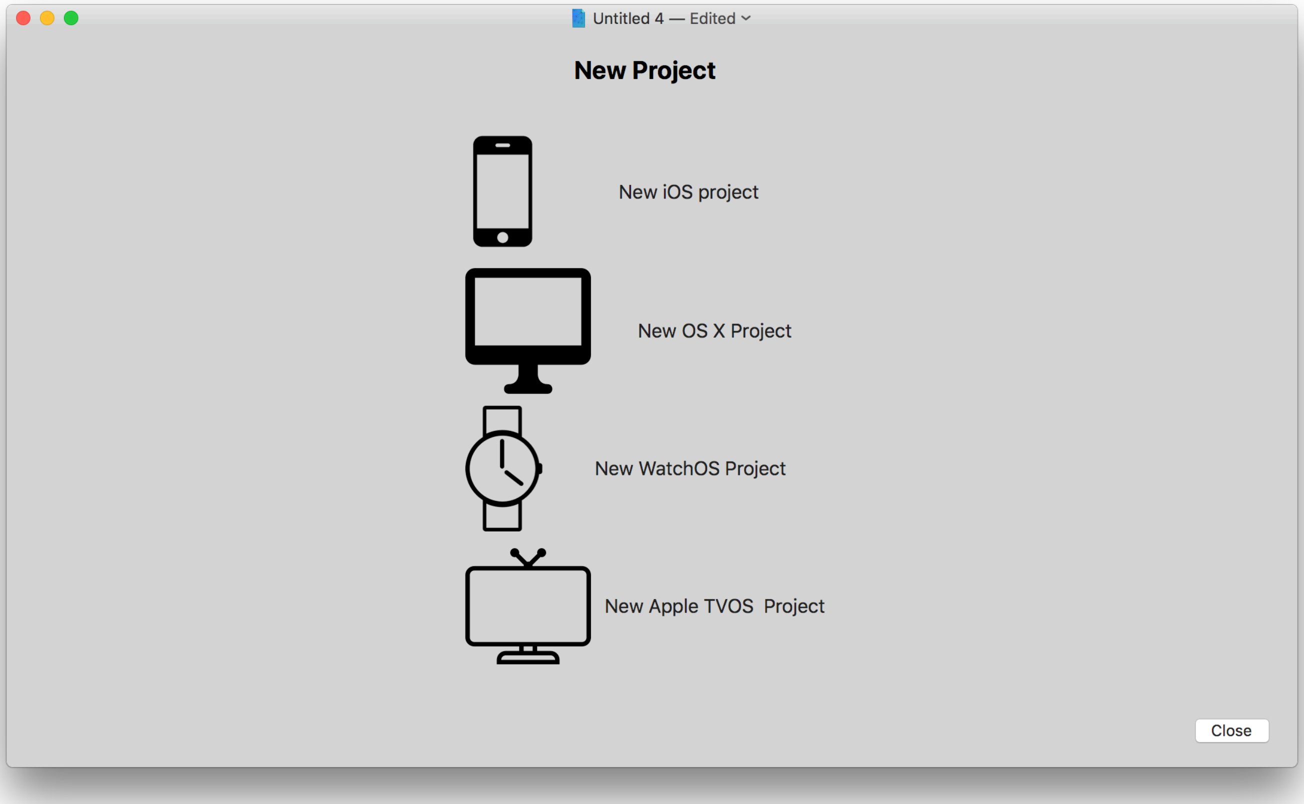
Devsketch

What is DevSketch

DevSketch is a tool for Swift and Objective-C developers. Its good for prototyping components and analysing existing code bases.

Getting started

Create a new document. You can choose from 4 preset sets of SDK's or just carry on by clicking the close button.



Setting up the workspace

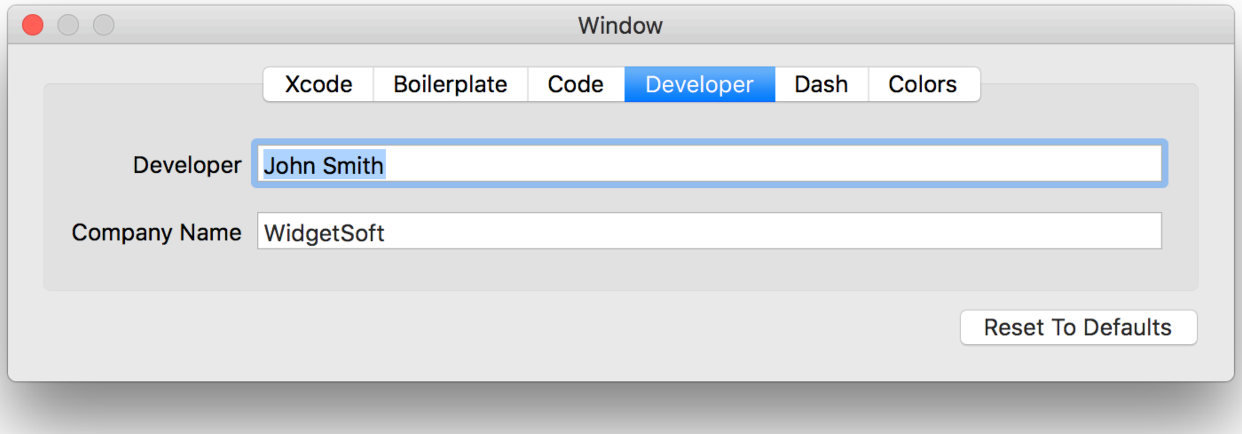
You can configure the workspace in a couple of places.

Application level preferences

Select **DevSketch > Preferences…**

This window allows you to pick which Xcode you are using.

