Xcode Tools for Core Data

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

Xcode Entity Modeling Tools for Core Data

The Xcode data modeling tool deals with entities and the relationships between them. You use the tool to define a schema for Core Data. The model ultimately becomes part of your build product and is used by your application at runtime.

Important: This document describes the features of the data modeling tool and how to use it. It does not explain the basic features and functionality of the Core Data framework. You must read the introductory material in *Core Data Programming Guide* before reading this chapter or attempting to use the data modeling tool. For more about specific Core Data classes, see the relevant API reference documentation.

The purpose of the Core Data data modeling tool is to create a data model (or schema) for use with the Core Data framework. At runtime, the model is turned into an instance of `NSManagedObjectModel` with a collection of `NSEntityDescription`, `NSAttributeDescription`, `NSRelationshipDescription`, and `NSFetchRequest` objects. In some respects this is analogous to the behavior of Interface Builder. With Interface Builder, you graphically create a collection of objects that are then saved in a file (a nib file) and re-created at runtime. As with user interface elements, it is possible to create a model directly in code at runtime; however, it is typically easier to do so graphically using the appropriate tool. Similarly, just as it is possible to modify the user interface after it has been loaded, it is also possible to customize a model after it has been loaded. (Note that a model does have a constraint not shared with a nib file: once loaded, a model cannot be modified after it has been used.)

As your application evolves, to accommodate new features you may need to change the schema. Core Data provides an infrastructure for migrating data from one schema (model version) to another—see *Core Data Model Versioning and Data Migration Programming Guide*. To use this infrastructure, you need to define a versioned model, and mappings between model versions. You create a versioned model using Xcode's data modeling tool (see "Model Versioning" (page 37)) and the mapping model using Xcode's mapping model tool.

You should read this document to learn how to use the Xcode entity modeling tool to create a managed object model for a Core Data application. For a task-based example of how to create a data model, see *Creating a Managed Object Model with Xcode*.

Organization of This Document

This document contains the following sections:

- "Creating a Data Model File" (page 9) describes how to create a model file.
- "Workflow" (page 11) describes the basic features of the data modeling tools and how you use them.
- "The Browser View" (page 15) describes the diagram view of the data modeler.
- "The Diagram View" (page 23) describes the diagram view of the data modeler.
- "The Predicate Builder" (page 29) describes the predicate builder.

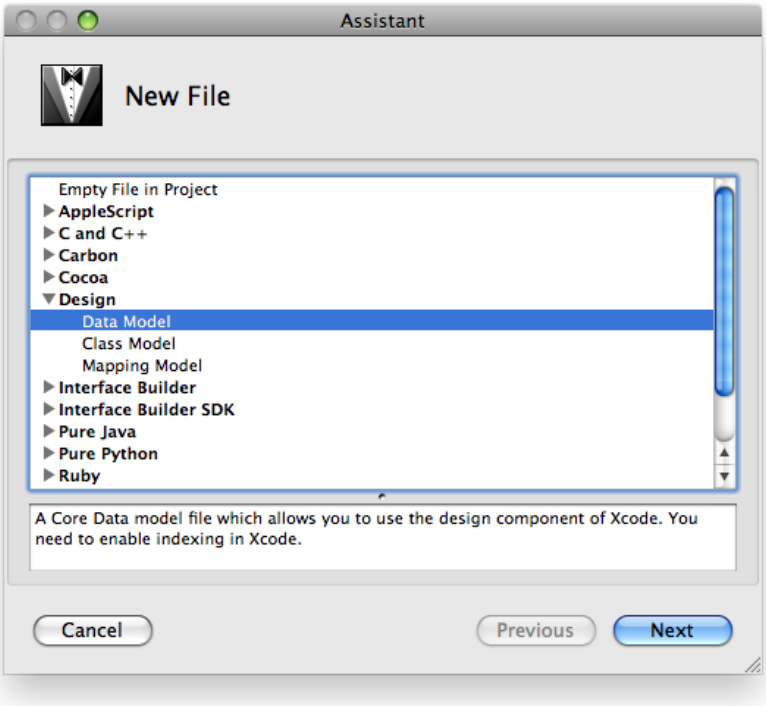
- "Code Generation" (page 35) describes how to generate source code for model entities and their properties.
- "Model Versioning" (page 37) describes how to create a versioned model and how to specify the current version in a versioned model.
- "Compiling a Data Model" (page 39) describes how to compile a data model and what compiler flags are available.
- "Creating a User Interface From a Data Model" (page 41) describes how to use the data modeler in conjunction with Interface Builder to create a user interface.

Creating a Data Model File

This article describes what the data modeling tool is and why you use it, and how you create a new model file.

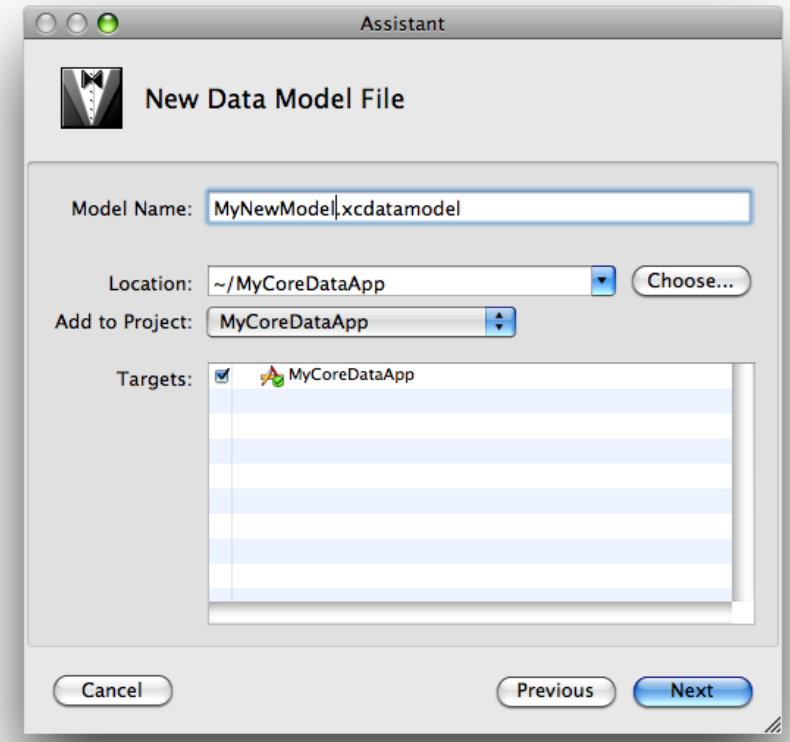
If you create a Core Data–based project, a data model is automatically created for you and added to the project. If you need to create a new model, choose File > New File and in the the New File assistant—shown in Figure 1—select Design > Data Model and press Next.

Figure 1 New File assistant



In the pane that appears (see "The Properties Pane" (page 17)), give the file a suitable name.

Figure 2 Creating a Data Model File



Press Next, and in the following pane select any groups or files that you want to be parsed for inclusion in the model (if any); then click Finish.

If you have an existing compiled (`.mom`) model file (see "Compiling a Data Model" (page 39)), you can import it into a model by choosing Design > Data Model > Import and selecting the `.mom` file in the open panel that is displayed.

Note: Data model files are actually file packages. Make sure you take this into account when setting up source code management (SCM), copying, and so on.

Since the data model is a runtime resource (it is compiled, and deployed, as part of the application), you should add new data models not only to the project, but also to the relevant target(s).

