Xcode Design Tools for Class Modeling

Tools > Xcode

2009-02-04

Contents

Figures

Introduction to Xcode Design Tools for Class Modeling

Xcode differs from other design tools in its rich control over how models are displayed. Other IDEs that provide a graphic class browser typically give you little control over display—you see what it wants to show you. With Xcode, you have coarse-, medium-, and fine-grained control over what is displayed and how. You can edit the diagram in the same way you might edit in common graphics editors. For example, you can color, move, and align elements to customize their appearance to meet your needs. You can zoom in and out of the diagram; and choose whatever page and grid sizes you want. Moreover, your models never become stale.

Organization of This Document

The document contains the following chapters:

- "Class Modeling With Xcode Design Tools" (page 9) gives an overview of the design tools and why you might want to use them.
- "Creating Models" (page 11) describes the ways you create a model.
- "Workflow" (page 13) describes how to use the tools.
- "The Info Window" (page 17) explains the roles of the Info window with the design tools.
- "The Browser View" (page 21) describes the browser view.
- "The Diagram View" (page 25) describes the diagram view.

Class Modeling With Xcode Design Tools

The Xcode class modeling tool helps you to explore and understand the classes in your project, whether they're written in Objective-C, C++, Java, or a mixture of those languages. It allows you to see class relationships (subclass and superclass relationships—including support for multiple inheritance in C++), protocols (or interfaces in Java), and categories. In the diagram view, color and text coding help you to quickly distinguish between classes, categories, and protocols, and between project and framework code. The visibility (public, private, protected) of member functions and variables is shown appropriately. (If you are not familiar with any of these terms, you should consult suitable programming texts.)

Class Modeling

Class modeling allows you to understand and explore the classes in your project, whether they're written in Objective-C, C++, Java, or a mixture of those languages. You can get a bird's eye view of your project structure, look at relationships among classes, or scan quickly through class member and method types, parameters, and return values. You can use class modeling to augment or replace the Xcode class browser. The class model is saved with your project (you can even commit it to your repository), so other team members can get an overview of your code structure from the class model.

You can use the tool to visualize and browse class hierarchies not only in terms of the class relationships (subclass and superclass), but also in terms of the protocols (or interfaces in Java) a class implements, and the categories that are present. You can even add comments to call out notable things about specific classes.

Unlike most other modeling tools, Xcode lets you control the set of files (groups, targets, and so forth) that are modeled, the position and layout of the classes in the diagram, and even the classes that are shown. Moreover the Xcode class information is always up to date. Class models always represent the actual classes in files, groups, and targets in your project, and are automatically updated as you change your source code—even if you add, remove, or refactor classes.

Why Are Modeling Tools Useful?

There are a number of reasons why modeling tools are useful. You can use the tool as an index into your project. From within the tool, you can navigate to the source code of your own classes (both the declaration and implementation), to the declaration in framework classes (those for which you do not have source code), and to corresponding documentation. You can create models that persist as part of your project to communicate design details to other team members, and you can create temporary models (that is, quick models) that serve to illuminate an immediate problem.

A graphical representation of your project gives you a better conceptual overview of your project than raw XML or a mass of source files. In particular, with Xcode, it gives you a developer's-eye view of your project, not a computer-generated representation. You can customize the view to see the information you need, not what the computer thinks is important or requires you to see. This is especially useful for communicating between members of a team, or for homing in on a specific aspect of a project. In addition, the class modeling tool may also be useful for learning about the functionality provided by existing libraries.

In terms of navigation, the browser view gives you an alternative means of navigating through your source, following relationships where appropriate. It provides a compact representation and summary of the classes in your project, including their properties and behaviors, and an easy way to get to relevant documentation.

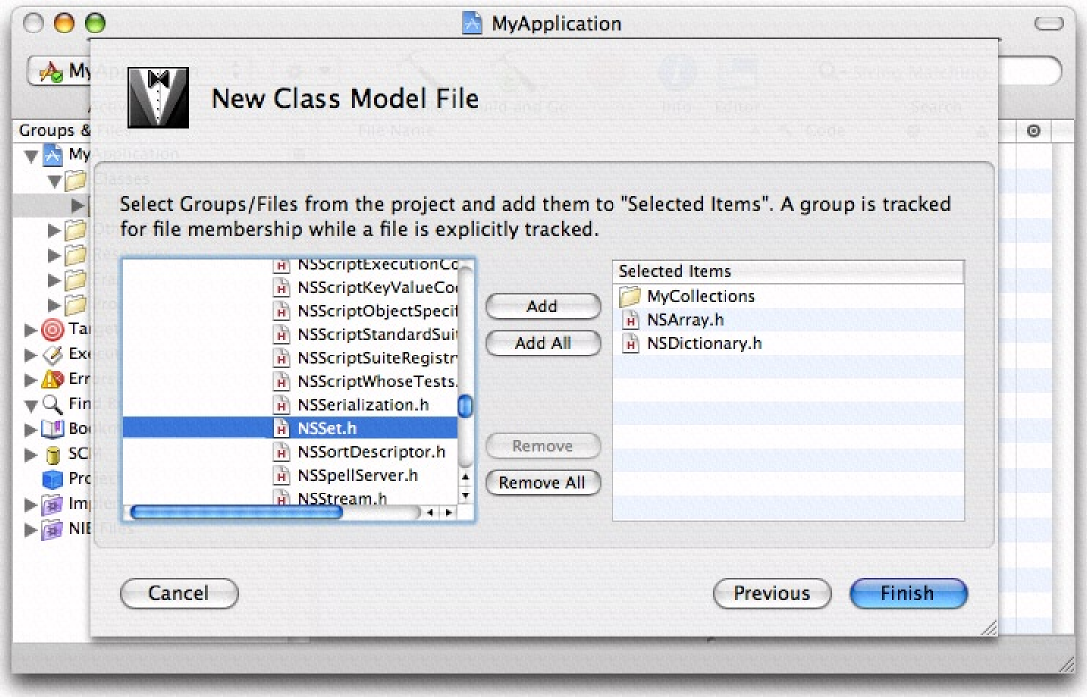
Creating Models

Xcode allows you to create models in two ways, as quick models and as project class model files (that is, model files you create with the New File Assistant). At first glance it may appear that the two sorts of class model are somehow different. It is important to realize that they are functionally the same but created in different ways and usually with a different immediate purpose in mind.

