Creating a Managed Object Model with Xcode

Tools & Languages: IDEs

2009-03-02

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

Simultaneously published in the United States and Canada.

Contents

Figures

Introduction

This document shows how to create a Core Data managed object model with Xcode.

A managed object model is an instance of the `NSManagedObjectModel` class. It describes a schema—a collection of entities—that you use in your application. (If you do not understand the term "entity"—or the related terms, "property," "attribute," and "relationship"—you should first read Core Data Basics and the "Object Modeling" section in Cocoa Design Patterns.)

You should read this document to learn how to use the Xcode modeling tool to create a Core Data managed object model. You should already be familiar with the basic architecture of Core Data as described in *Core Data Programming Guide*; it is also helpful to have followed the tutorial in *Core Data Utility Tutorial* to understand how you can create the managed object model programmatically. The data modeling tool itself is described in Data Modeling for Core Data.

Organization of This Document

This document contains the following chapters:

1. "Creating the Model File and the Entities" (page 9)
2. "Adding Properties" (page 13)

See Also

Core Data Programming Guide

Core Data Utility Tutorial

Model Object Implementation Guide

Introduction

Creating the Model File and the Entities

In this tutorial you create a new data model that models two entities—Employee and Department—and the relationships between them. The model includes a number of constraints on the possible values that the entities' attributes and relationships can have. These constraints are automatically checked before any data is saved to a persistent store.

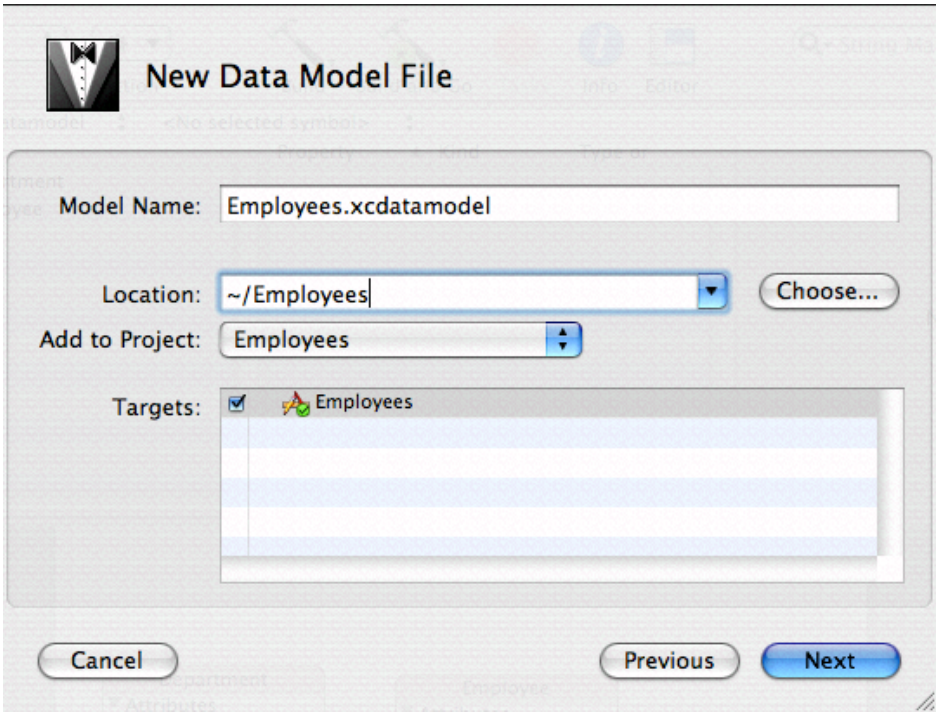
The first task is to create the model file itself in Xcode; the second is to add the entities.

Creating the Model File

First create a new Cocoa-based project. If you create a Core Data based project, a data model is automatically created for you and added to the project. If you choose a Cocoa Application or a Cocoa Document-based Application, or if you have an existing project, then follow the next steps to add a data model to your project.