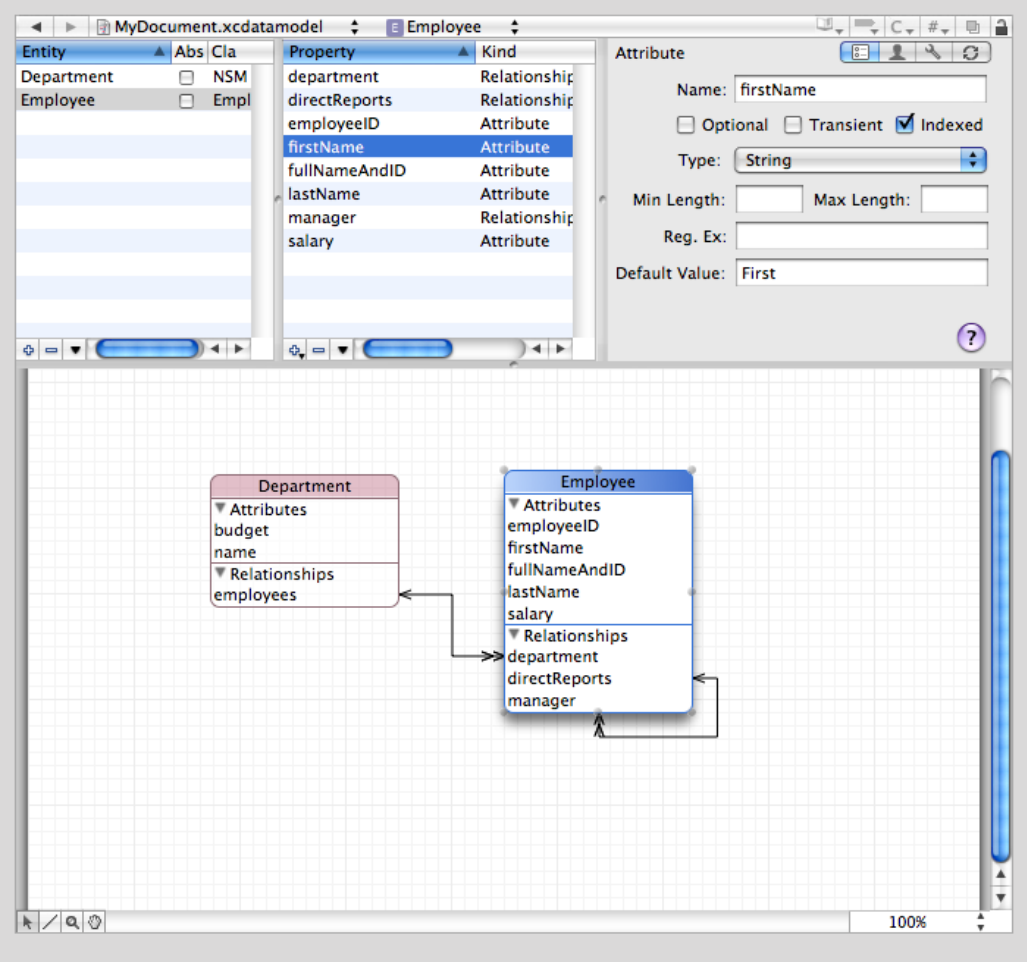
Workflow

This article describes the basic features of the data modeling tools and how you use them.

Basic Features

The data modeling tool employs a browser that you can use to navigate the entity hierarchy, to view the properties of an individual entity or the properties of a collection of entities, and to inspect attributes of a property; and a diagram view that you can use to visualize their contents. Figure 2 shows the entity browser and the diagram view. They are described in greater detail in "The Browser View" (page 15) and "The Diagram View" (page 23).

Figure 1 Browser view and diagram view for a data model



The diagram and browser views have different roles. The diagram view is typically best when you need a high-level overview of the schema. The browser view gives you more detailed information, and it can be especially useful when, for example, you want to edit several attributes simultaneously. When you have large collections of entities, you can minimize the information shown in the diagram (for example, view just entity names and relationship lines) and get the detailed information from browser. The diagram view offers a variety of different configurations, so you can tailor your view to any need.

Workflow

The Xcode design tools offer a wide range of options and features to ease your workflow, from automatic page creation and deletion in the diagram view, to multiple selection editing in the browser. In addition, it is possible to add to the toolbar shortcuts for actions such as Add Entity, and Add Attribute.

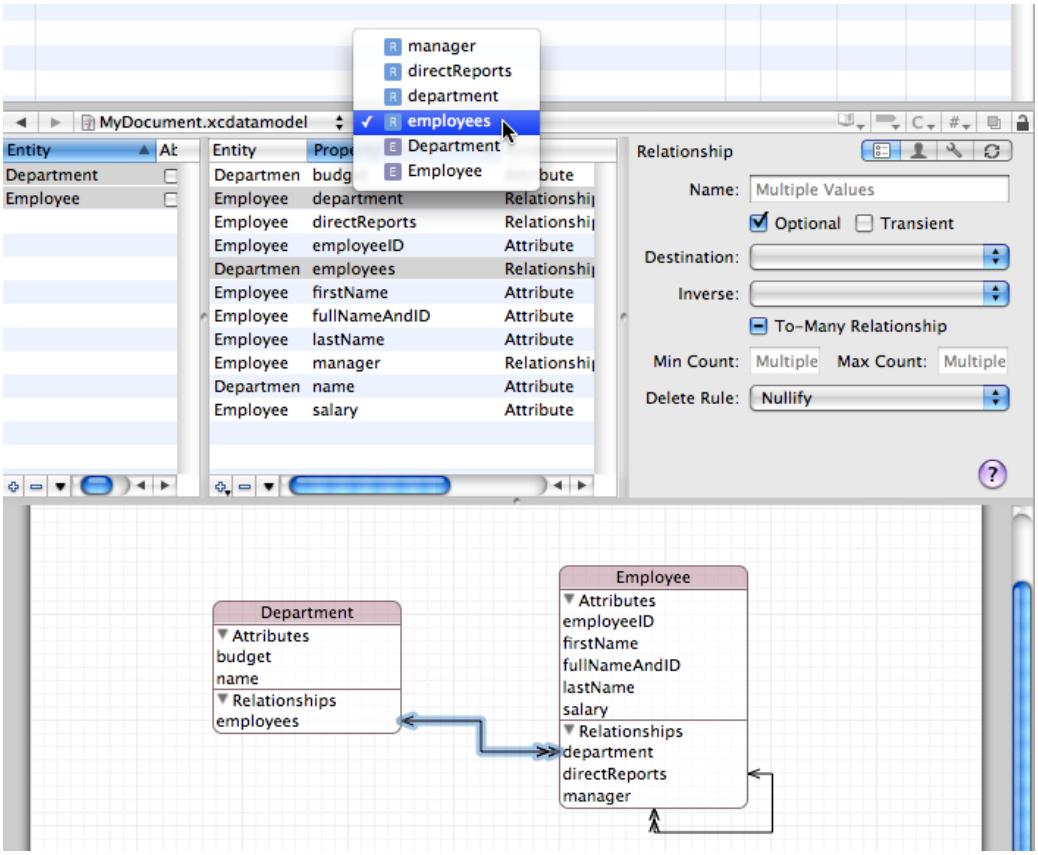
Navigation

You can use the browser and diagram views in conjunction for navigation—the selection in the two views is kept synchronized. As a result, if you make a selection in the browser, the same item is selected in the diagram, and vice versa.

If you want to see a large model in the diagram view, you can maximize the viewable area of a diagram in the main project window by hiding the toolbar, the navigation bar, the status bar, the Favorites bar, and the browser.

If you have a large diagram, there are two strategies you can use to aid navigation. First, you can begin typing the name of the entity you want. As you type characters, Xcode selects the alphabetically “topmost” entity whose name has the prefix you typed. Second, you can use the pop-up menu at the top of the document pane. The pop-up shows a list of elements . When you select an item from the pop-up menu (see Figure 2), the corresponding element is selected and brought into view in the browser and diagram view. This feature may be particularly useful if the browser is hidden.

Figure 2 The Elements pop-up menu



The browser view, however, is useful when you have a large schema with all compartments rolled up and you want to see more details about a given entity but don't want to make the diagram bigger. The browser also shows more information than is available in the diagram (such as the property type).

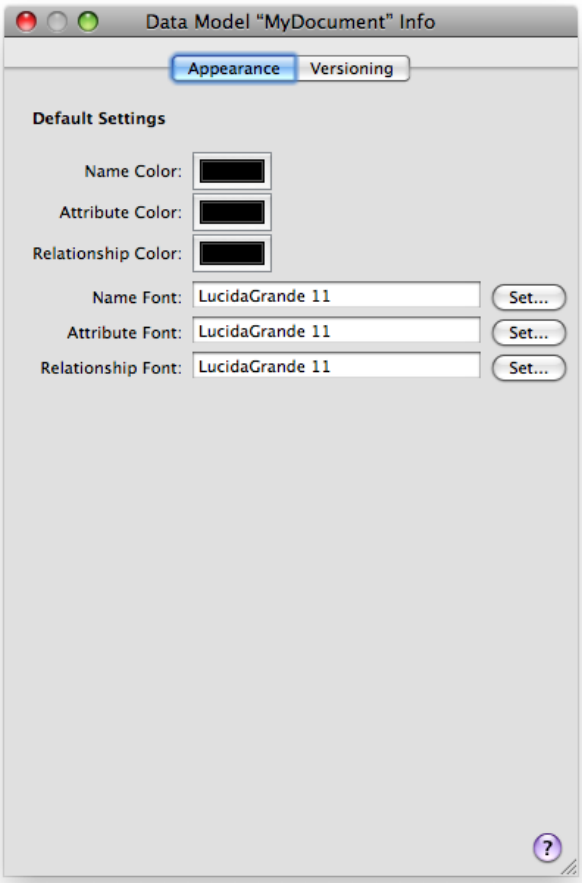
Contextual menus

Most menu-based commands are also available from contextual menus associated with the relevant user interface element. You can Control-click a node for immediate access to operations that apply to it or its context—for example, to expand compartments. You can Control-click the diagram background to perform operations related both to the visual representation and to the model itself. For example, you can hide grid lines, zoom, set the alignment of drawing elements, and add entities.