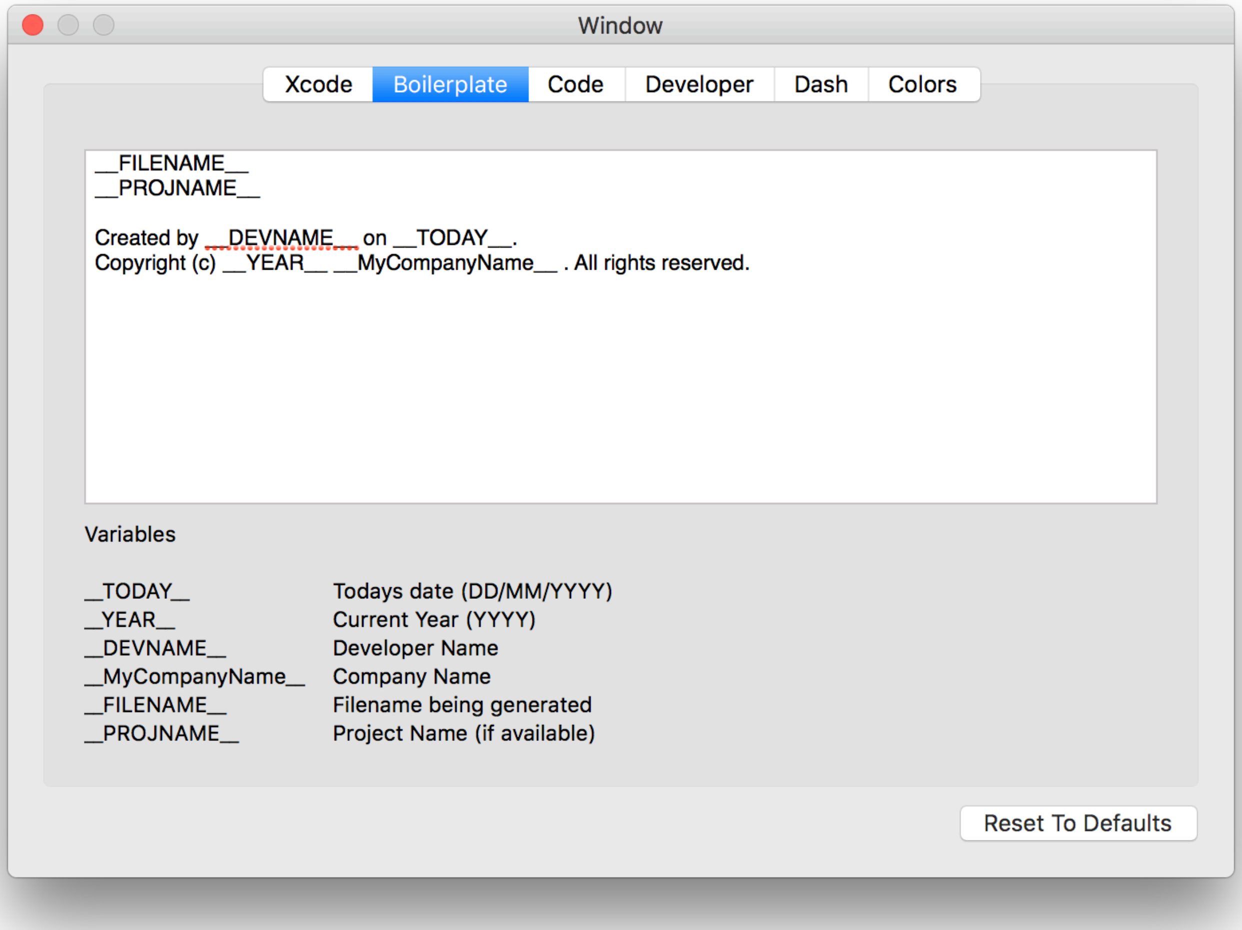
You can tell DevSketch who you are:



Developer: John Smith

Company Name: WidgetSoft

How you'd like your headers to be generated:



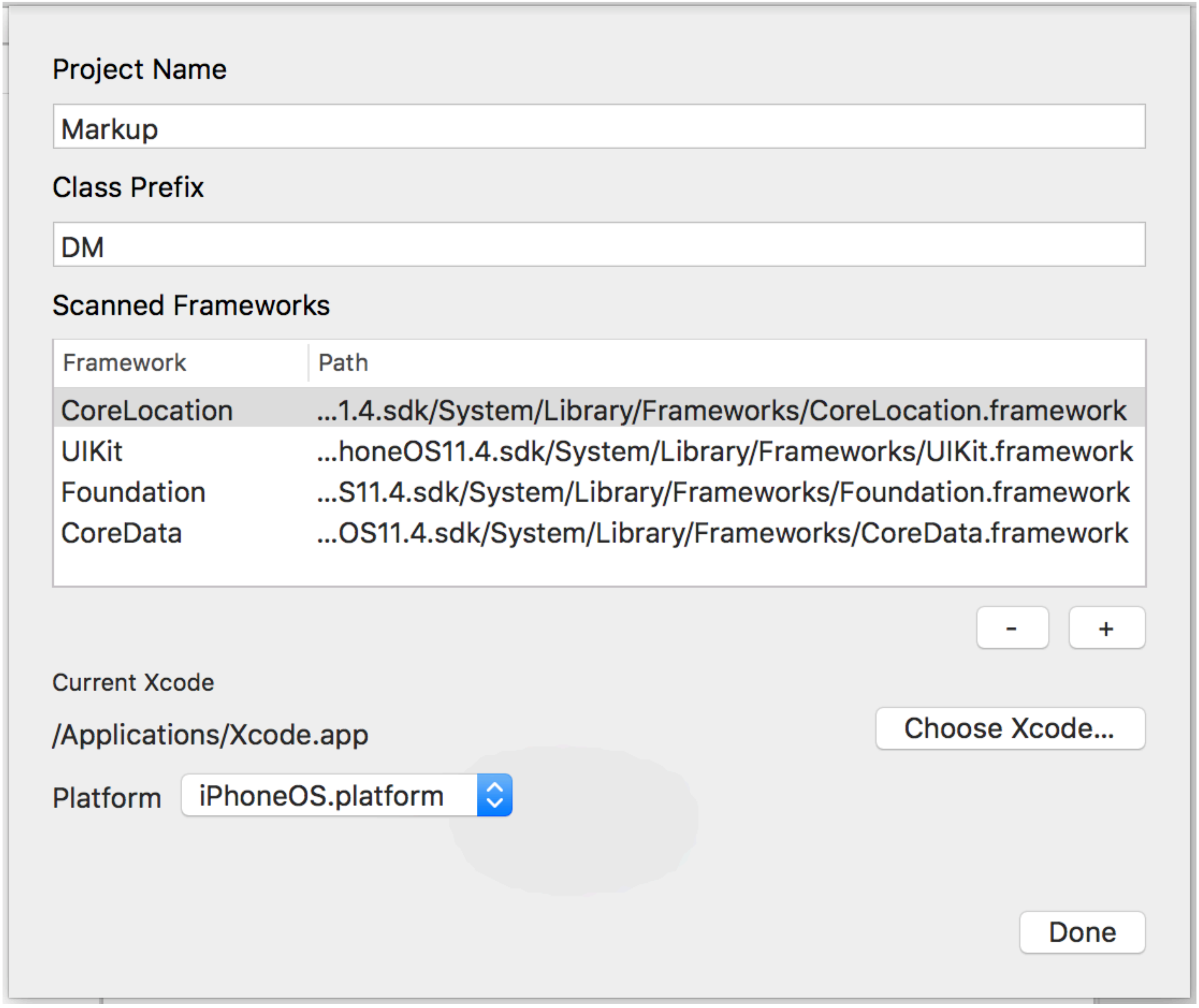
```
__FILENAME__
__PROJNAME__

Created by __DEVNAME__ on __TODAY__.
Copyright (c) __YEAR__ __MyCompanyName__ . All rights reserved.
```