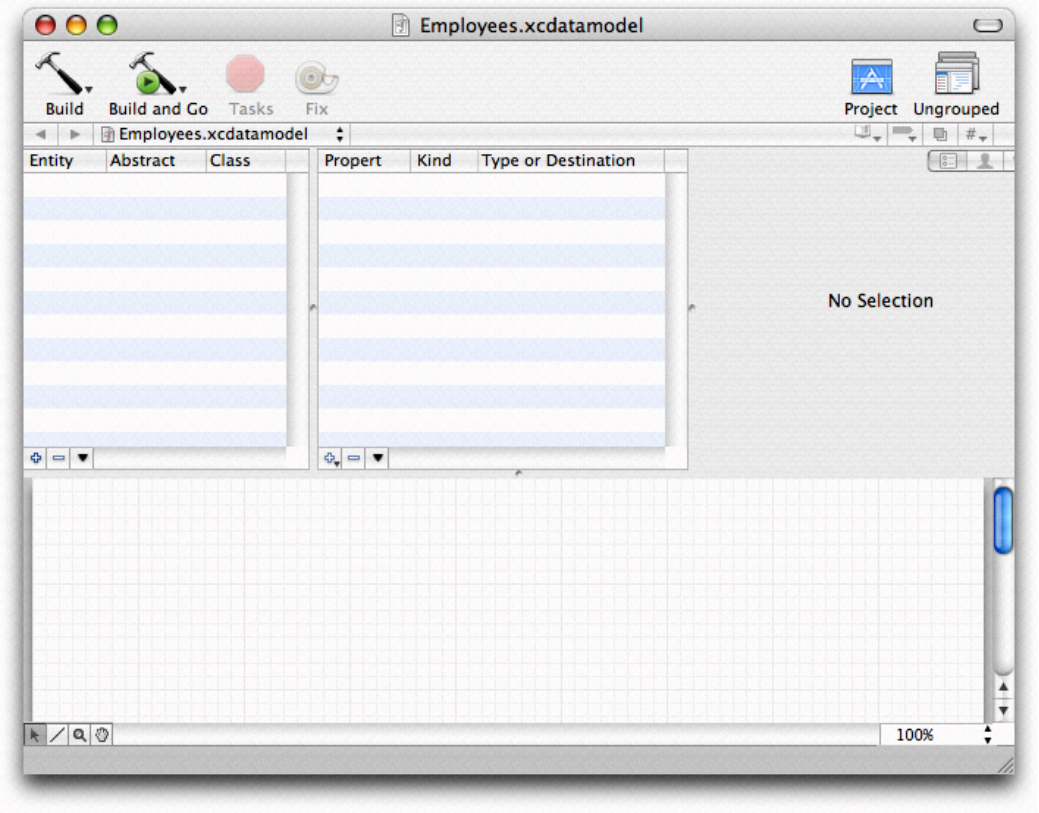
From the File menu choose New File and add a file of type Data Model (from the Design) list. Give the file a suitable name and ensure that the file is added to your application target, as illustrated in Figure 1-1, and press Next.

Figure 1-1 Creating a new persistence model file



On the following panel, do not select any groups or files, simply click Finish. You should now see a new data model window like that shown in Figure 1-2.

Figure 1-2 Empty persistence model file



The window is split into a number of sections with different functions. The table views and buttons in the top section allow you to add, delete, and edit entities and properties of entities. The area on the top right serves as an inspector to allow you to examine and edit a selection in more detail. The bottom section presents a graphical representation of the model. The features are described in more detail in *Xcode Tools for Core Data*.

You can edit the model graphically if you prefer—this task illustrates use of the table views.

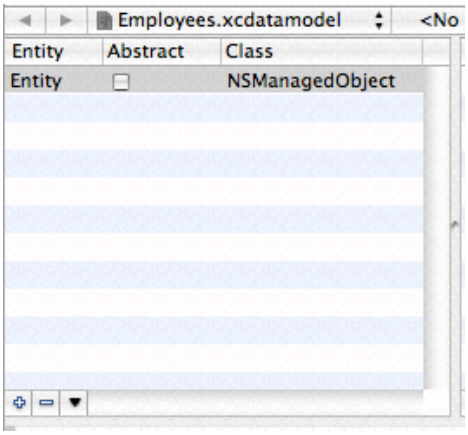
Adding Entities

The schema for the model to create is shown in Figure 1-3 (page 11).