The Info Window

The Info Window (inspector) contains an Appearance pane and a Versioning pane. You use the Appearance pane to set default colors and fonts for element names and properties. Figure 3 shows an Appearance pane with custom settings. You use the Versioning pane to set the model version identifier—for more about the identifier, see *Core Data Model Versioning and Data Migration Programming Guide*.

Figure 3 Appearance pane



Important: To use the Info window, you must make a selection within the browser or the diagram in the document (you can just click the background of the diagram view, for example), not in the Groups & Files browser (for a quick model there may not even be a file icon). If you select a model icon in the Groups & Files list and then choose Get Info, you get an Info window with General, File, and SCM panes.

The Browser View

The browser view gives you a different perspective on the whole of your model. It has three separate parts: two table view panes—the entity pane and the properties pane—and the detail pane. You can resize a pane by dragging the vertical divider. You can also hide a pane by resizing its width to zero—to hide the detail pane you must drag the divider on its left side most of the way to the right, past its minimum size.

Overview

The browser view is by default always present in the model's editor view—you cannot shrink it beyond a minimum height. You can hide it by choosing Design > Hide Model Browser (and show it again by choosing Design > Show Model Browser). The view has three separate parts: the entities pane, the properties or fetch requests pane, and the detail pane. The detail pane itself has four separate views: the General pane, the User Info pane, the Configurations pane, and the Sync pane, as shown in Figure 1.

Figure 1 The browser view

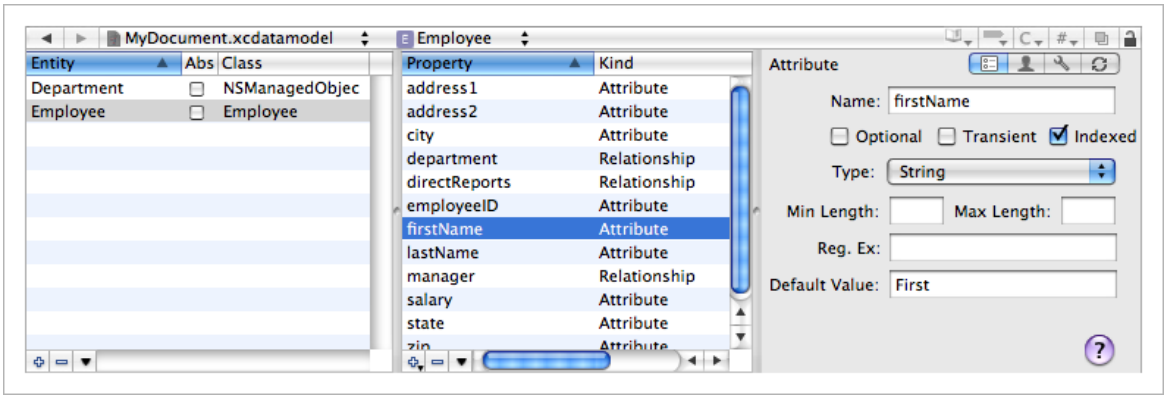


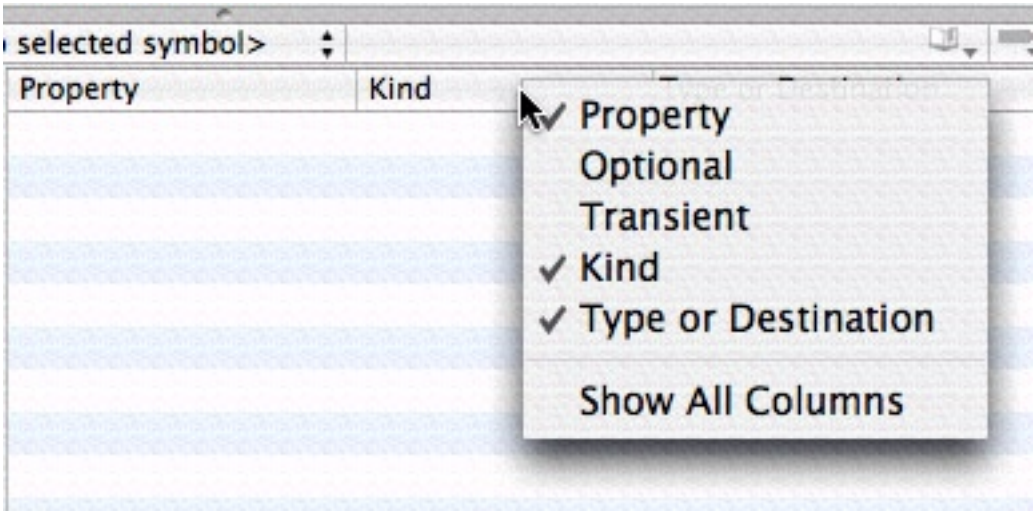
Table view panes

The left-most pane displays and allows you to edit information about the entities that are in the model. The middle pane displays and allows you to edit information about the properties of the currently selected entity. If you make a multiple selection in the entity table, the properties table shows the union of all properties of the selection.

As with most table views, you can rearrange and re-sort the columns. You rearrange the columns simply by dragging a header cell; you can change the sort order by clicking in header cells.

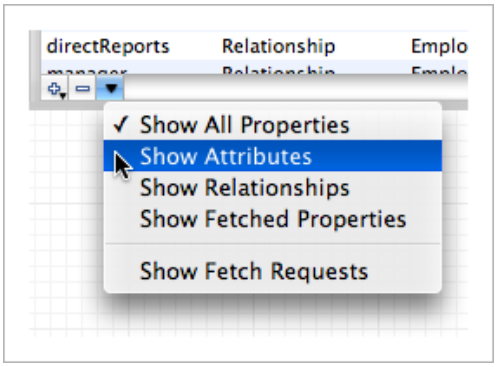
You can choose which columns to see by Control clicking table header cells to display a pop-up list that you can use to toggle the display of columns (see Figure 2). If you have a multiple selection, Show All Columns means that only the set of columns common to all members of the selection may be displayed, otherwise you get a specific set (dependent upon what you have selected).

Figure 2 Browser column options



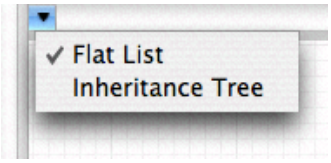
You can also choose which properties are shown by viewing the command pop-down menu in the property table. Each command, shown in Figure 3, acts as a toggle to change the visibility of properties and operations in the table view.

Figure 3 Property list view options



Finally, you can display the entity list either as a flat list or as an inheritance hierarchy. Select the view option you want in the pop-down menu of the leftmost pane, as shown in Figure 4.

Figure 4 Entity view options



Detail pane

The detail pane displays detailed information about whatever was last selected in either the entity or the property table. If you make a multiple selection, the editor shows the best representation it can of the union of the selected items. If you're using the data modeling tool, you can easily apply changes to a number of entities or properties simultaneously.

The Entities Pane

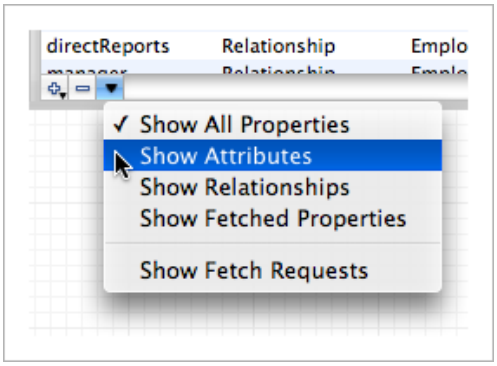
The table in the entities pane (the leftmost pane) lists all the entities in the model, either as a flat list or in an inheritance hierarchy. The table has three columns, showing the entity name, the class used to represent the entity, and a checkbox that indicates whether the entity is abstract. You can edit the entity and class names directly in the text field cells—double click the text to make it editable—and toggle the abstract setting of an entity by clicking the checkbox. For more about entities, see `NSEntityDescription`.

To add a new entity to the model, you click the plus (+) button to the left of the horizontal scroll bar, or choose Design > Data Model > Add Entity. You delete a selected entity or selected entities by clicking the minus (–) button , or by pressing the Delete key.

The Properties Pane

The table in the properties pane lists the properties or fetch requests associated with the selected entities. You choose what features you want to view by choosing from the command pop-down menu shown in Figure 5.

Figure 5 Properties table options



Note that the properties table shows the set of all properties of all entities selected in the entities table. Moreover, you can select and edit multiple properties at the same time so if several entities have a similar property you can change them all simultaneously.