Variables

__TODAY__	Todays date (DD/MM/YYYY)
__YEAR__	Current Year (YYYY)
__DEVNAME__	Developer Name
__MyCompanyName__	Company Name
__FILENAME__	Filename being generated
__PROJNAME__	Project Name (if available)

Document Level Preferences

Select **Document > Edit Document Settings…**



Project Name

Markup

Class Prefix

DM

Scanned Frameworks

Framework	Path
CoreLocation	...1.4.sdk/System/Library/Frameworks/CoreLocation.framework
UIKit	...honeOS11.4.sdk/System/Library/Frameworks/UIKit.framework
Foundation	...S11.4.sdk/System/Library/Frameworks/Foundation.framework
CoreData	...OS11.4.sdk/System/Library/Frameworks/CoreData.framework

Current Xcode

/Applications/Xcode.app

Platform iPhoneOS.platform

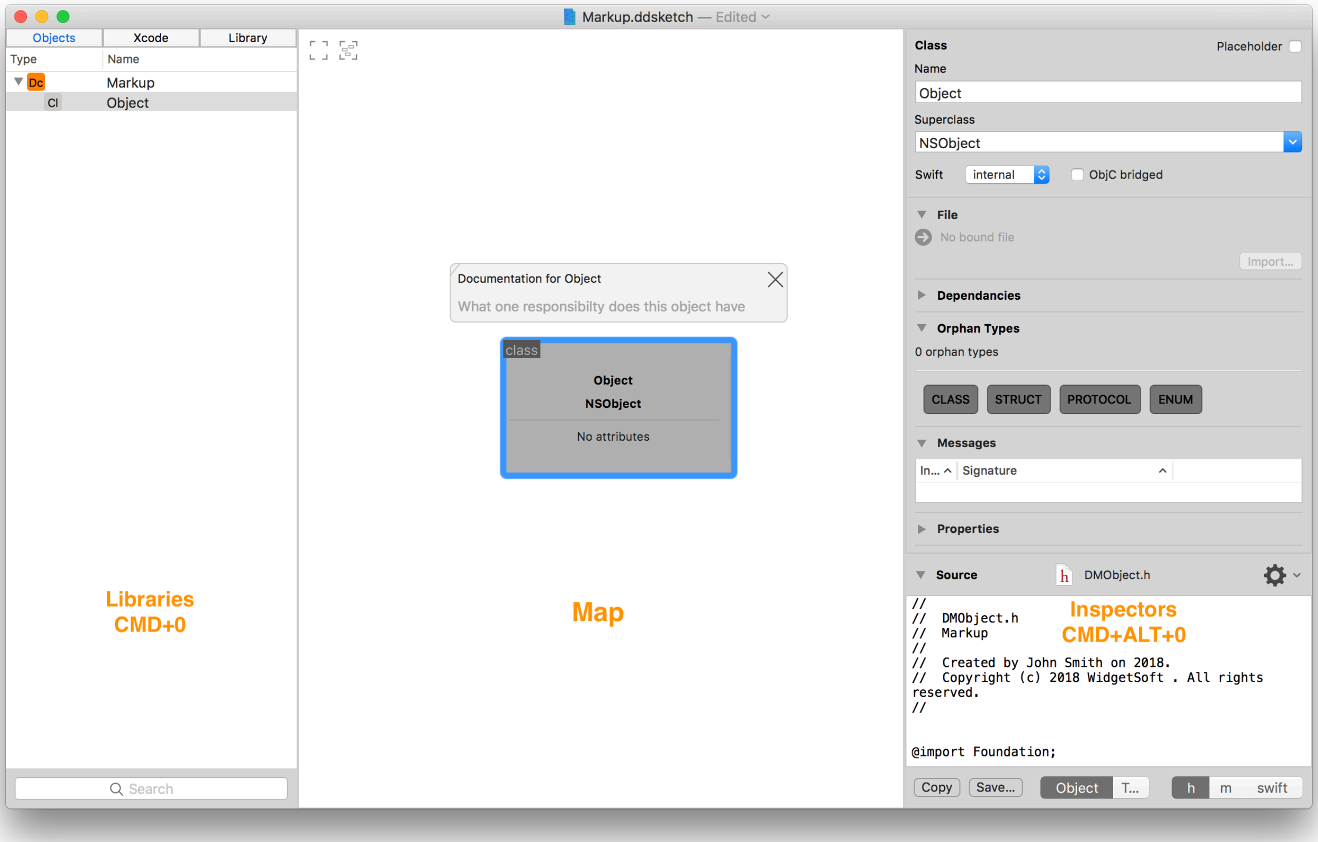
Here you can add or remove SDKs and give the project a name.