Creating a Class Model File

To create a class model file, choose File > New File and select Class Model from the Other group. You then name the file, and click Next. From the subsequent panel, shown in Figure 1, you select the files and containers that you want to contribute to the model.

Figure 1 Selecting groups and files to be in the model



Select Groups/Files from the project and add them to "Selected Items". A group is tracked for file membership while a file is explicitly tracked.

Selected Items
- MyCollections
- NSArray.h
- NSDictionary.h

When you click Finish, Xcode creates the model file, adds it to the project, and displays the class browser and diagram.

Creating a Quick Model

To create a Quick Model, select in the Groups & Files list the files and containers that you want to contribute to the model. Then choose Design > Class Model > Quick Model. Xcode displays the class browser and diagram.

A quick model is untitled and ephemeral. It does not appear in the project file browser, and if unchanged, it is closed without warning. If you make changes, however, you are prompted to save when you close the project. You can also save the model at any point using Save or Save As if you decide you want to keep the model.

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 Quick Model.

Models and Your Project

When you create a model, you add the source or the header files (or both), or containers (such as groups, targets, or projects, but not smart-groups, build phases, find results, and so on) that you want to contribute to the model. A model is dynamic, though. It is continuously updated in response to changes in your source code and the organization of your project. For this reason, you might add both a file and a group containing that file to a model, so that if the file is moved out of group it remains in the model (this strategy may be particularly useful early in a project's lifetime when groups are likely to change).

When you add a class to a model, its immediate superclass is implicitly added to the model (even if it's not in the project).

Models are considered integral to the project, so you should typically add new model files (that you create using the New File menu item) to the project. Quick models, on the other hand, are initially considered temporary, so you don't have to add them to the project immediately. If you decide to save a quick model, you can add it to the project later.

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

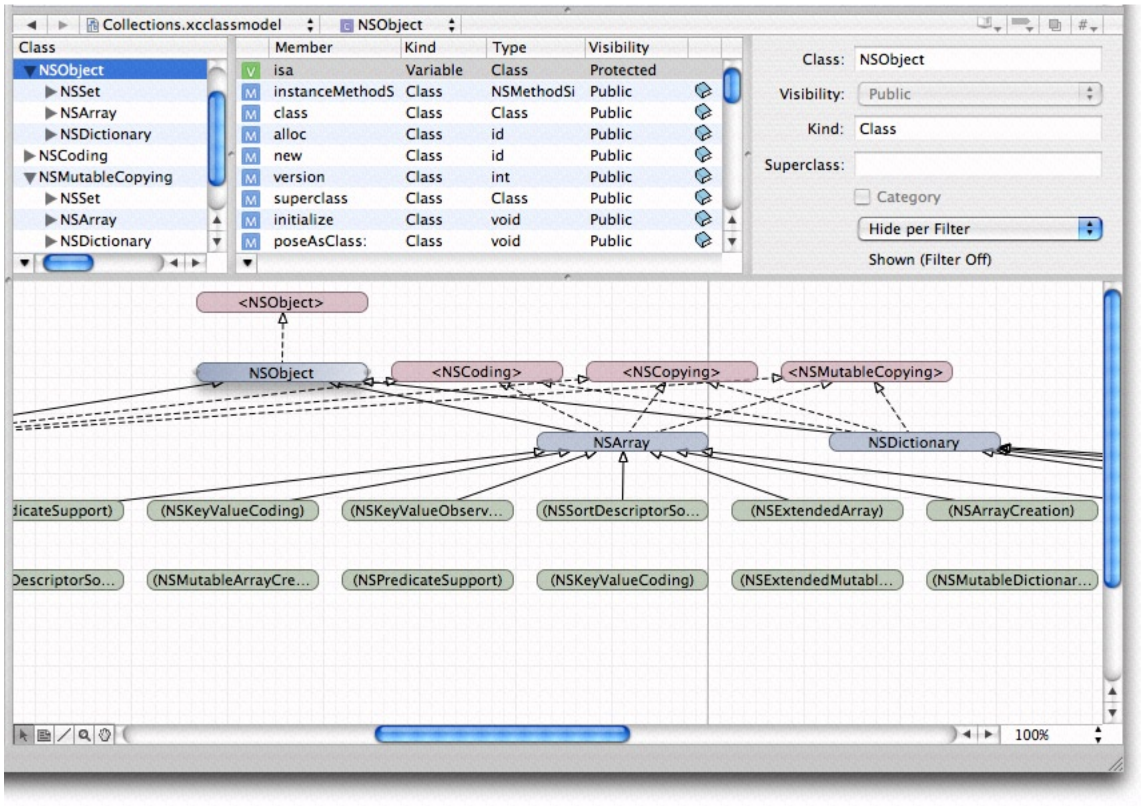
You might find it useful to create several different class models in your project. Each model may present subsets of the data in different ways. Each gives a its own perspective on the project, and so may be useful for different situations.

Because class model files depend on the project index, they can't survive outside the project.

Model Views

The model has two main views—the diagram and the browser. These views have different roles. The diagram view is typically best when you need a high-level overview of the project. The browser view gives you more detailed information. When you have large collections of classes, you can minimize the information shown in the diagram (for example, view just class names and inheritance 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. Figure 1 shows the class browser and a class hierarchy diagram.

Figure 1 Browser view and diagram view for a class model



Collections.xcclassmodel NSObject

Class
▼NSObject
▶NSSet
▶NSArray
▶NSDictionary
▶NSCoding
▼NSMutableCopying
▶NSSet
▶NSArray
▶NSDictionary

	Member	Kind	Type	Visibility
V	isa	Variable	Class	Protected
M	instanceMethodS	Class	NSMethodSi	Public
M	class	Class	Class	Public
M	alloc	Class	id	Public
M	new	Class	id	Public
M	version	Class	int	Public
M	superclass	Class	Class	Public
M	initialize	Class	void	Public
M	poseAsClass:	Class	void	Public

Class: NSObject
Visibility: Public
Kind: Class
Superclass:
Category
Hide per Filter
Shown (Filter Off)

You can adjust the relative sizes of the browser and diagram views—and hide the diagram view completely—by moving the split view divider that separates them. 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 the browser view by choosing Design > Hide Browser View (and show it again by choosing Design > Show Browser View).