Properties table

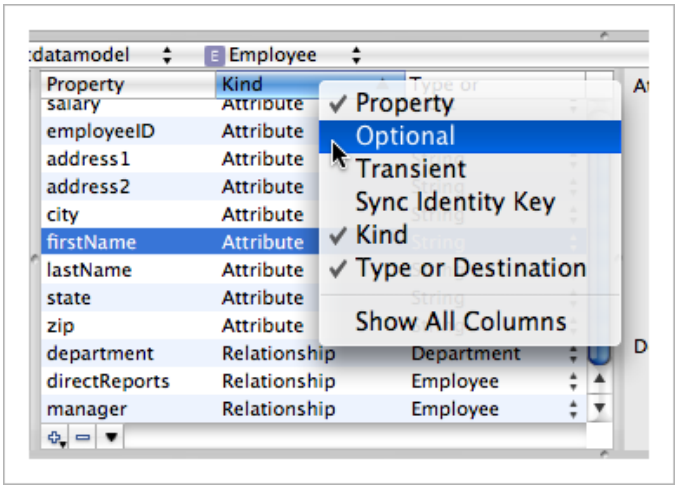
The properties table has up to six columns:

1. A text field that shows the name of the property
2. A checkbox that indicates whether the property is optional
3. A checkbox that indicates whether the property is transient
4. A checkbox that indicates whether the property is a sync identity key
5. A text field that shows the kind of property (attribute, relationship, or fetched)—this text field is not editable
6. A popup menu that shows type (for example, date or integer if the property is an attribute) or destination entity (if the property is a relationship) of the property

Displaying columns

You can add and remove columns from the properties view by Control-clicking the table header cell—this displays a popup menu, as shown in Figure 6; selecting a menu item toggles the corresponding column's visibility.

Figure 6 Property column display options



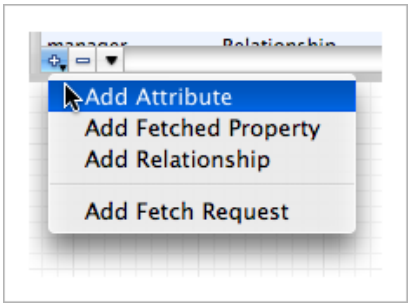
You can re-order the columns in the view by dragging the table header cells. You can order the contents of the table view by clicking on the column header.

Adding properties

There are several ways to add new properties:

- You can choose Design > Data Model and select the appropriate menu item (such as Add Attribute).
- If you have chosen to view just a single type of property, you click the plus (+) button to the left of the horizontal scroll bar to add a new property of the displayed type.
- If you have chosen to view all properties, you click the plus (+) button to the left of the horizontal scroll bar to display the pop-down menu shown in Figure 7). From the pop-down menu, you choose what sort of property you want to add.

Figure 7 Adding a property



Editing properties

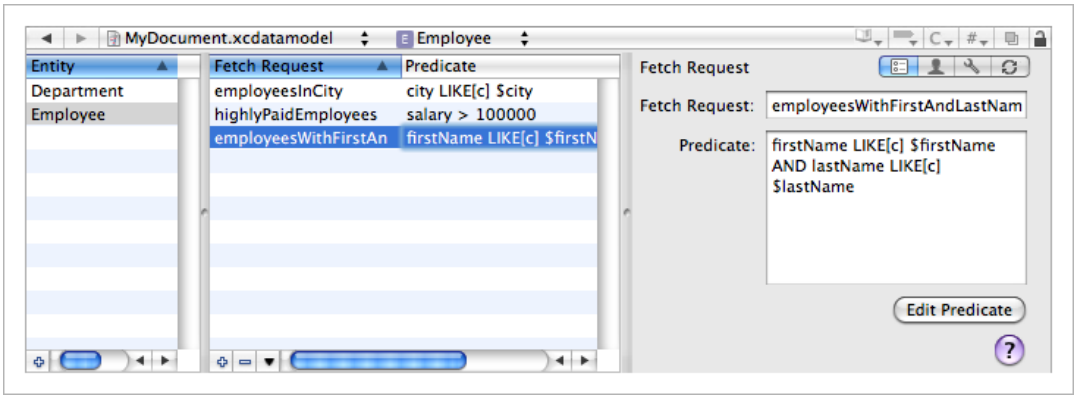
You can edit most property values directly in the properties table—the exception is the property type ("Kind"), which you specify when you first add the property.

You typically edit the predicate associated with fetched properties from the detail pane, using the predicate builder (see "Fetched Properties and Fetch Request Templates" (page 29)). Note that for a fetched property *you must select a destination entity before you edit the predicate*. If you do not do so, the predicate builder does not display any properties.

Fetch requests view

The fetch requests view displays the fetch requests associated with an entity as shown in Figure 8. You add fetch requests using the plus sign button. Although you can edit the name of the fetch request and the predicate directly in the table view, you typically construct the predicate graphically using the predicate builder from the detail pane. For more about fetch requests, see `NSFetchRequest`, and for details about the predicate syntax, see Predicate Format String Syntax in *Predicate Programming Guide*.

Figure 8 Fetch request templates for a selected entity



When you add a fetch request to an entity, you are specifying that that entity is the one against which the fetch will be performed. For example, if you add a fetch request called "highlyPaidEmployees" to the Employee entity, in code you would retrieve it from the model using:

```
NSFetchRequest *fetchRequest = [managedObjectModel
        fetchRequestTemplateForName:@"highlyPaidEmployees"];
```

The entity for the returned fetch request is set to Employee. Since fetch requests are nevertheless general to the model, *fetch request names must be unique across all entities*. If you try to set a duplicate name, you get a warning sheet, and you must choose a unique name before you can proceed.

The Detail Pane

The detail pane itself has four panes: the General pane, the User Info pane, the Configurations pane, and the Synchronization pane. You choose which pane to display by clicking the corresponding element in the segmented control in the upper right of the pane, as shown in Figure 9.

Figure 9 The control for choosing a pane in the detail pane

General pane

The general pane is different for entities, attributes (and for different types of attribute), relationships, and fetch requests. It changes automatically to the appropriate view depending on the last selection. Each view shows, and allows you to edit, details of the selected element.

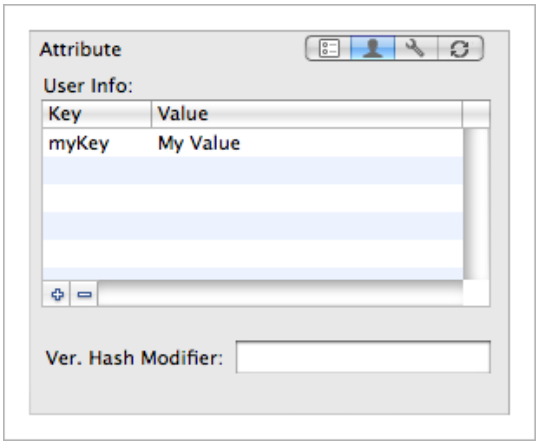
- For entities, you can edit the entity name, the name of the class used to represent the entity, and the parent entity, and you can specify whether or not the entity is abstract.
- For attributes, you can specify the name and type, and whether or not it is optional, transient, or indexed. When you specify the type, the pane updates to allow you to specify other options for the attribute.

For example, for numeric and date attributes you can specify maximum, minimum, and default values; for string attributes you can specify maximum and minimum length, a default value, and a regular expression that the string must match; for transformable attributes, you specify the name of the value transformer to use.

- For relationships, you can specify the name, cardinality, and destination of the relationship. You can also specify a delete rule, and—for to-many relationships—maximum and minimum counts.
- For fetched properties, you specify the name, the destination entity, and the predicate to be used for the fetch. To edit the predicate using the predicate builder, click the Edit Predicate button. For more details about the predicate builder, see "The Predicate Builder" (page 29).
- For fetch requests, you specify the name and the predicate. As with fetched properties, to edit the predicate using the predicate builder, click the Edit Predicate button—see "Fetched Properties and Fetch Request Templates" (page 29).

User Info pane

Entities, attributes, and relationships may have an associated info dictionary that you can retrieve at runtime (see `NSPropertyDescription` and `NSEntityDescription`). The user info pane shows the info dictionary associated with the currently selected model element. The dictionary comprises key-value pairs. You use the info dictionary pane to specify any custom keys and string values that may be of use to you in your application. You add and remove key-value pairs using the plus (+) and minus (–) buttons respectively; you edit the values directly in the table view.

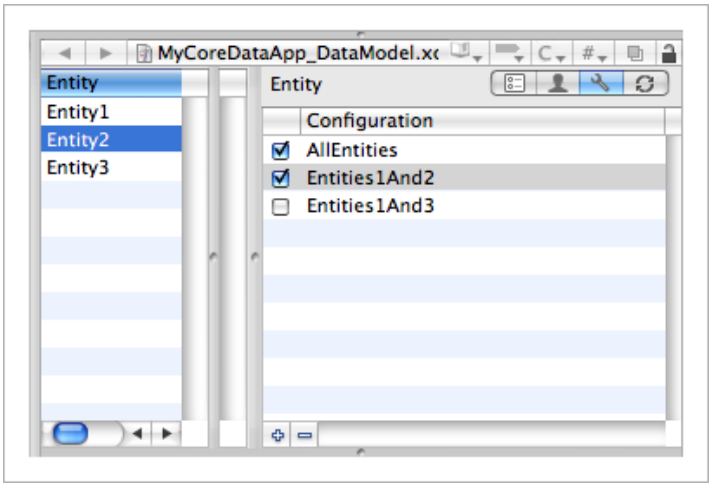


Using the User Info pane, you can also set a version hash modifier for entities and properties. You use a modifier to mark or denote an entity or property as being a different "version" than another even if all of the values which affect persistence are equal. Such a difference is important in cases where, for example, the structure of an entity is unchanged but the format or content of data has changed. For more details, see Versioning in *Core Data Model Versioning and Data Migration Programming Guide*.