If you are generating Objective-C code you can add a prefix to all type names.

Working with the workspace.

Main window

The document window has 3 sections.



Class

Placeholder ☐

Name: Object

Superclass: NSObject

Swift: internal ☐ ObjC bridged

No bound file

0 orphan types

Documentation for Object

What one responsibilty does this object have

Object
NSObject
No attributes

```
//
//  DMObject.h
//  Markup
//
//  Created by John Smith on 2018.
//  Copyright (c) 2018 WidgetSoft . All rights
reserved.
//


@import Foundation;
```

Libraries on the left.

This is where you find objects you have created or objects from the SDK's that your project contains.

Visual Map in the center.

This is visual map of your objects and its relationships to other objects. An object can exist in the library but not be visible on the map.

Inspectors on the right.

The inspectors tell you about the selected object (or objects).

You can collapse or expand the left and right sections by:

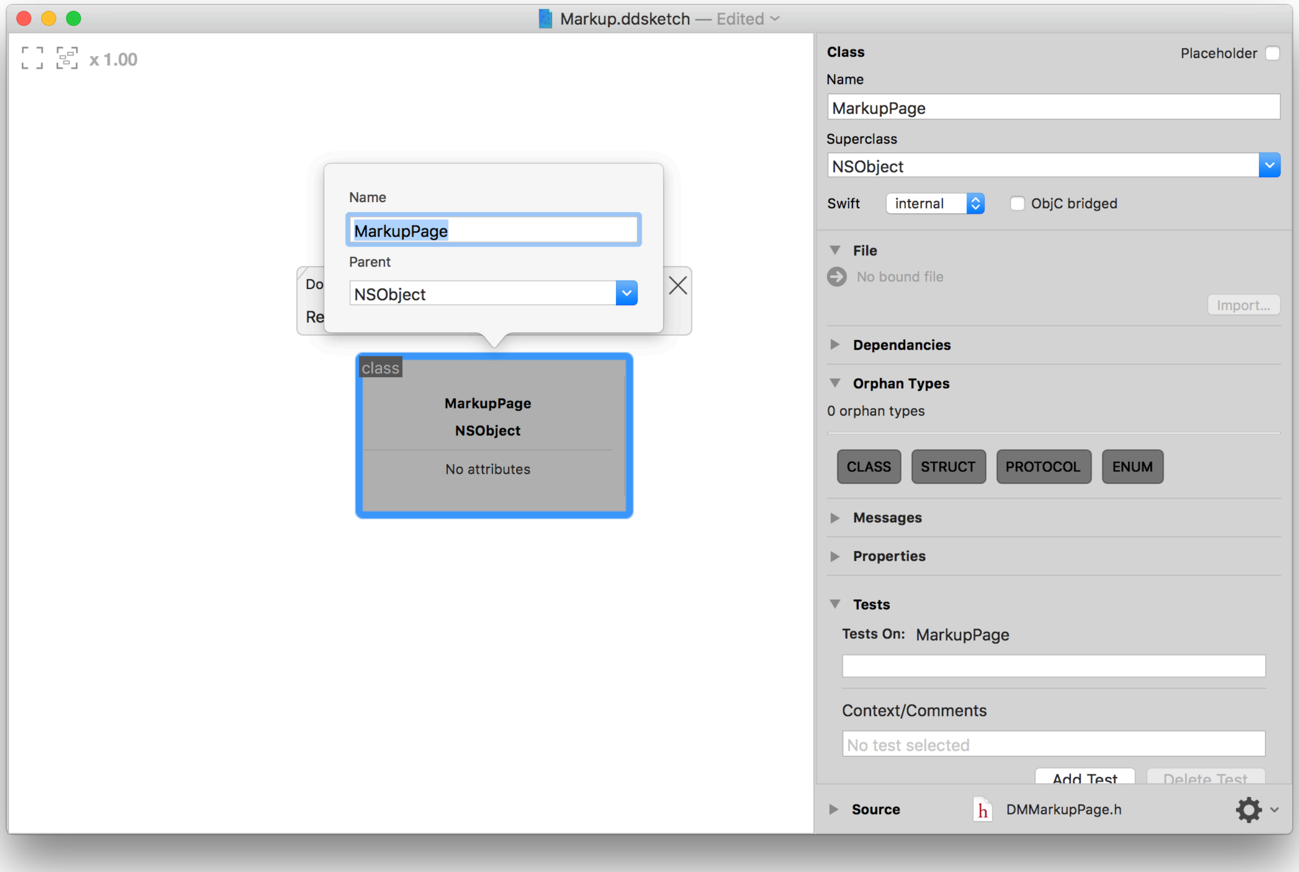
- dragging the vertical split.
- using the key shortcuts CMD+0 for the left, CMD+ALT+0 for the right.

Visual Map

This contains a representation of the objects in your workspace.

A new document contains a stub class representation.