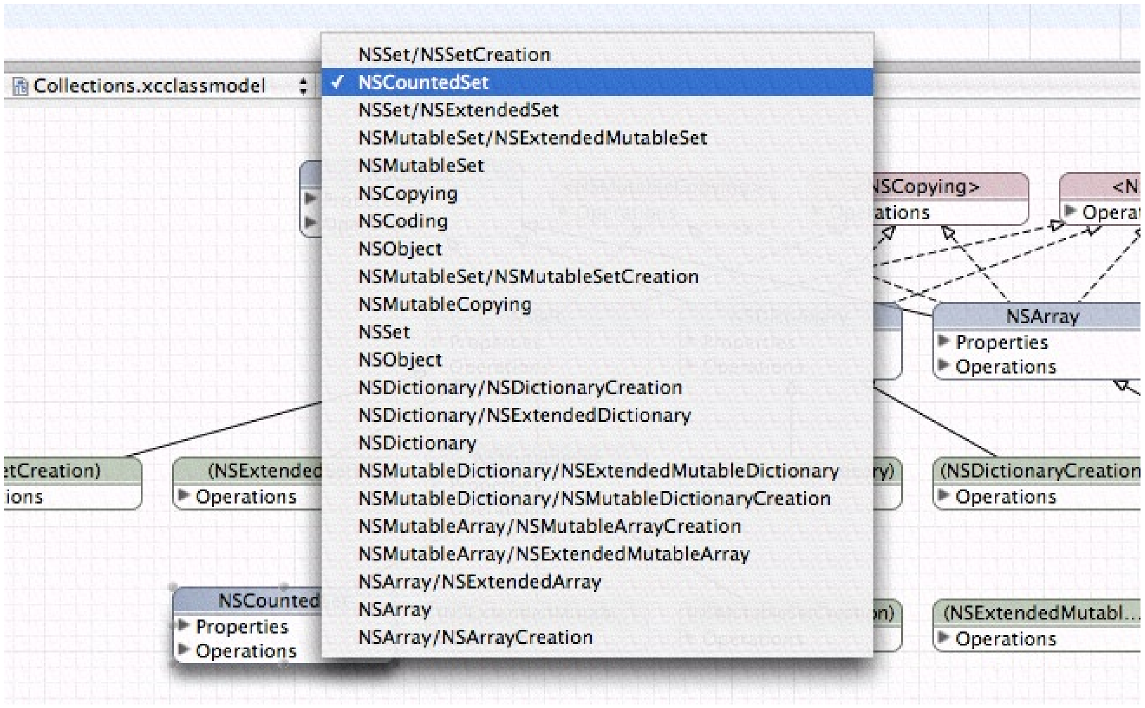
Navigation

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 view.

If you have a large class diagram, there are two strategies you can use to aid navigation. First, you can begin typing the name of the class you want. As you type characters, Xcode selects the alphabetically "topmost" class 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 in the diagram and brought into view. This feature may be particularly useful if the browser is hidden.

Figure 2 The Elements pop-up menu


Collections.xcclassmodel
NSSet/NSSetCreation
✓ NSCountedSet
NSSet/NSExtendedSet
NSMutableSet/NSExtendedMutableSet
NSMutableSet
NSCopying
NSCoding
NSObject
NSMutableSet/NSMutableSetCreation
NSMutableCopying
NSSet
NSObject
NSDictionary/NSDictionaryCreation
NSDictionary/NSExtendedDictionary
NSDictionary
NSMutableDictionary/NSExtendedMutableDictionary
NSMutableDictionary/NSMutableDictionaryCreation
NSMutableArray/NSMutableArrayCreation
NSMutableArray/NSExtendedMutableArray
NSArray/NSExtendedArray
NSArray
NSArray/NSArrayCreation
NSArray
Properties
Operations
(NSDictionaryCreation
(NSExtendedMutabl…
NSCounted
(NSExtended

The browser view, however, is useful when you have a large class diagram with all compartments rolled up and you want to see more details about a given class but don't want to make the diagram bigger. The browser also shows more information than is available in the diagram (parameters, return types, and so on).

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 or navigate to documentation. You can Control-click the diagram background to perform operations related to the visual representation, for example, you can hide grid lines, zoom, and set the alignment of drawing elements.

The Info Window

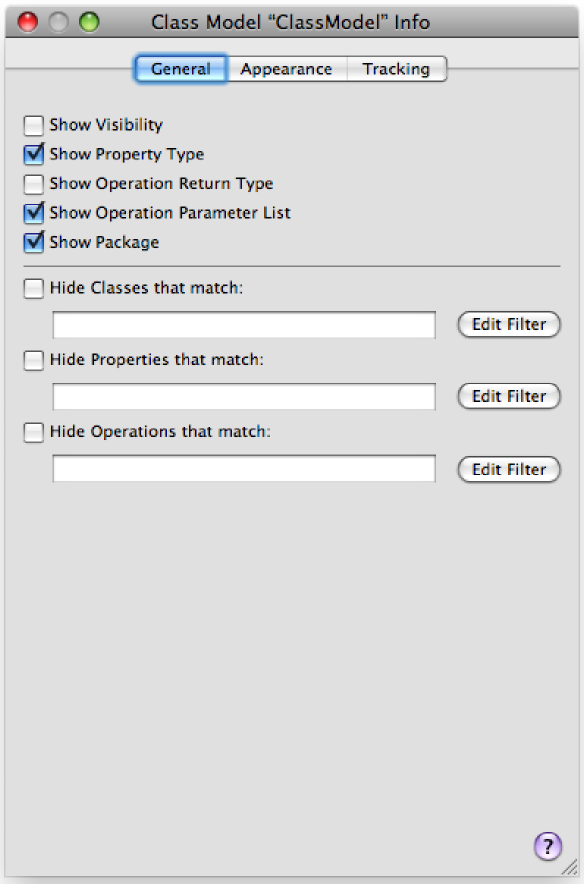
The Info Window (inspector) contains three panes, for General settings, Appearance, and Tracking.

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 General Pane

You use the general pane to. Figure 1 shows an Appearance pane with custom settings.