Figure 1-3 Schema for task model

You typically add and edit entities using the entities table view on the left of the browser pane, as illustrated in Figure 1-4. Add a new entity called "Employee" as follows:

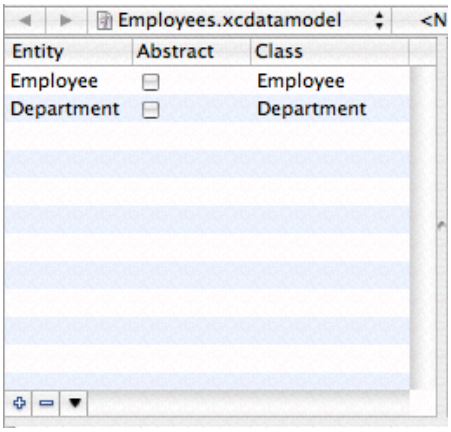
Figure 1-4 Adding an entity to a data model



1. Add a new entity to the model. In the entities table view, click the button with the plus sign. Alternatively, in the Design menu choose Data Model > Add Entity. (When you have completed this step, you should also notice that an node representing the Employee entity is added to the diagram view.)
2. In the table view, double click in the Entity cell in the newly-added row to select the entity name and change it from "Entity" to "Employee."
3. Tab to the Class cell and enter the text "Employee."
4. Do not check the "Abstract" switch.

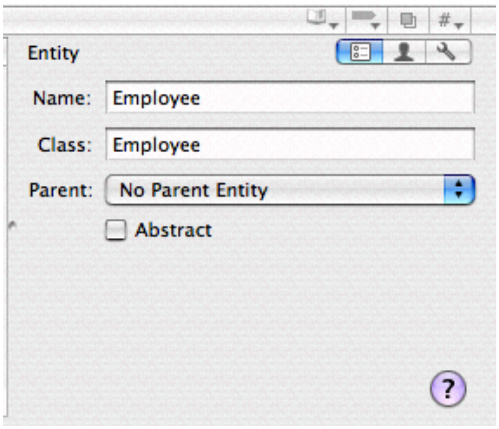
Add a Department entity by repeating steps 1-4, but using "Department" in place of "Employee." Alternatively, add a Department entity using the contextual menu in the diagram view. Note that you will still have to set the class name in the browser. When you have finished, the entities table should look like that shown in Figure 1-5.

Figure 1-5 Model with Employee and Department entities



You have now defined two entities—Employee and Department—and importantly you have specified that in your application these entities are represented by the Employee and Department classes respectively. (If you want to test your model without creating custom classes for these entities, you must specify that the entities are represented by `NSManagedObject`.) The entities are not specified as abstract—you want to create instances of the corresponding classes in your application. To see more details about each entity, look at the detail pane. Ensure that you have selected the General button in the segmented control to the top right of the pane. This displays the entity detail pane, as illustrated in Figure 1-6.

Figure 1-6 Entity detail pane



Leave the remaining values at their default settings. There is no parent entity—the entities do not inherit from any other entity. There is no need to add any user info keys, or to set up any configurations.

You can now add properties (attributes and relationships) to the entities. Note that the name of a property must not be the same as any no-parameter method name of `NSObject` or `NSManagedObject`.

Adding Properties

The next task is to add properties to the entities. There are two sorts of property—attributes and relationships. They share a number of features in common (in their implementation, `NSAttributeDescription` and `NSRelationshipDescription` both inherit from `NSPropertyDescription`), but there are differences in the way you configure them in the tool.

Adding Attributes

When you add an attribute you must specify name and its type—for example, string, number, date, and so on. In addition, you can specify a default value and constraints on values that a property can have—for example the maximum or minimum values of a numeric value, a regular expression that a string value must match, or the cardinality of a relationship. You can specify all these sorts of constraint in the model, and they will be automatically applied when you try to save an instance at runtime. If you want to specify other constraints on property values, or to enforce inter-property constraints, you must implement validation methods in the custom class for the entity. For more details, see `NSManagedObject`.

You can now add attributes to the entities, as described by the following steps.

1. In the entity list, select the Employee entity.
2. Add a new attribute. In the properties pane, ensure that either the Show All Properties or the Show Attributes item is selected from the "v" pop-up menu. Click the plus button to the lower left of the table view, as illustrated in "Adding a property," and choose Add Attribute from the pop-up menu. Alternatively, in the Design menu choose Data Model > Add Attribute.
3. Select the new attribute in the table view. Use the detail pane to set the attributes of the employee's `firstName` attribute as follows: the attribute is neither optional nor transient; its type is String; there are no constraints specified for Max Length or Reg. Ex; Min Length is 2; and there is no default value. When you have finished the pane should look like that shown in "Editing an attribute."
4. Follow steps 1-3 to add a `lastName` attribute to the Employee entity, and then a `name` attribute to the Department entity.