Editing



Markup.ddsketch — Edited

x 1.00

Name

MarkupPage

Parent

NSObject

class

MarkupPage

NSObject

No attributes

Class

Placeholder □

Name

MarkupPage

Superclass

NSObject

Swift internal □ ObjC bridged

File

No bound file

Orphan Types

0 orphan types

Tests

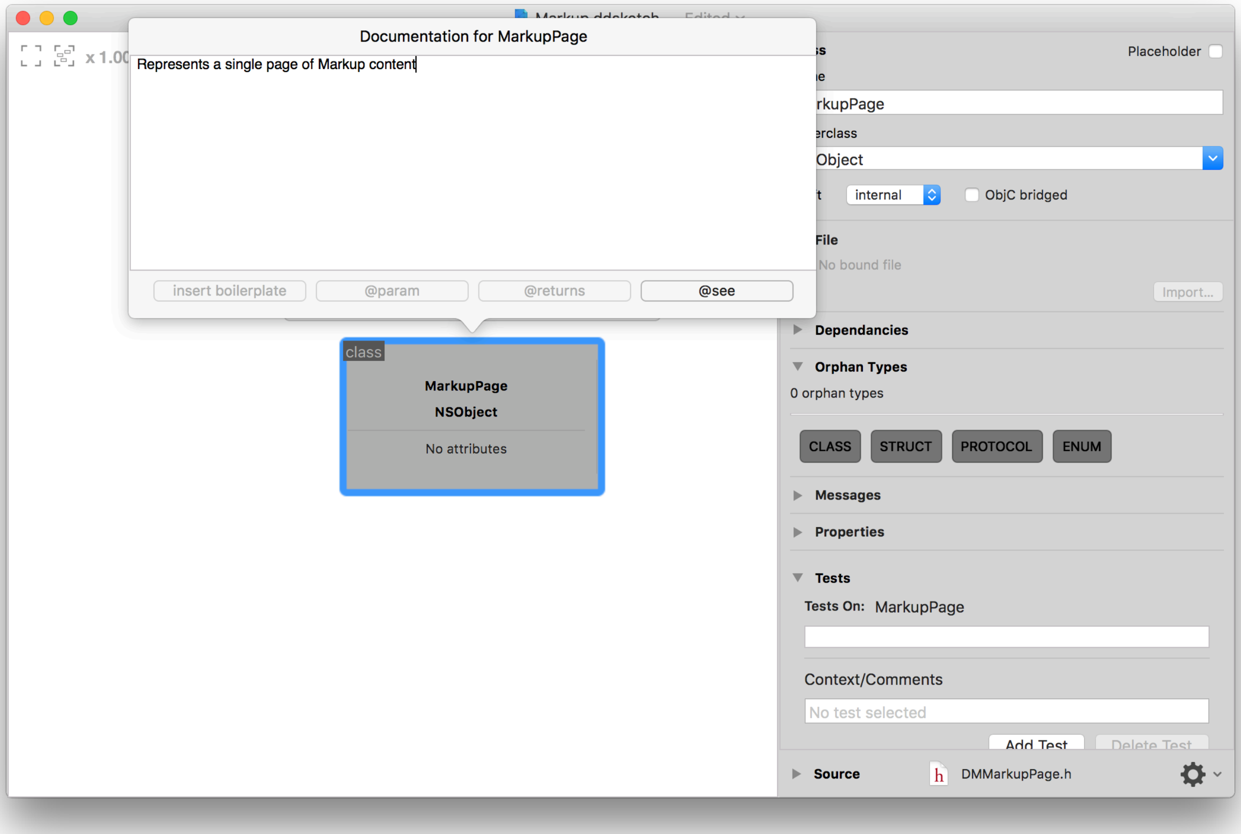
Tests On: MarkupPage

Context/Comments

No test selected

Source DMMarkupPage.h

- Double click the object to edit the name and parent.



Documentation for MarkupPage

Represents a single page of Markup content

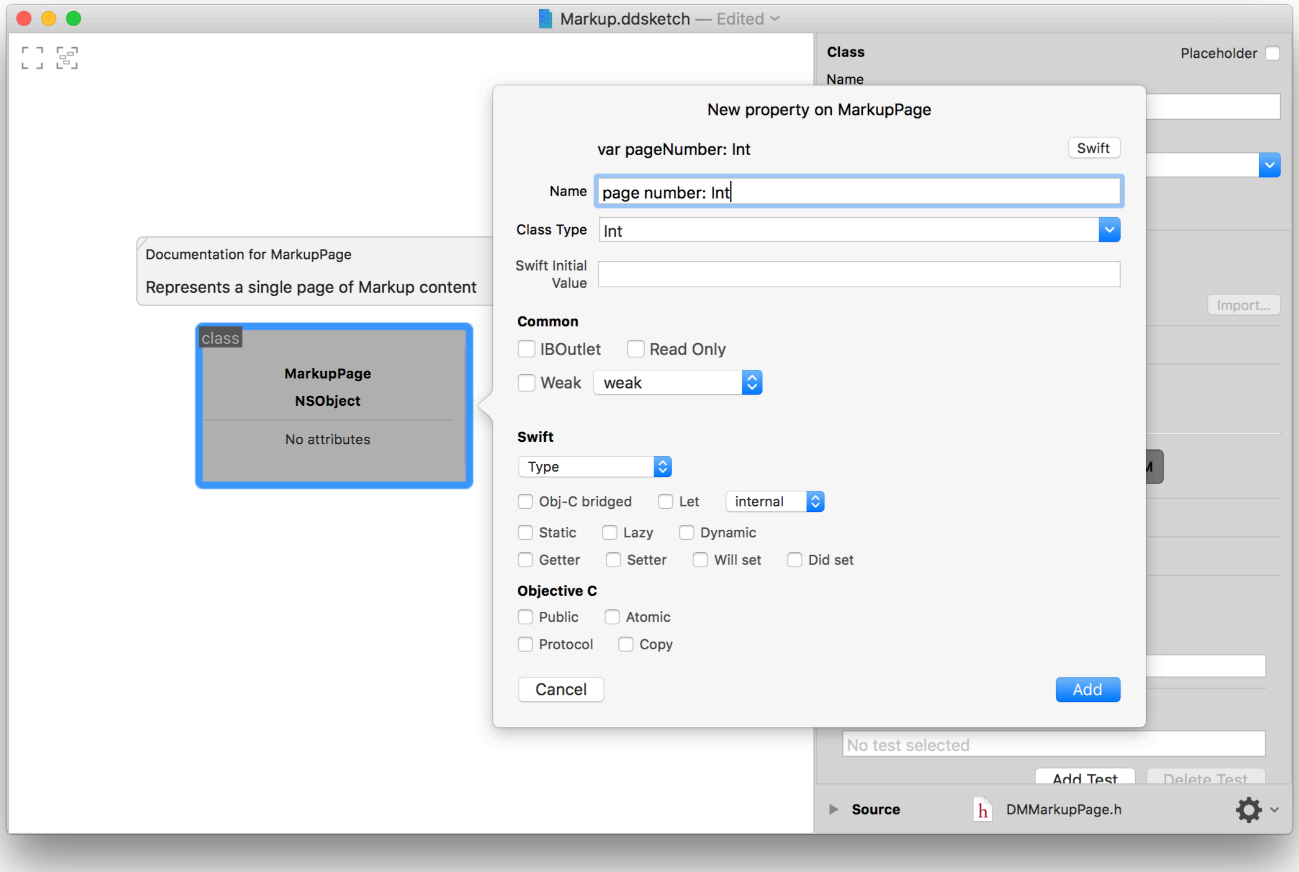
- Double click the documentation panel (ALT+Space) to edit the description.