Figure 1 The General pane

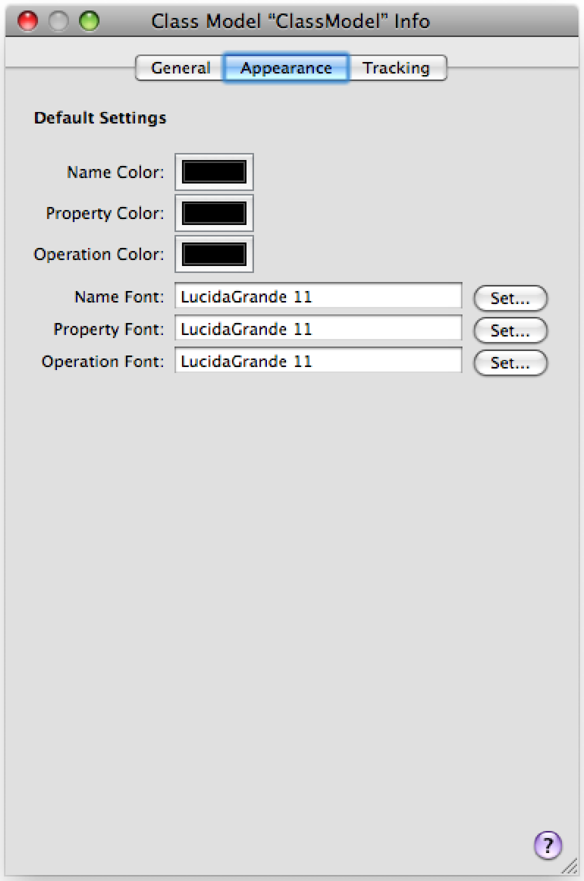


- [ ] Show Visibility
- [x] Show Property Type
- [ ] Show Operation Return Type
- [x] Show Operation Parameter List
- [x] Show Package
- [ ] Hide Classes that match:
- [ ] Hide Properties that match:
- [ ] Hide Operations that match:

The Appearance Pane

You use the Appearance pane to set default colors and fonts for element names and properties. Figure 2 shows an Appearance pane with custom settings.

Figure 2 The Appearance pane



Default Settings

Name Color:	■	
Property Color:	■	
Operation Color:	■	
Name Font:	LucidaGrande 11	Set...
Property Font:	LucidaGrande 11	Set...
Operation Font:	LucidaGrande 11	Set...

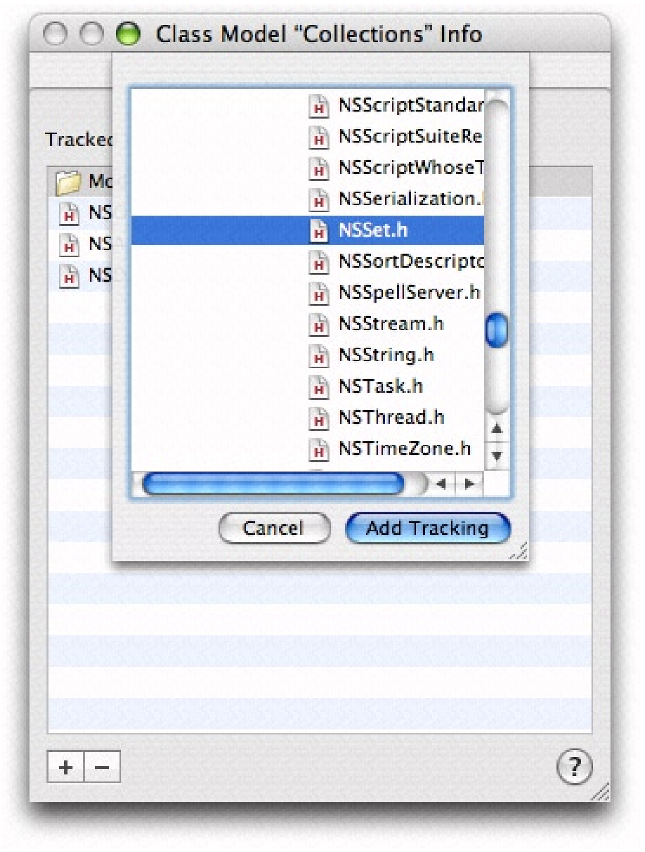
Tracking

The tool uses the project indexer to track changes to your project. The class models always represent the actual classes in the files and groups in your project. Xcode automatically updates them as you change your source code—even if you add, remove, or refactor classes. To function properly, therefore, the class model requires that the project indexer be enabled.

If the project indexing is not complete, the model pane simply displays the word "Indexing" until indexing is complete. If indexing is disabled or you open a project on a read-only partition, you see an appropriate warning.

You use the Tracking pane of the Info window (inspector) as shown in Figure 3) to change the list of tracked items that belong to the model. Click the plus (+) or minus (–) button (in the lower left of the pane) to add or remove files and groups.

Figure 3 Adding a file in the Tracking pane



As you add and remove files from any project groups that make up a model, corresponding classes appear in and disappear from the browser and diagram as appropriate.

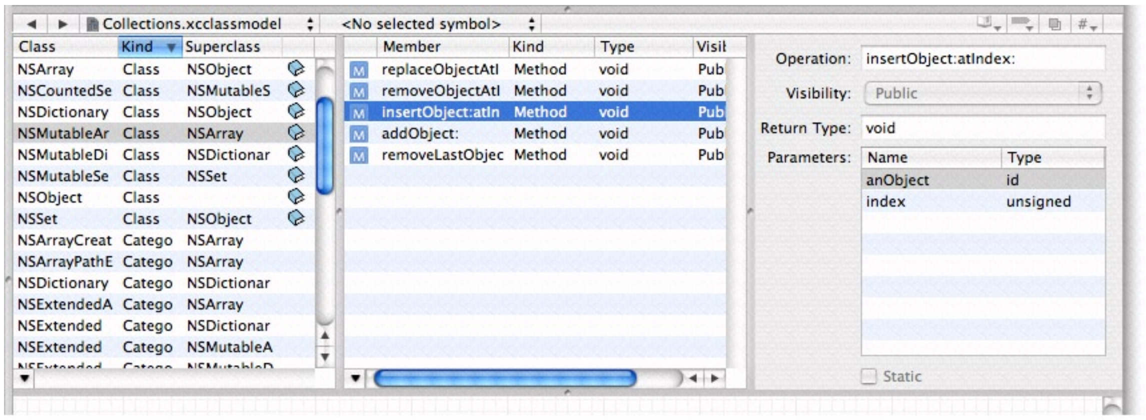
The Browser View

The browser view gives you a different perspective on the whole of your model (all classes are shown in the browser view even if they are hidden in the diagram view). Note that the design tools browser view is distinct from the Class Browser.