Configurations pane

A configuration is a named collection of entities in the model. The configuration pane (show in Figure 10) therefore applies only to entities. You use it to add and remove configurations and to associate entities with configurations. For more about configurations and how to use them, see Managed Object Models.

Figure 10 Configurations pane of the detail pane



A model may have an arbitrary number of configurations. You add configurations using the plus (+) button. Configurations appear in the list for all entities. The checkbox specifies whether the currently-selected entity is associated with the given configuration.

Synchronization pane

You use the synchronization pane to configure entities and properties for use with Sync Services (see *Sync Services Programming Guide*). For more about sync schemas, see Creating a Sync Schema; see also "Adding Information to Managed Object Models" in Syncing Core Data Applications.

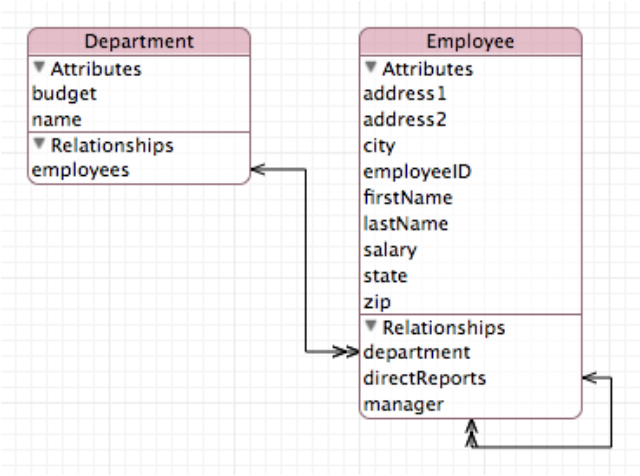
The Diagram View

This section describes the diagram view of the model.

Diagram Elements

The diagram view contains two main elements, rounded rectangles—which represent nodes—and lines, as shown in Figure 1.

Figure 1 A diagram view of an entity model



Nodes

Nodes are the base elements in the model—the entities.

A node may be split into two sections: The title bar containing the name of the entity, and a compartment that shows attributes and relationships (see Figure 4 (page 26)).

Lines

Lines represent relationships between entities and entity inheritance hierarchies.

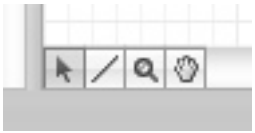
A line with one or two open arrow heads represents a relationship. A single arrowhead denotes a to-one relationship; a double arrowhead denotes a to-many relationship. The direction of an arrow indicates the direction of the relationship—the arrow points to the destination entity. Figure 1 shows an example of the diagram view of a simple data model, with all compartments expanded.

A line with a closed arrowhead denotes inheritance.

Diagram Tools

The diagram view provides several tools, whose function should be familiar from other drawing packages. You select the tools from the palette in the bottom-left corner of the diagram view, shown in Figure 2.

Figure 2 Diagram tools



- **Arrow**. You use the arrow tool to make selections and to move and resize graphic elements.
- **Line**. You use the line tool in data model to add a relationship. To connect two elements, select the line tool, then drag from one end of the connection to the element at the other end. You make the connection from the source to the destination of the relationship.
- **Magnifying glass**. You use the magnifying tool to zoom into part of the diagram or, by holding down the Option key, to zoom out. See "Layout" (page 25) for other ways to zoom. To effect the zoom, you select the tool, then click inside the diagram.
- **Hand**. You use the hand tool to move the diagram if its bounds extend beyond the current view.

Editing the Model

You can edit the model directly from the diagram:

- To add a new entity, you Control-click the background of the diagram and select Add Entity from the contextual menu.
- To add properties to an entity, you Control-click within its node and select the appropriate item from the contextual menu.
- To delete entities and properties, you select the item then press the Delete key.
- To establish new relationships, you select the line tool then drag from the source node to the destination node.

  Note that although most relationships are implicitly bidirectional, relationships do not have to be modeled in both directions—although unless you have very good reason not to model a relationship in both directions, you are strongly encouraged to do so. To specify a bidirectional relationship, you must model both sides of the relationship. Moreover, within the model, you must specify which relationships are the inverse of each other. To do this you need to use the model browser.

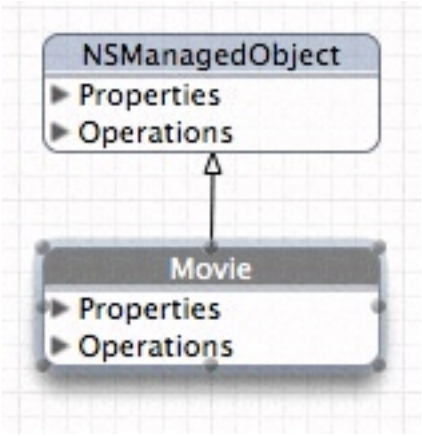
Layout

There are a number of options for moving and resizing elements; you can also constrain the way the elements can be moved and resized, and even prevent them from being moved and resized. Furthermore, you can zoom into and out of the diagram and arrange the page layout as you wish.

- **Moving and resizing shapes**. You can rearrange elements in a diagram to suit your needs—lines that join elements are updated appropriately. Use the arrow tool to select an element, and then simply drag it. You can move all the elements in the current selection (see "Multiple Selection") in the same way.

  When you select a shape, "handles" appear around its edges (as shown in Figure 3). You can drag the handles to resize the shape.

  Figure 3 Diagram view showing element handles

  You can also automatically resize elements in several ways, by choosing the Design > Diagram > Size. In the Size submenu, Make Same Width and Make Same Height resize the selected elements appropriately; Size to Fit resizes the selected elements so that they fully enclose their contents with minimal padding.

- **Multiple selection**. You can use multiple selection in the diagram view to move a collection of elements in a flotilla drag, or for roll-up, expand all, and so on. You can make multiple selection in several ways:

  - You can select a single element, then hold down the Shift key and click additional elements. Unselected elements are added to the current selection; selected elements are removed from the current selection.
  - You can drag the background of the diagram to create a selection rectangle. Elements whose boundaries intersect with this rectangle are selected.
  - You can select entities in the browser—the browser selection and diagram selection are kept synchronized.

  Choose Edit > Select All to select all elements in the diagram. Note that for items in the Diagram menu, clicking on the background (rather than on a drawn element) is the equivalent of selecting all but may be faster.

- **Alignment and grid**. You can use a variety of options to automatically align selected elements and to help you keep elements aligned. By choosing Design > Diagram > Alignment menu, you can perform a number of operations—aligning specified edges or centers of a selection and aligning a selection in a row or column.

  You can also use a grid to help keep elements aligned. By default, the diagram view displays a background grid, and move and resize operations are snapped to it. By choosing Design > Diagram, you can turn the grid display on and off; you can also independently turn the snap-to-grid feature on and off.

- **Locking**. You can lock individual graphic elements in place by choosing Design > Diagram > Lock, or the Lock contextual menu. If you subsequently apply automatic layout, locked elements are unaffected. To unlock an item, choose Design > Diagram > Unlock, or use the Unlock contextual menu.

- **Zoom**. You can zoom into and out of the diagram in three different ways:

  - Choose Design > Diagram to zoom in, out, and to fit.
  - Use the pop-up menu to select a percentage zoom.
  - Use the magnifying glass tool (click to zoom in; Option-click to zoom out).

- **Page layout**. If you move diagram elements outside the current diagram bounds (whether directly, or through applying automatic layout, or by unhiding elements), the page area automatically expands. Conversely, if you remove elements such that a page is left blank, the page area automatically contracts.

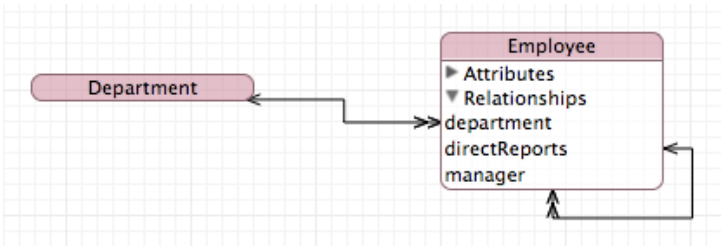
  To adjust the size of a page, choose File > Page Setup. The page layout adjusts automatically to accommodate a change in page size.

Roll-Up and Expansion

You can display a node and the compartments within it in a variety of ways:

- Rolled up, so that just the name of the entity is showing. This gives the most compact representation, with maximum information density in the diagram. (In Figure 4 the Department node is rolled up.)
- Compartment titles showing. The titles are Attributes and Relationships in the data model. This gives a compact representation but with easy access to detail.
- Compartments expanded. All the information in a compartment is visible but at the cost of screen real estate. (In Figure 4 the Relationships compartment of Employee is expanded, but the Attributes compartment is not.)