Deleting

- Delete the object from the map with the Delete key. (**Edit > Delete from view**)The object will still be in the database.
- Delete the object from the database with ALT+Delete (ALT + **Edit > Delete from database**)

Adding a property

Select the object and press CMD+L (**Object > Add Property**)



New property on MarkupPage

var pageNumber: Int

Name: page number: Int

Class Type: Int

Swift Initial Value:

Common

□ IBOutlet □ Read Only

□ Weak weak

Swift

Type

□ Obj-C bridged □ Let internal

□ Static □ Lazy □ Dynamic

□ Getter □ Setter □ Will set □ Did set

Objective C

□ Public □ Atomic

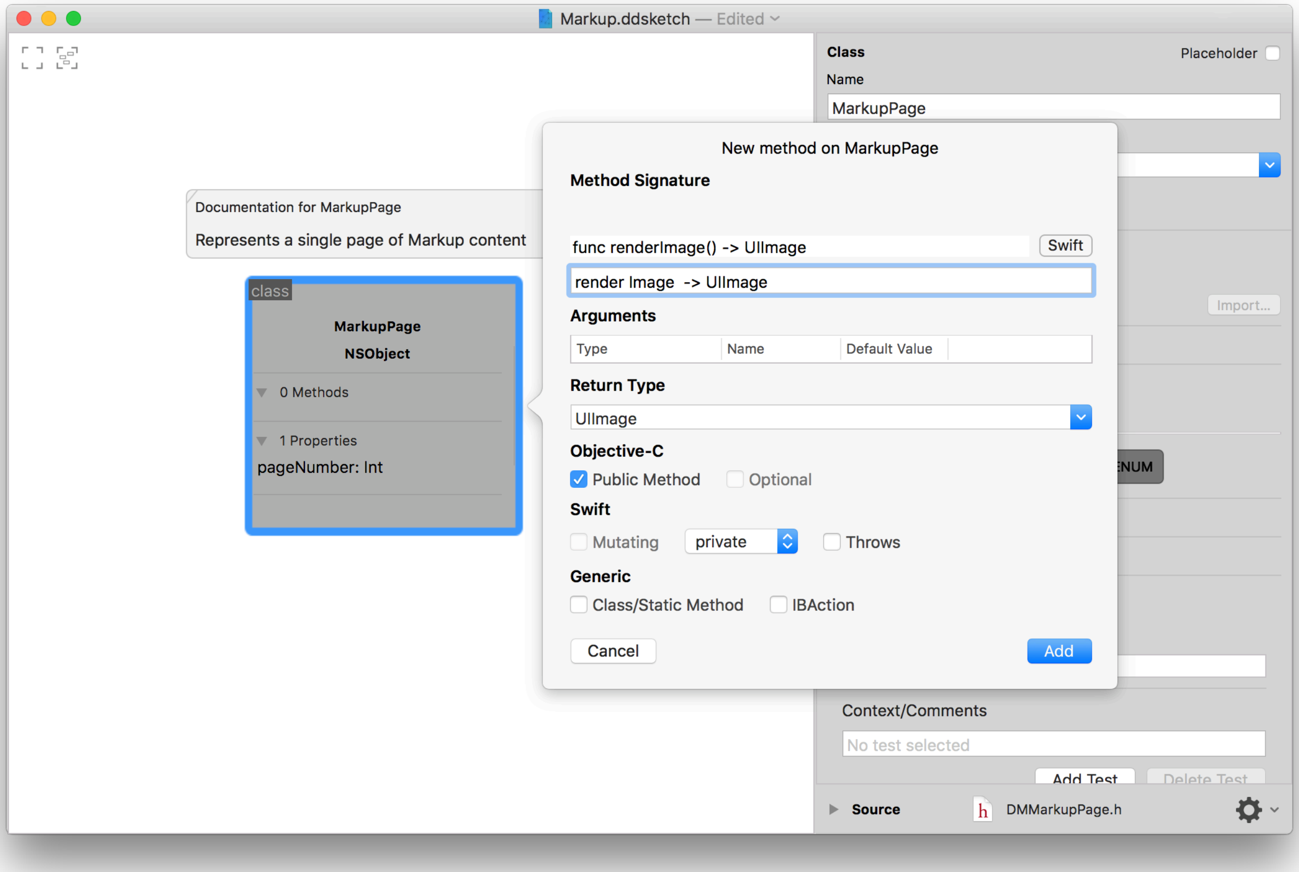
□ Protocol □ Copy

- You can enter the name in spaced language and that will be converted to Camel-Case e.g *"page number"* becomes *pageNumber*
- You can enter the type Swift-style with a colon *name:type* or just fill in the Type field.

Click **Add** to commit the addition.