Figure 2-1 Adding a property

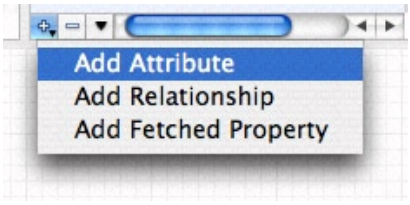
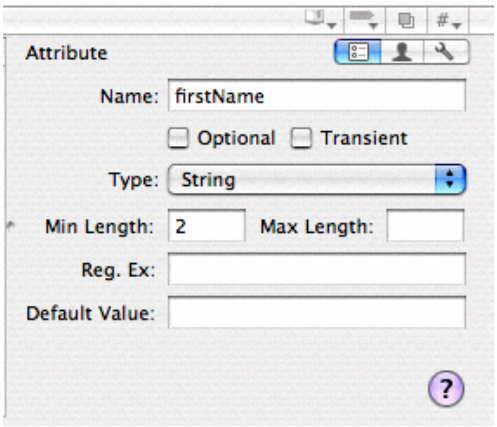


Figure 2-2 Editing an attribute



Follow similar steps to add a `salary` attribute to the Employee entity and a `budget` attribute to the Department entity. These differ from the name attributes in that they are numeric values. Set the type to Decimal, and impose suitable constraints. Both have a minimum value of `0`. Set for each whatever maximum value you think is reasonable.

Adding Relationships

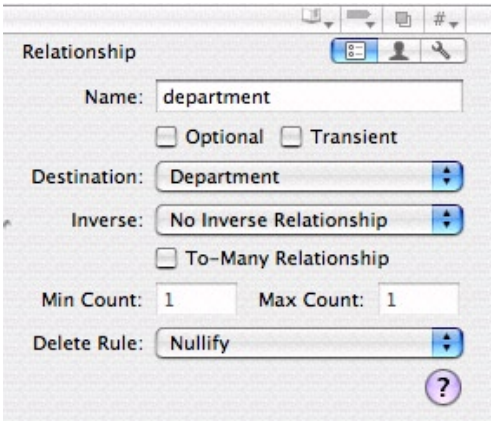
In Core Data, you should almost always model both sides of a relationship. Core Data relies on this information so that it can ensure referential integrity. If you do not model both sides of a relationship, you must do additional work yourself to ensure that your data is not corrupted. Modeling just one side of a relationship is strongly discouraged.

For the current example, start by adding a "department" relationship from Employee to Department, as described by the following steps.

1. In the entity list, select the Employee entity.
2. Add a new relationship. In the properties pane, ensure that either the Show All Properties or the Show Relationships item is selected from the "v" pop-up menu. Click the plus button to the lower left of the table view, as illustrated in "Adding a property," and choose Add Relationship from the pop-up menu. (Alternatively, either in the Design menu choose Data Model > Add Relationship, or—in the diagram view—select the line tool, drag from the Employee node to the Department node.)
3. Select the new relationship in the table view. Use the Property view in the inspector to describe Employee's `department` relationship, as shown in "The Employee entity's department relationship." Note the following:
    - The relationship name is `department`.
    - The destination is Department.
    - Max and Min Count are 1 (and not editable).
    - The relationship is not Optional, and not Transient.
    - The Inverse relationship is currently No Inverse Relationship—it has not yet been created.

- The Delete Rule is Nullify—this means that if an Employee instance is deleted, it is removed from the backwards relationship (once it is specified) from its Department instance.