Figure 4 A rolled-up node and a partially expanded rolled-down node



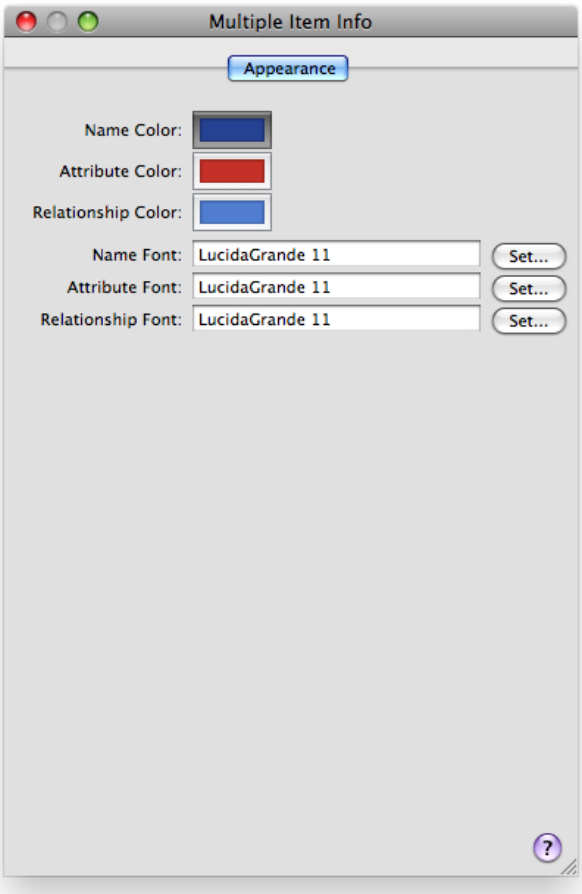
To roll up or roll down the node, choose, respectively, Design > Roll Up Compartments or Design > Roll Down Compartments. To hide or expose compartment information, you use the disclosure triangle within a compartment or choose Design > Expand Compartments (or Design > Collapse Compartments).

Colors and Fonts

The diagram view provides default coloring for various elements. By default, all text is black, and the title bar and outline of drawing elements are colored. In the data model the color is the same for all entities.

You change the background color of the title bar and color of the outline of elements by dropping a color swatch from the Color panel onto the element. You change the other color settings, and the font used for the title, property, and operations text, using the Appearance pane of the Info window (inspector). You can also select multiple elements and change their color and text settings simultaneously, as shown in Figure 5.

Figure 5 Appearance pane showing multiple selection



You can also change the default settings for the entire model using the Appearance pane—see "The Info Window" (page 13).

The Predicate Builder

You use the predicate builder to create predicates for fetched properties and for fetch request templates. As with the rest of the modeling tool, the predicate builder simply provides a graphical means of defining a collection of objects that you could otherwise create programmatically.

Fetched Properties and Fetch Request Templates

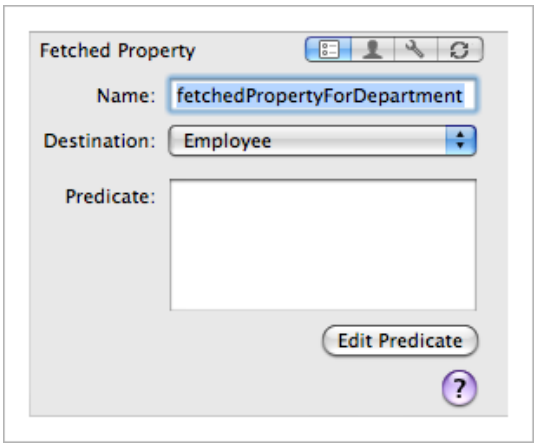
Fetched properties are weak, unidirectional relationships from one entity to another, defined by a fetch request. For more about fetched properties, see `NSFetchedPropertyDescription`. Note that fetched property names must begin with a lowercase letter.

Fetch request templates allow you to create predefined instances of `NSFetchRequest` that are stored in the model. You can either define all aspects of a fetch or you can allow for runtime substitution of values for given variables. Fetch templates are associated with the entity against which the fetch will be made, that is, instances of which the fetch will return. For more about fetch request templates, see Creating Predicates and the reference documentation for `NSManagedObjectModel`. For more about predicates in general, see *Predicate Programming Guide*.

The Predicate Builder

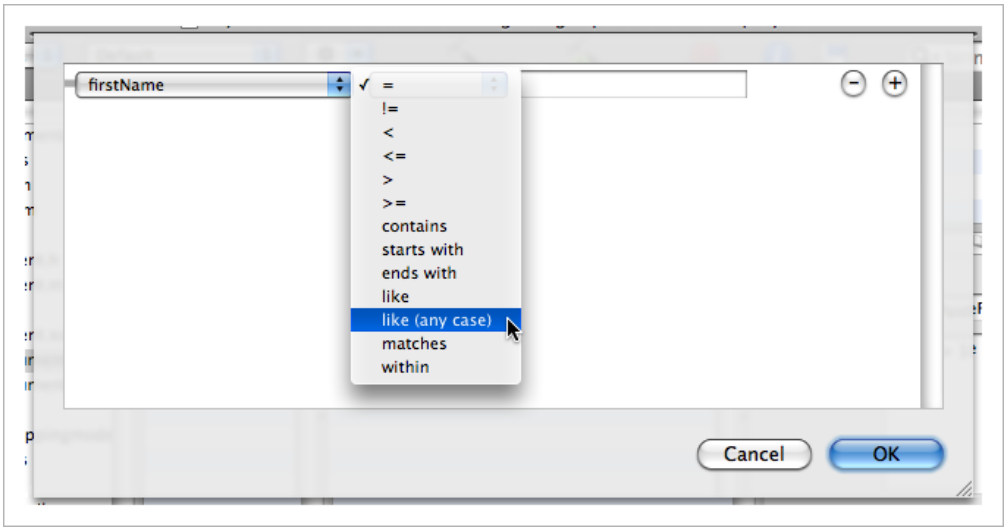
You access the predicate builder by clicking the "Edit Predicate" button that appears in the detail pane for fetched properties and for fetch request templates (see "The Browser View" (page 15) and Figure 1 (page 15) respectively).

Figure 1 Fetched property detail pane



Using the predicate builder, you can build predicates of arbitrary complexity. The initial display shows a simple comparison predicate. The left-hand side is pop-up menu that allows you to choose the key used in a key path expression, the right-hand side is a text field that allows you to specify a constant value, and between them is a pop-up menu that allows you to choose a comparison operator. Note that *for a fetched property you must specify the destination entity before you edit the predicate*, otherwise the predicate builder does not display any properties.

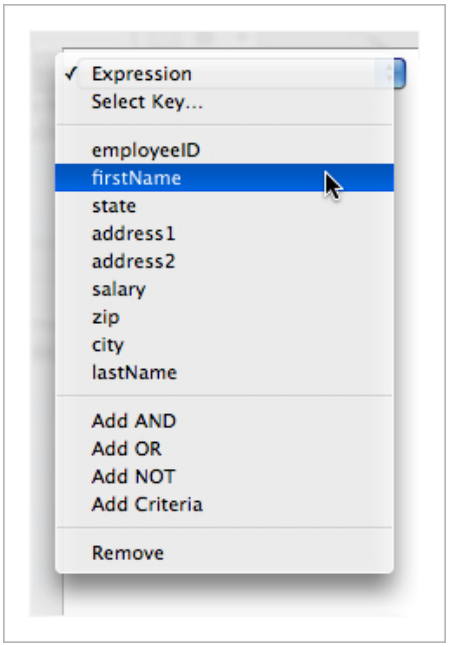
Figure 2 The predicate builder



Left-Hand Side

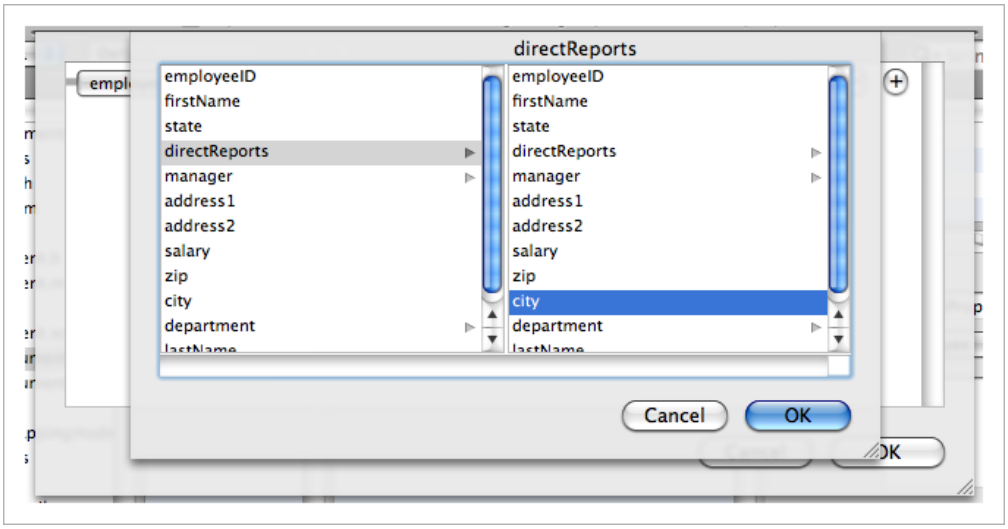
The key pop-up menu, shown in Figure 3, displays attributes of the entity with which the predicate is associated. You can choose Expression to replace the pop-up menu-based expression editor with a free-form text field editor—see "Properties table" (page 18).