Adding a method

Select the object and press CMD+M (**Object > Add Method**)



Markup.ddsketch — Edited

Documentation for MarkupPage

Represents a single page of Markup content

class

MarkupPage

NSObject

0 Methods

1 Properties

pageNumber: Int

Class

Placeholder □

Name: MarkupPage

New method on MarkupPage

Method Signature

func renderImage() -> UIImage

render Image -> UIImage

Arguments

Type	Name	Default Value

Return Type: UIImage

Objective-C

☑ Public Method □ Optional

Swift

□ Mutating private □ Throws

Generic

□ Class/Static Method □ IBAction

Context/Comments

No test selected

Source DMMarkupPage.h

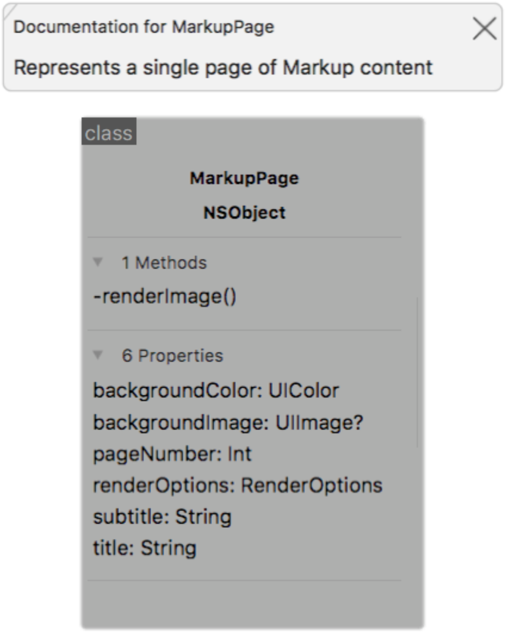
- You can enter the message in Swift like format. DevSketch will do its best to figure out what you want.
- Spaced signatures will get Camel-Cased.
- Use **(** to start arguments composition.
- Use **->** to start return type description.

e.g

- *do this(foo* becomes *func doThis(foo: Any)*
- *do that -> Thing* becomes *func doThat() -> Thing*
- *look at that(foo: Bar) Baz* becomes *lookAtThat(foo: Bar) -> Baz*
- *render image(completion:(UIImage)->()* becomes *renderImage(completion:(UIImage) -> ())*

Expand and collapse

The map representation of an object can be collapsed or expanded.



Documentation for MarkupPage

Represents a single page of Markup content

class

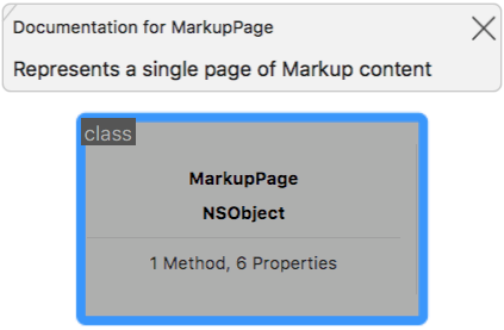
MarkupPage
NSObject

1 Methods

-renderImage()

6 Properties

backgroundColor: UIColor
backgroundImage: UIImage?
pageNumber: Int
renderOptions: RenderOptions
subtitle: String
title: String



Documentation for MarkupPage

Represents a single page of Markup content

class

MarkupPage
NSObject

1 Method, 6 Properties

- Use CMD+[to collapse (**Object > Collapse Representation**)
- Use CMD+] to expand (**Object > Expand Representation**)

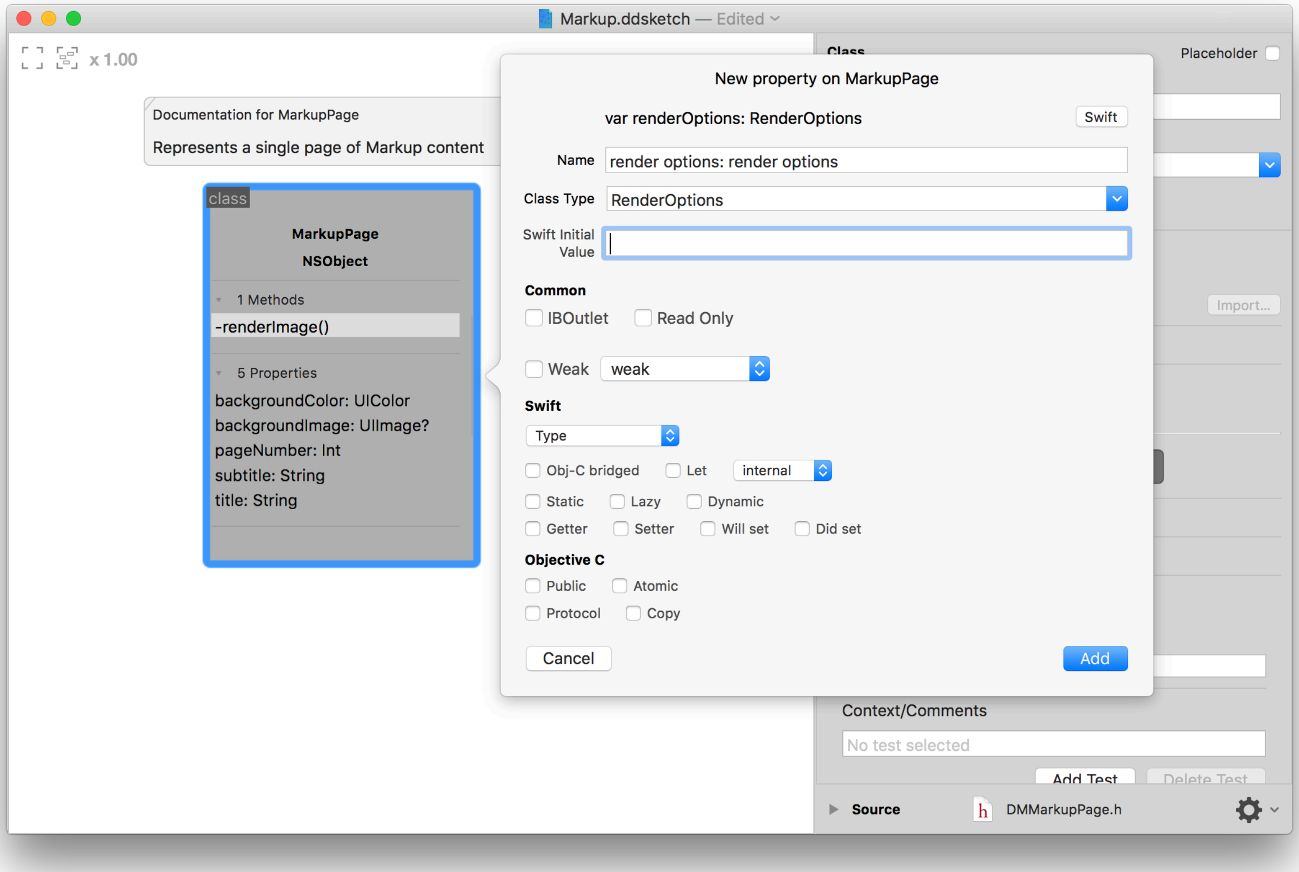
Expanded representations will hide the table at smaller magnifications.