Figure 2-3 The Employee entity's department relationship

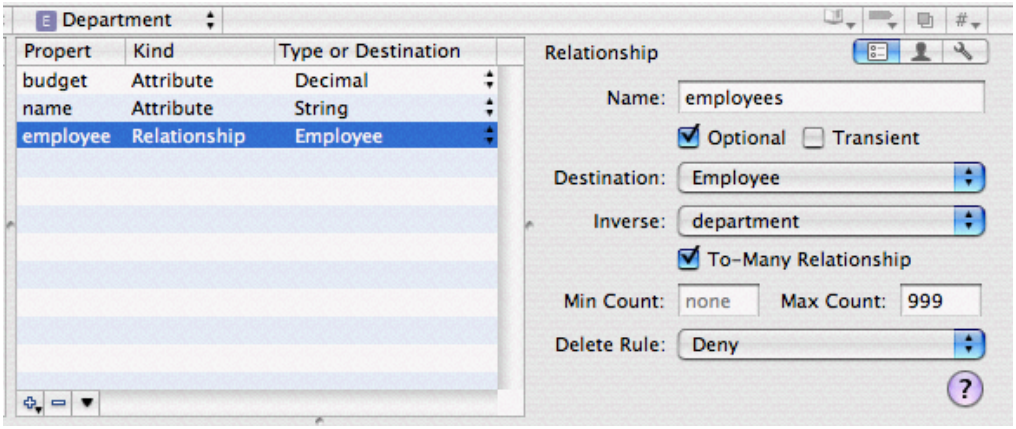


Also note that in the diagram view, an arrow is added that points from the Employee entity to the Department entity.

Now follow these steps to add an `employees` relationship to the Department entity.

1. In the entity list, select the Department entity.
2. Add a new relationship. In the properties section, ensure that button is selected in the segmented control. Click the plus button to the lower left of the table view, as illustrated in "Adding a property," and choose Add Relationship from the pop-up menu. Alternatively, in the Design menu choose Data Model > Add Relationship.
3. Select the new relationship in the table view. Use the Property view in the inspector to describe the department's `employees` relationship, as shown in "Department entity's employees relationship." Note the following:
   - The relationship name is `employees`.
   - The destination is Employee.
   - Min Count is `0` and Max Count is `999`.
   - The relationship is Optional and not Transient.
   - The Inverse relationship is department.
   - The Delete Rule is Deny—this means that a Department cannot be deleted if it still has employees.

Figure 2-4 Department entity's employees relationship



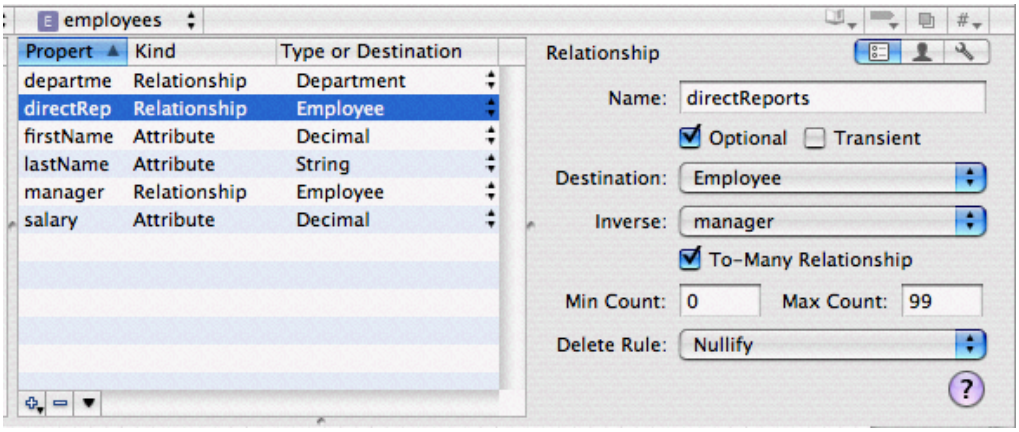
Now that you have added the Department to Employee relationship, you can specify the inverse relationship for the Employee to Department relationship.

1. In the entity list, select the Employee entity.
2. In the properties table view, select the `department` relationship. Use the Property view in the inspector to set the Inverse relationship to `employees`

You have fully specified the relationship between the Department and Employee entities. You can now specify the reflexive relationship in the Employee entity to describe the relationship between a manager and direct reports.

Follow the same steps as before to create two relationships from the Employee entity to itself—that is, the destination entity is Employee. The `manager` relationship is a to-one relationship, the `directReports` relationship is a to-many relationship. Both are optional, and the delete rule for both is Nullify. Remember to specify the inverse relationship for both. When you have finished, the `directReports` relationship should look like the image shown in "Employee's directReports relationship."

Figure 2-5 Employee's directReports relationship



Document Revision History

This table describes the changes to *Creating a Managed Object Model with Xcode*.

Date	Notes
2009-03-02	First release of this document for iOS.
2007-08-07	A new document that shows how to create a Core Data managed object model using Xcode.

Document Revision History