Overview

The browser view is by default always present in the model's editor view. You can hide it by choosing Design > Hide Browser View (and show it again by choosing Design > Show Browser View). The view has three separate parts: two table view panes—the class 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.

Figure 1 The Browser View



Class	Kind	Superclass
NSArray	Class	NSObject
NSCountedSe	Class	NSMutableS
NSDictionary	Class	NSObject
NSMutableAr	Class	NSArray
NSMutableDi	Class	NSDictionar
NSMutableSe	Class	NSSet
NSObject	Class	
NSSet	Class	NSObject
NSArrayCreat	Catego	NSArray
NSArrayPathE	Catego	NSArray
NSDictionary	Catego	NSDictionar
NSExtendedA	Catego	NSArray
NSExtended	Catego	NSDictionar
NSExtended	Catego	NSMutableA

Member	Kind	Type	Visil
replaceObjectAtI	Method	void	Pub
removeObjectAtI	Method	void	Pub
insertObject:atIn	Method	void	Pub
addObject:	Method	void	Pub
removeLastObjec	Method	void	Pub

Operation: insertObject:atIndex:
Visibility: Public
Return Type: void
Parameters:

Name	Type
anObject	id
index	unsigned

Static

Table View Panes

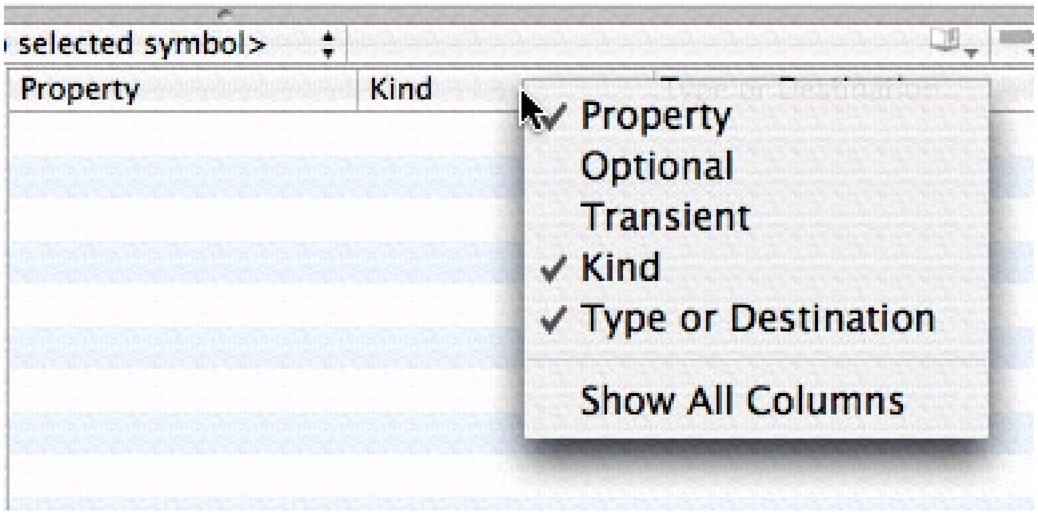
The table view in the leftmost pane lists the classes, categories, and protocols in the model. The columns in the class list show the element name, the element type (class, category, or protocol), the hidden status, and a link to documentation if there is any associated with the element.

The table view in the properties pane shows summary information about the properties and methods associated with the current selection in the classes table, including the name, type, and visibility, and a link to documentation if there is any associated with the element. If you make a multiple selection in the class 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.

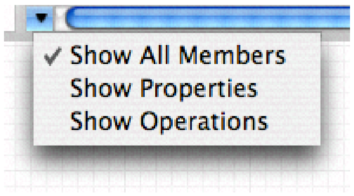
You can choose which columns to see by Control clicking table header cells to display a pop-up list that you can use to toggle the display of columns (see Figure 2). If you have a multiple selection, Show All Columns means that only the set of columns common to all members of the selection may be displayed,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

Figure 2 Browser column options



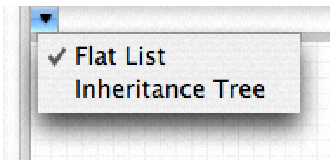
You can also choose which properties are shown by viewing the command pop-down menu in the property table. Each command, shown in Figure 3, acts as a toggle to change the visibility of properties and operations in the table view.

Figure 3 Property list view options



Finally, you can display the class 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 Class view options



Detail Pane

The detail pane shows information about the most recently selected element in the classes or properties table. You use the detail panel to set the hidden status of individual class elements. This setting determines how much is displayed in the diagram view, as described in "Hiding" (page 33). If you make a multiple selection, the editor shows the best representation it can of the union of the selected items.

The Diagram View

The diagram contains two important shapes, rounded rectangular nodes and lines. It may also contain annotations. Although you can change the layout and visual appearance of the nodes and lines (and optionally hide classes, properties and so forth), you can modify classes, the relationships between them, their properties, and so forth only by editing source files. For editing annotations, see "Annotations" (page 27).

Diagram Elements

The diagram view contains two main elements, rounded rectangles—which represent nodes—and lines. Diagrams may also contain annotations.

Nodes

Nodes can be classes, categories, or protocols (interfaces). The name is given in the title bar, and for C++/Java, includes the namespace or package. Nodes are color coded to help you readily identify different types; different text styles help to further differentiate element and method types. Compartments within a node represent features of the class—properties for instance variables, and operations for methods.

A node may be split into three sections: The title bar containing the name of the element (including the namespace or package for C++/Java), and two compartments. The compartments show properties (instance variables) and operations (methods or member functions)—see Figure 2 (page 27).

You can use the nodes for navigation. To go to your source files (or to the header file for system files), you can click the (>) symbol in a node's title bar; you can also choose Design -> Class Model or use the contextual menu to navigate to any declaration, definition, or documentation that is available.

Text and Color Coding

The names of classes, categories, and protocols are represented differently: Class names appear unadorned, category names are surrounded by parentheses, and protocol names are surrounded by angle brackets. If an operation name is underlined, it is a class method.

By default, classes are represented in blue, categories in gray, and protocols (interfaces) in red. Project and framework classes are further differentiated by the saturation of the color (externals are dimmer). You can change both the default colors and the colors for individual nodes (see "Colors and Fonts" (page 30)).

