accessor method that creates the predicate template lazily on demand:

```
// Assume an instance variable:
// NSPredicate *predicateTemplate;

- (NSPredicate *)predicateTemplate {
    if (predicateTemplate == nil) {
        predicateTemplate = [[NSPredicate predicateWithFormat:
            @"<#Key#> <#Operator#> <#Variable#>"] retain];
    }
    return predicateTemplate;
}
```

When you need to use the template, you create a dictionary containing the substitution variables and generate the predicate using `predicateWithSubstitutionVariables:`.

```
NSManagedObjectContext *context = <#Get the context#>;

NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
NSEntityDescription *entity = [NSEntityDescription entityForName:@"<#Entity
name#>"
    inManagedObjectContext:context];
[fetchRequest setEntity:entity];

NSDictionary *variables = [[NSDictionary alloc] initWithObjectsAndKeys:
    <#Value#>, @"<#Variable#>", nil];
NSPredicate *predicate = [[self predicateTemplate]
    predicateWithSubstitutionVariables:variables];
[request setPredicate:predicate];

NSError *error;
NSArray *fetchedObjects = [context executeFetchRequest:fetchRequest error:&error];
if (fetchedObjects == nil) {
    // Handle error
}

[fetchRequest release];
[variables release];
```

For more about predicates, see *Predicate Programming Guide*.

## Fetch with Sorting and a Predicate

To fetch managed objects that meet given criteria and in a particular order, in addition to the components in the basic fetch (described in “Basic Fetch” (page 11)) you need to specify a predicate and an array of sort orderings.

```
NSManagedObjectContext *context = <#Get the context#>;

NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
```

## Fetching Managed Objects

```
NSEntityDescription *entity = [NSEntityDescription entityForName:@"<#Entity
name#>" inManagedObjectContext:context];
[fetchRequest setEntity:entity];

NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc] initWithKey:@"<#Sort
key#>" ascending:YES];
NSArray *sortDescriptors = [[NSArray alloc] initWithObjects:sortDescriptor,
nil];
[fetchRequest setSortDescriptors:sortDescriptors];

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"<#Predicate string#>",
<#Predicate arguments#>];
[request setPredicate:predicate];

NSError *error;
NSArray *fetchedObjects = [context executeFetchRequest:fetchRequest error:&error];
if (fetchedObjects == nil) {
    // Handle error
}

[fetchRequest release];
[sortDescriptor release];
[sortDescriptors release];
```

For more about predicates, see *Predicate Programming Guide*. For an alternative technique for creating the predicate that may be more efficient, see [“Fetch with a Predicate Template”](#) (page 12).

# Creating and Deleting Managed Objects

---

This article contains snippets you use when creating or deleting a managed object.

## Creating a Managed Object

When you create a new managed object, you need to specify its entity. Typically, however, you don't actually need access to the model directly. Instead, you can `NSEntityDescription`'s class method `insertNewObjectForEntityForName:inManagedObjectContext:` and pass the managed object context in which you want to create the new managed object. The method returns an instance of whatever class is defined in the managed object model to represent the entity, initialized with the default values given for its entity in the model.

To learn how to retrieve the managed object context, read ["Getting a Managed Object Context"](#) (page 7).

```
NSManagedObjectContext *context = <#Get the context#>;
<#Managed Object Class#> *newObject = [NSEntityDescription
    insertNewObjectForEntityForName:@"<#Entity name#>"
    inManagedObjectContext:context];
```

It is typically important to cast the new instance to the managed object class so that you can use the appropriate accessor methods without the compiler generating a warning (or, if you're using dot syntax, an error).

## Saving a Managed Object

Simply creating a managed object does not cause it to be saved to a persistent store. It is simply associated with the managed object context. To commit changes to the store, you send the context a `save:` message.

To learn how to retrieve the managed object context, read ["Getting a Managed Object Context"](#) (page 7).

```
NSManagedObjectContext *context = <#Get the context#>;
NSError *error;
if (![context save:&error]) {
    // Handle error
}
```

## Deleting a Managed Object

Simply being deallocated does not cause a managed object to be deleted from the persistent store. To delete a managed object you have to delete it from the context then save the context.

To learn how to retrieve the managed object context, read [“Getting a Managed Object Context”](#) (page 7)—or you can simply ask the object itself what context it belongs to.

```
NSManagedObject *aManagedObject = <#Get the managed object#>;
NSManagedObjectContext *context = [aManagedObject managedObjectContext];
[context deleteObject:aManagedObject];
NSError *error;
if (![context save:&error]) {
    // Handle error
}
```



# Document Revision History

---

This table describes the changes to *Core Data Snippets*.

Date	Notes
2009-03-04	New document that provides snippets of code that you can use when writing a program that uses Core Data.