Figure 3 Predicate keys



To use a key path (that is, to follow relationships), choose the Select Key item in the key pop-up menu. This displays a browser, shown in Figure 4, from which you can choose the key or key path you want.

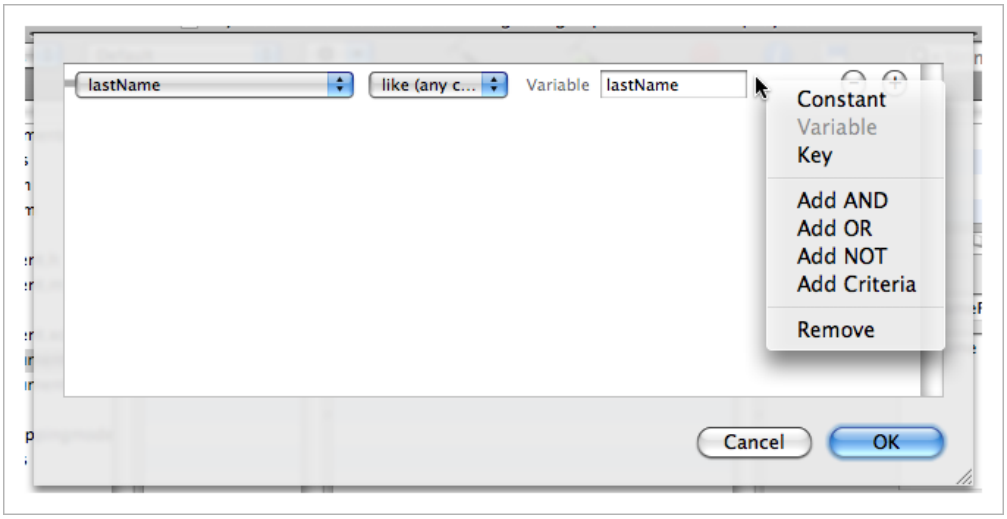
Figure 4 Adding a key path



Right-Hand Side

You change the type of the right-hand side expression using the contextual menu shown in Figure 5 (you must Control-click "empty space" in the line of the criteria—for example, at the end of the line or between the pop-up menus). This changes the constant value field into a variable field or a key pop-up menu as appropriate.

Figure 5 Right-hand side expression type

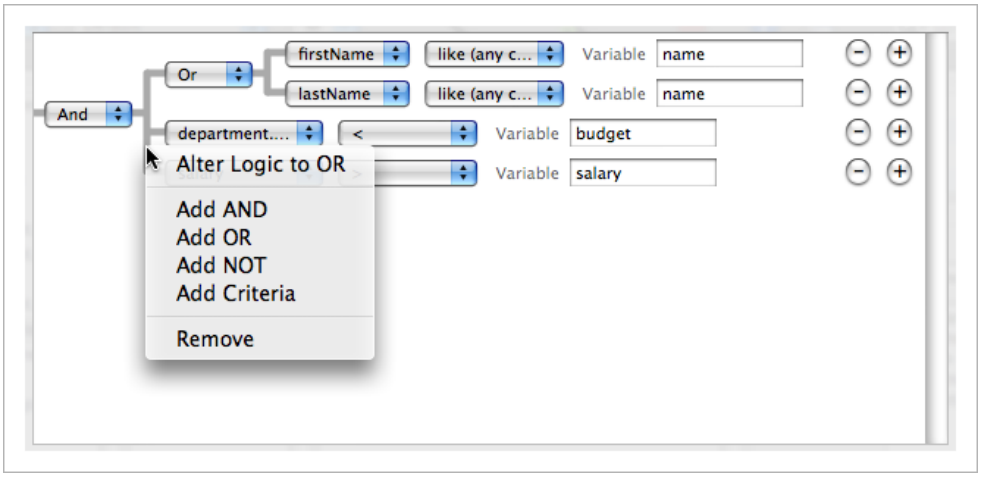


In addition to a constant value, you can also define the right-hand side of a comparison predicate to be a variable or another key. You use this if you are creating either a fetch request template that requires substitution variables or defining a fetched property and need to use the `$FETCH_SOURCE` or `$FETCHED_PROPERTY` variables in the predicate.

Compound Predicates

You can add logical operators (AND, NOT, and OR) to create compound predicates of arbitrary depth and complexity. To add a specific logical operator, use the contextual menu shown in Figure 6.

Figure 6 Creating a compound predicate



You can also add peer predicates by either clicking the plus (+) button, or choosing Add Criteria in the contextual menu—these add an AND operator. You can change the logical operator using the pop-up menu.

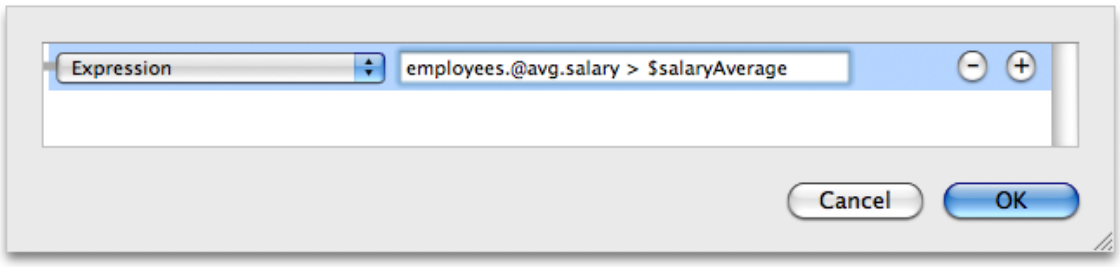
You can rearrange the predicate hierarchy by dragging a row or part of the predicate hierarchy.

To remove a predicate, click the minus (–) button, or use the contextual menu. The predicate builder will try to rebuild the remaining predicate as it can, removing comparison operators where appropriate.

Expressions

To replace the pop-up menu-based expression editor with a free-form text field editor, you can choose Expression from the left-hand side key pop-up menu shown in Figure 3 (page 31). The text field allows you to create more complex expressions that include, for example, functions such as `sum` and `avg`. "Displaying columns" (page 18) shows a simple predicate that you could use for a fetch request template to fetch departments where the sum of the salaries of the employees is greater than a given amount.

Figure 7 The expression editor



You would use the template as shown in the following code fragment.

```
NSFetchRequest *fetchRequest = [theManagedObjectModel
        fetchRequestFromTemplateWithName:@"departmentsExceedingSalaryAverage"
```

```
        substitutionVariables:[NSDictionary dictionaryWithObject:salaryAverage
 forKey:@"salaryAverage"]];
```

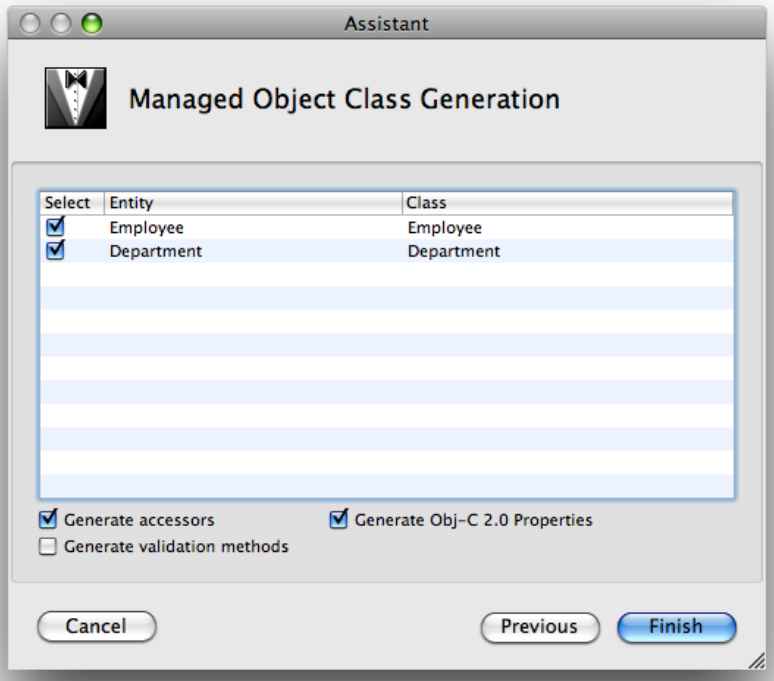
For more about fetch request templates, see Creating Predicates and the reference documentation for `NSManagedObjectModel`. For more about fetched properties, see the reference documentation for `NSFetchedPropertyDescription`.