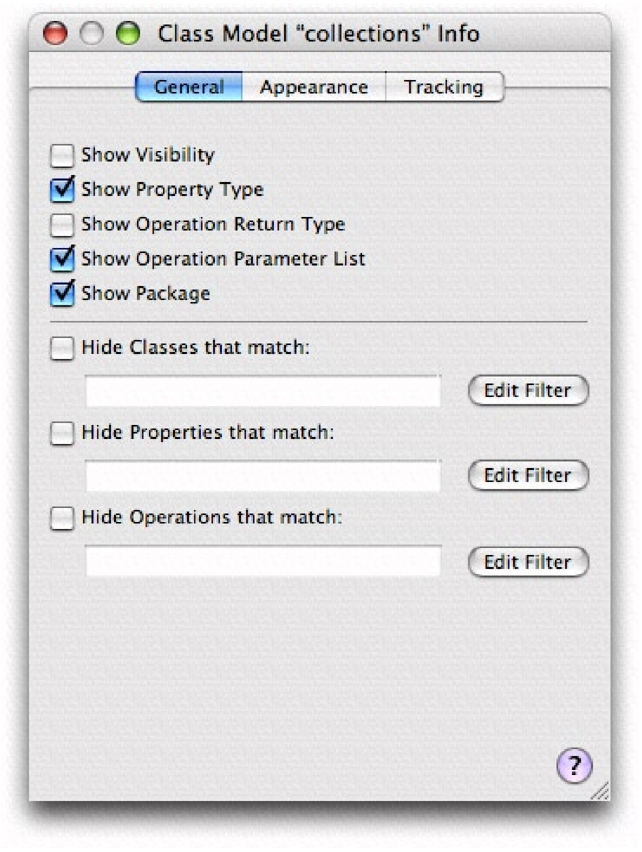
Compartments and View Options

Compartments within a node represent features of the class. The Properties compartment lists instance variables; the Operations compartment lists methods—class methods are underlined.

Within a node, you can display additional information. Using the General pane of the Info window (shown in Figure 1), you can choose whether or not to show:

- Visibility flag (public, private, and protected may be indicated by an icon in the compartment)
- Property types (for each property, shown after a colon in the compartment)
- Operation return types and parameter types
- Package information

Figure 1 Info window for a class model diagram


Class Model "collections" Info

- [ ] Show Visibility
- [✓] Show Property Type
- [ ] Show Operation Return Type
- [✓] Show Operation Parameter List
- [✓] Show Package
- [ ] Hide Classes that match:
- [ ] Hide Properties that match:
- [ ] Hide Operations that match:

You have further control over what is displayed in the diagram—for an explanation, see "Filtering and Hiding" (page 31).

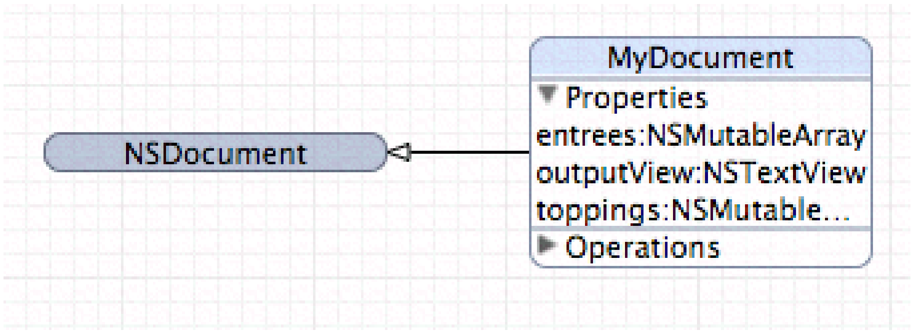
Roll-Up and Expansion

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

- Rolled up, so that just the name of the class is showing. This gives the most compact representation, with maximum information density in the diagram. (In Figure 2 the NSDocument node is rolled up.)
- Compartment titles showing. The titles are Properties and Operations. 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 2 the properties compartment of MyDocument is expanded, but the operations compartment is not.)

Figure 2 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).

Lines

Lines indicate different things depending on whether they are solid or dashed, what sort of arrowhead is present, and what objects they connect.

A solid line with an open arrowhead:

- Denotes inheritance when it connects classes
- Specifies the class of which the category is a category when it connects a class and a category

A dashed line with an open arrowhead denotes implementation of a protocol (in Java, an interface).

You can edit only lines that go to or from annotations—other lines are created automatically based on the contents of your project.

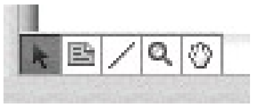
Annotations

You can add annotations to the diagram to provide explanatory text using the text tool. You have full access to text styling options from the Format menu. You can connect a comment to a class using the line tool.

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 3.

Figure 3 Diagram tools



- **Arrow**. You use the arrow tool to make selections and to move and resize graphic elements.
- **Text**. You use the text tool to annotate diagrams in a class model. To edit text in the text area, simply double-click inside the element. You can use the Formatting menu, Font panel, Colors panel, and so on to format the text as you wish. You can use the line tool to connect a text area to a specific class.
- **Line**. You use the line tool to connect a comment to a specific class. To connect two elements, select the line tool, then drag from one end of the connection to the element at the other end.
- **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 28) 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.

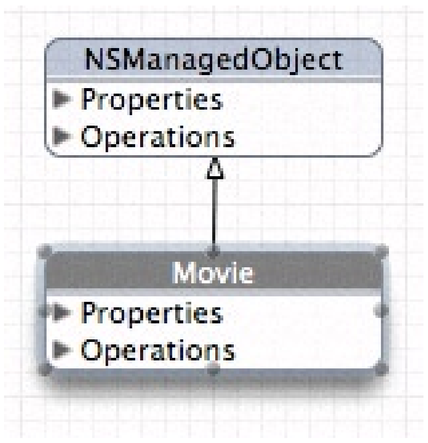
Layout

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" (page 29)) in the same way.

  When you select a shape, "handles" appear around its edges (as shown in Figure 4). You can drag the handles to resize the shape.

  Figure 4 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.

- **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.

- **Automatic layout**. The class modeler provides two ways to automatically lay out elements in a diagram, force-directed layout and hierarchical layout. To access them, choose Design > Automatic Layout.

  With hierarchical layout, parent elements are typically at the top of the diagram, children at the bottom. This gives a generally horizontal layout, and is typically better for deep hierarchies.

  The force-directed layout tends to produce circular arrangements, with commonly referenced classes in the middle. This is usually better for shallow hierarchies. Note that the force-directed layout can take unbounded time to calculate—and so is not recommended for very large collections.

  You can apply the automatic layout feature just to selected items or to the whole diagram. The layout respects the current size of elements in the selection. If you expand or contract a node and then perform automatic layout again, the layout differs from the one that existed before the change in size.

- **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.

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 classes in the browser—the browser selection and diagram selection are kept synchronized.

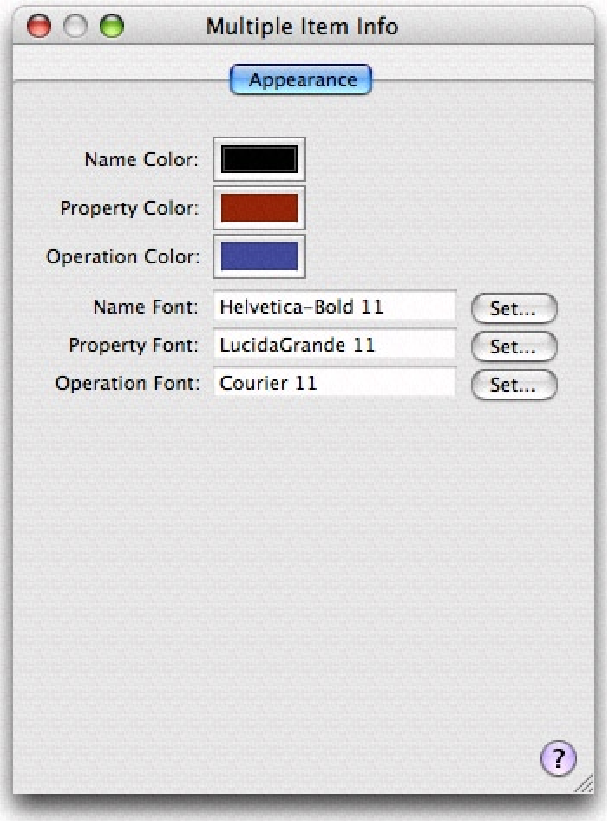
Choose Edit > Select All to select all elements in the diagram. Note that for items in the Diagram menu, clicking on the background (rather than on a drawn element) is the equivalent of selecting all but may be faster.

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. Classes, categories, and protocols each have distinct colors; moreover, the color of project resources is lighter than that of imported resources.

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



Name Color: ⬛
Property Color: 🟥
Operation Color: 🟦
Name Font: Helvetica-Bold 11
Property Font: LucidaGrande 11
Operation Font: Courier 11

You can also change the default settings for the entire model using the Appearance pane—see "The Appearance Pane" (page 18).

Filtering and Hiding

Sometimes diagrams contain more information than you want to see, and it can be useful to reduce clutter. Removing irrelevant classes makes it easier to concentrate on important ones—for example, you might remove `NSObject` from a class diagram to make it easier to see other relationships; or it might be that a tracked file contains definitions of several classes and you are interested in only one of them. You might also want to hide other details, such as private variables, or instance variables whose name starts with an underscore.

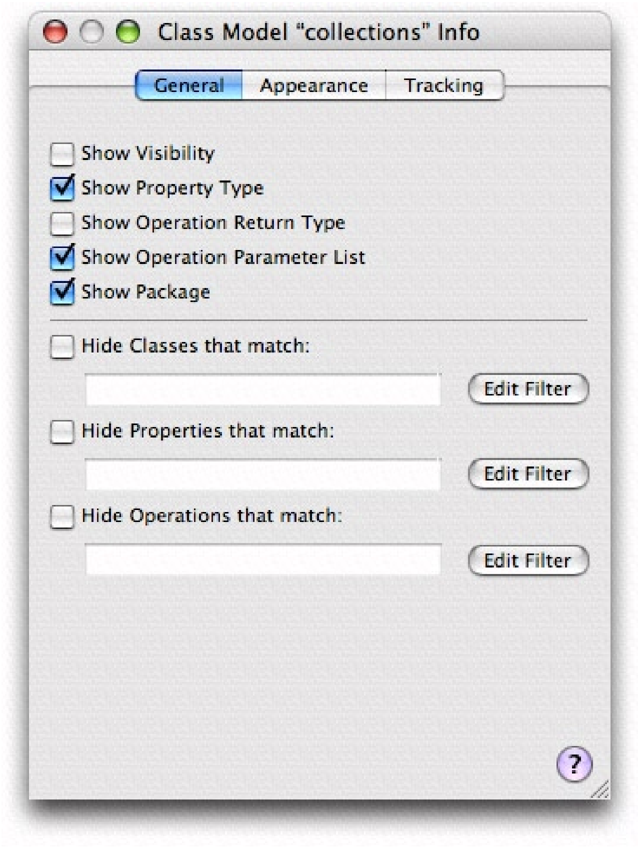
You can use a predicate to filter what classes and methods are shown in the diagram. On a class level, you can choose to use or override the filter. You can show a class if a filter would normally hide it, and vice versa.

Filtering and hiding settings affect only the diagram. The browser view still lists all the contents (because you need a way to be able to select a class if it's hidden!). Filtering and hiding are different from tracking. Tracking determines whether or not files contribute to the model at all.

Filtering

Filters apply as changes are made to files that contribute to the model. If you choose, for example, to hide all classes whose names begin with "XYZ", then in your header file you rename the class "XYZWidget" to "WXYWidget", and a node for "WXYWidget" appears in your diagram. To set up filtering, you use the General pane of the Info window, shown in Figure 6.