Code Generation

For each entity in the model, you specify a class that will be used to represent it in your application. By default, the class is set to `NSManagedObject`, which is able to represent any entity. Typically, at the beginning of a project, you use `NSManagedObject` for all your entities. Later, as your project matures, you define custom subclasses of `NSManagedObject` to provide custom functionality.

You can use the New File Assistant to create a default implementation of a managed object class. First, select an entity or a collection of entities in the model, then choose File > New File. In the file type outline view select Design > Managed Object Class and press Next. (If you have not selected any entities, you do not see the entry for Managed Object Class.) In the subsequent pane select the appropriate project and targets, then again press Next. In the following pane (see Figure 1), select the entities for which you want Xcode to generate default class implementations. Check the relevant boxes to specify whether or not the implementations should contain custom accessor, validation methods, or Objective-C properties (see Declared Properties). When you press Finish, Xcode creates the files you specified.

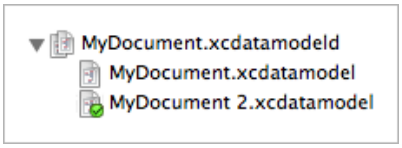
Figure 1 The Managed Object Class Generation pane



Model Versioning

A managed object model can contain multiple schema versions. (For more about model versioning, see *Core Data Model Versioning and Data Migration Programming Guide*.)

To create a versioned model, select a model file and choose Design > Data Model > Add Version. This converts an existing `.xcdatamodel` file into a `.xcdatamodeld` directory containing the original model and a copy of the original model with " 2" appended to the filename.



You can add more versions using Design > Data Model > Add Version.

The current version of the model is denoted by a green check mark on the file symbol. You can change the current version by selecting a different model and choosing Design > Data Model > Set Current Version.

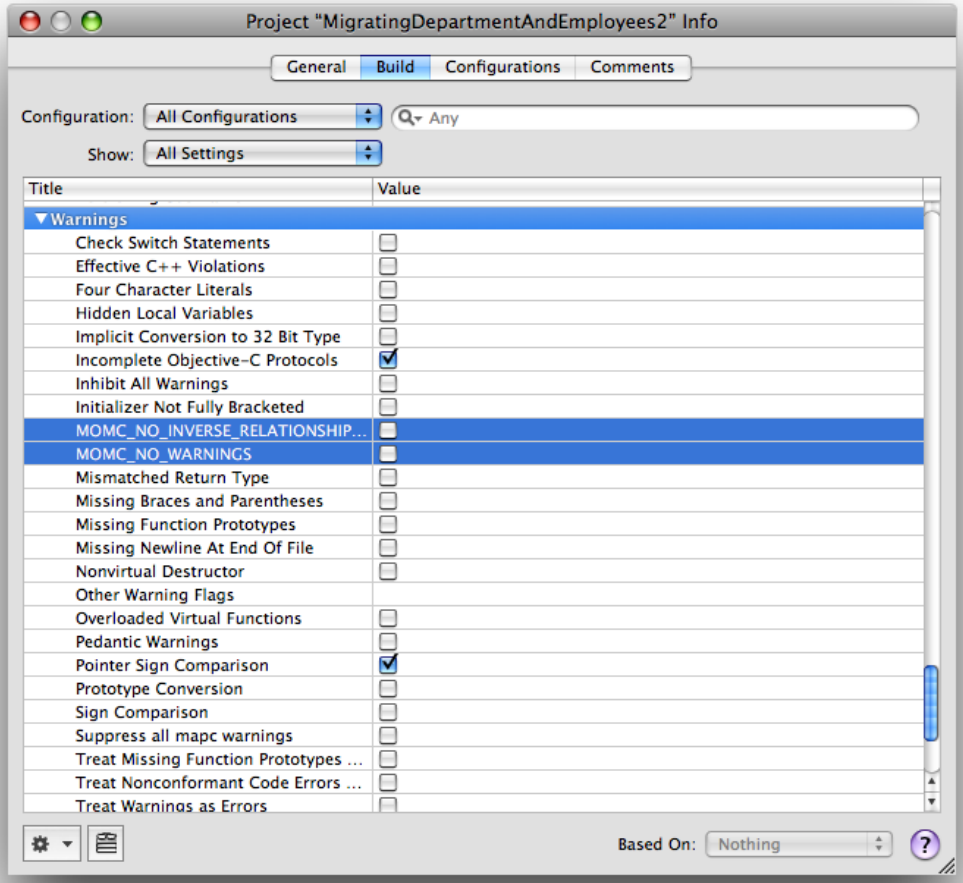
Compiling a Data Model

A data model is a deployment resource. A data model must not only be a project file, it must be associated with the target that uses it. In addition to details of the entities and properties in the model, the model contains information about the diagram—its layout, colors of elements, and so on. This latter information is not needed at runtime. The model file is compiled to remove the extraneous information and make runtime loading of the resource as efficient as possible.

The model compiler, `momc`, is located in `Library/Xcode/Plug-ins/XDCoreDataModel.xdplugin/Contents/Resources/` in the Developer directory. If you want to use it in your own build scripts, its usage is `momc source destination`, where *`source`* is the path of the Core Data model to compile and *`destination`* is the path of the output mom file.

The compiler can generate warnings for various model configuration problems (such as unidirectional relationships). You can toggle these warnings by checking the appropriate boxes in the Warnings section of the project Build panel, as shown in Figure 1.

Figure 1 Project build panel showing `momc` warning flags



Creating a User Interface From a Data Model

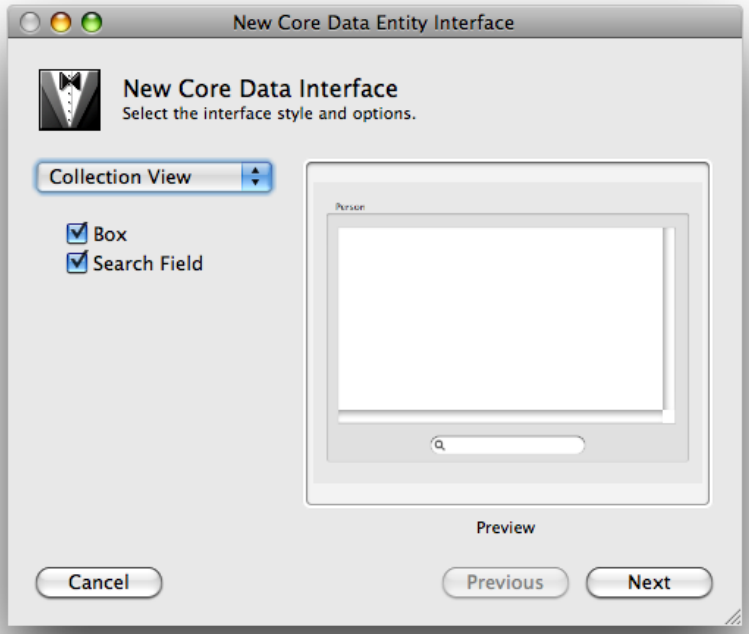
You can use the Xcode modeling tool to quickly create a user interface for managing entity instances in a Cocoa application. This provides a useful strategy for testing a model—with little effort you can create an application to use for testing.

1. Open a nib file (from your project) in Interface Builder, and ensure that you can see the user interface window in which you want the user interface to be created.

2. In Xcode, click an entity node in the diagram view of the data modeling tool.

3. Option-drag the entity node to the user interface window so that a cursor appears showing a "+" symbol. (You must make sure that Xcode is the foreground application when you start to do this—Option-clicking Xcode while it is not foreground makes it foreground and hides all other applications, including Interface Builder.)

4. Release the mouse. You will be presented with an alert asking you to select the interface style and options. From the popup menu, you can select an interface for a single item, a master/detail view, or a collection view. Choose whichever is appropriate.



Each interface style has a different set of options, for example the collection view allows you to add a box and a search field.

Interface Builder automatically creates a user interface appropriate for the selection you made. For example, if you select Master/Detail, the interface contains a table view, a search field, text fields for individual attributes, pop-up menus for to-one relationships, and optionally buttons to add, remove, and fetch instances of the entity. Object controllers are also added to the nib file to manage collections of entities as appropriate. (Recall that object controllers that contain managed objects use the entity name, and not the name of the class. If at a later stage in the development cycle you specify and implement a custom class for an entity, the interface will continue to work.)

Document Revision History

This table describes the changes to *Xcode Tools for Core Data*.

Date	Notes
2009-03-02	First release of this document for iOS.
2008-04-15	New document that describes the Xcode modeling tools for Core Data.