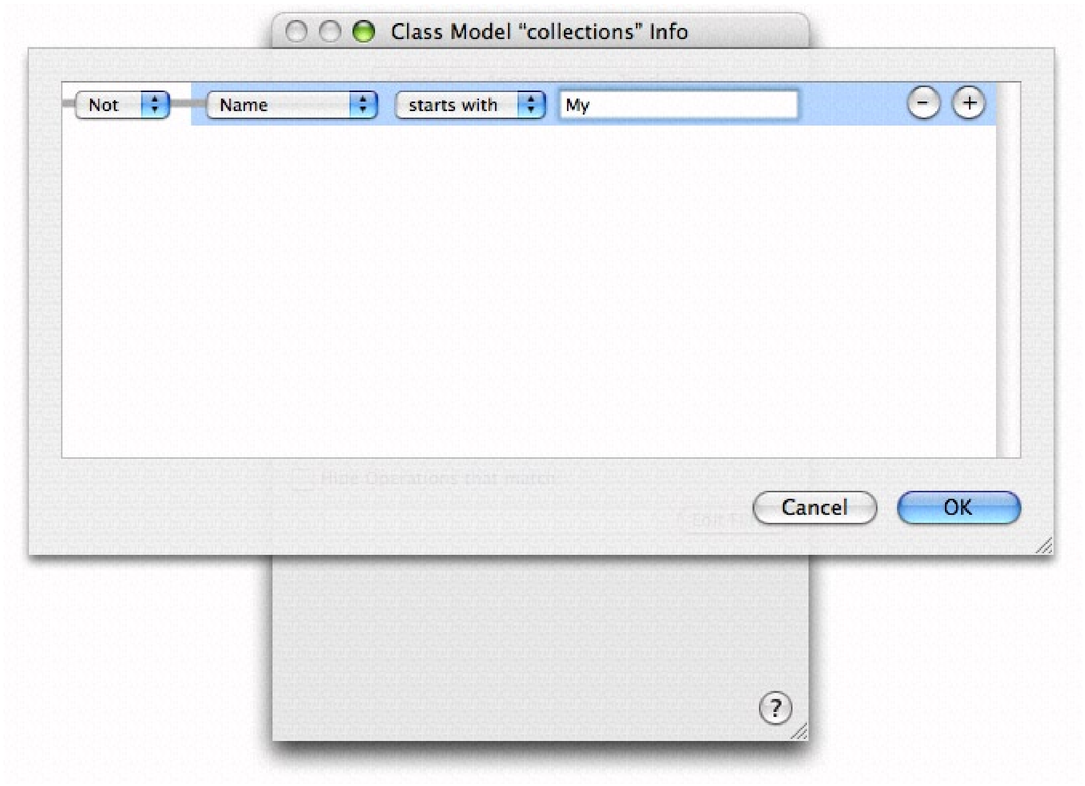
Figure 6 General pane of the Info window



- [ ] Show Visibility
- [✓] Show Property Type
- [ ] Show Operation Return Type
- [✓] Show Operation Parameter List
- [✓] Show Package
- [ ] Hide Classes that match:
- [ ] Hide Properties that match:
- [ ] Hide Operations that match:

You can set up independent filters for classes, properties, and operations, based on options such as their name. You can also toggle filters on and off as required. If you click Edit Filter, a sheet in which you can edit the filter appears, as shown in Figure 7. You can either enter a predicate directly into the appropriate text field or use the predicate builder.

Figure 7 The filter editor



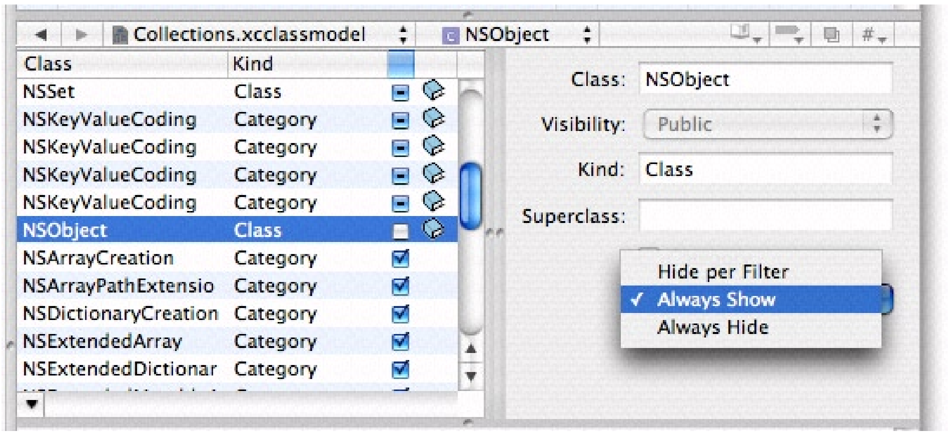
Note that string matches for all the filters are case-sensitive. For example, for a property type, "int" will match all integers, but "Int" will not. For some filter options, there are only a limited number of strings that are appropriate. For the class filter, such options are kind, where the strings are "Category", "Class", "Interface", and "Protocol", and language, where the strings are "C++", "Java", and "Objective-C". For the properties and operations filters, the only such option is visibility, where the strings are "Public", "Protected", "Package", and "Private".

Note: Filters are labeled to hide classes, properties, or operations. Sometimes you want to show only a subset. You can apply a `NOT` expression to a predicate to invert its meaning. Therefore if you want to show only those classes whose names begin with "XYZ", you hide those whose names do not begin with "XYZ".

Hiding

Hiding allows you to override the filtered state of a class (or protocol or category). You can specify that an element follow the filtered setting or that it be either always shown or always hidden. You can set the hiding state using the browser view, in either the Hidden column in the property pane or using the pop-up menu in the detail pane, as shown in Figure 8.

Figure 8 Setting hiding in the detail pane



Class	Kind	
NSSet	Class	▣
NSKeyValueCoding	Category	▣
NSKeyValueCoding	Category	▣
NSKeyValueCoding	Category	▣
NSKeyValueCoding	Category	▣
NSObject	Class	☐
NSArrayCreation	Category	✓
NSArrayPathExtensio	Category	✓
NSDictionaryCreation	Category	✓
NSExtendedArray	Category	✓
NSExtendedDictionar	Category	✓

Class: NSObject
Visibility: Public
Kind: Class
Superclass:

In the detail pane, you choose the setting from the pop-up menu. The browser has a three-state checkbox you can use to modify the hiding setting. The states correspond to the same states defined in the pop-up menu.

Note: If you find you are making frequent changes to the hiding settings, you may consider creating different models to provide different perspectives. Models are lightweight, and so do not consume significant system resources. Because they use tracking, they are always up to date.

Document Revision History

This table describes the changes to *Xcode Design Tools for Class Modeling*.

Date	Notes
2009-02-04	Made minor content fixes.
2008-04-15	New document that describes the Xcode Design Tools for Class Modeling.