Orphan Inspector: All about orphans

DevSketch will examine your object definition and look for types that don’t exist in your database or the imported SDK’s.

What this means is you can declare a type ahead of time:

- Declare *RenderOptions* as a type.



Documentation for MarkupPage

Represents a single page of Markup content

class

MarkupPage

NSObject

1 Methods

-renderImage()

5 Properties

backgroundColor: UIColor
backgroundImage: UIImage?
pageNumber: Int
subtitle: String
title: String

New property on MarkupPage

var renderOptions: RenderOptions

Swift

Name: render options: render options

Class Type: RenderOptions

Swift Initial Value:

Common

□ IBOutlet □ Read Only

□ Weak weak

Swift

Type

□ Obj-C bridged □ Let internal

□ Static □ Lazy □ Dynamic

□ Getter □ Setter □ Will set □ Did set

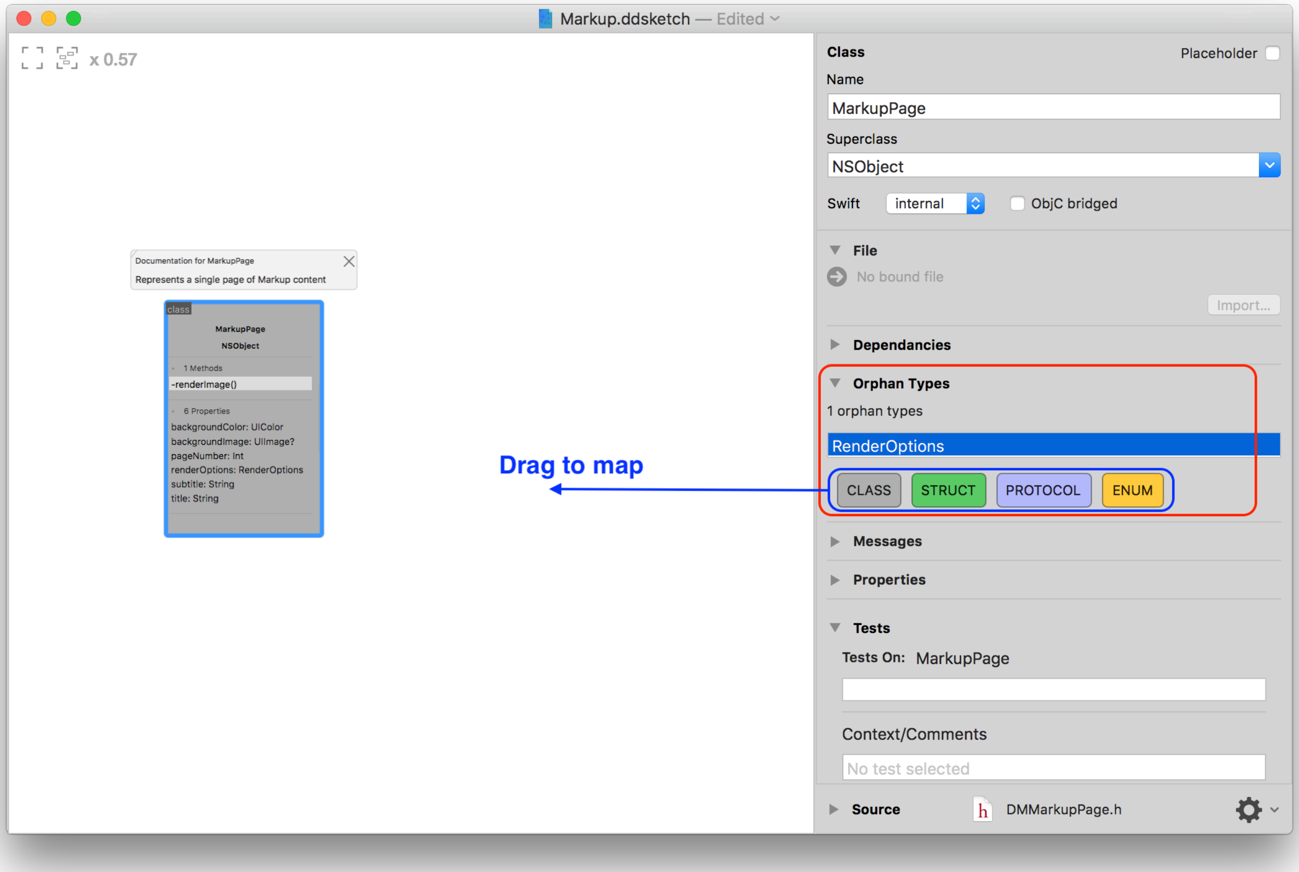
Objective C

□ Public □ Atomic

□ Protocol □ Copy

Cancel Add

- *RenderOptions* appears as an orphan type. Drag a object type to the map to create.



Markup.ddsketch — Edited

x 0.57

Documentation for MarkupPage

Represents a single page of Markup content

class

MarkupPage

NSObject

1 Methods

-renderImage()

6 Properties

backgroundColor: UIColor
backgroundImage: UIImage?
pageNumber: Int
renderOptions: RenderOptions
subtitle: String
title: String

Drag to map

Class

Placeholder □

Name: MarkupPage

Superclass: NSObject

Swift: internal

□ ObjC bridged

File

No bound file

Dependancies

Orphan Types

1 orphan types

RenderOptions

Messages

Properties

Tests

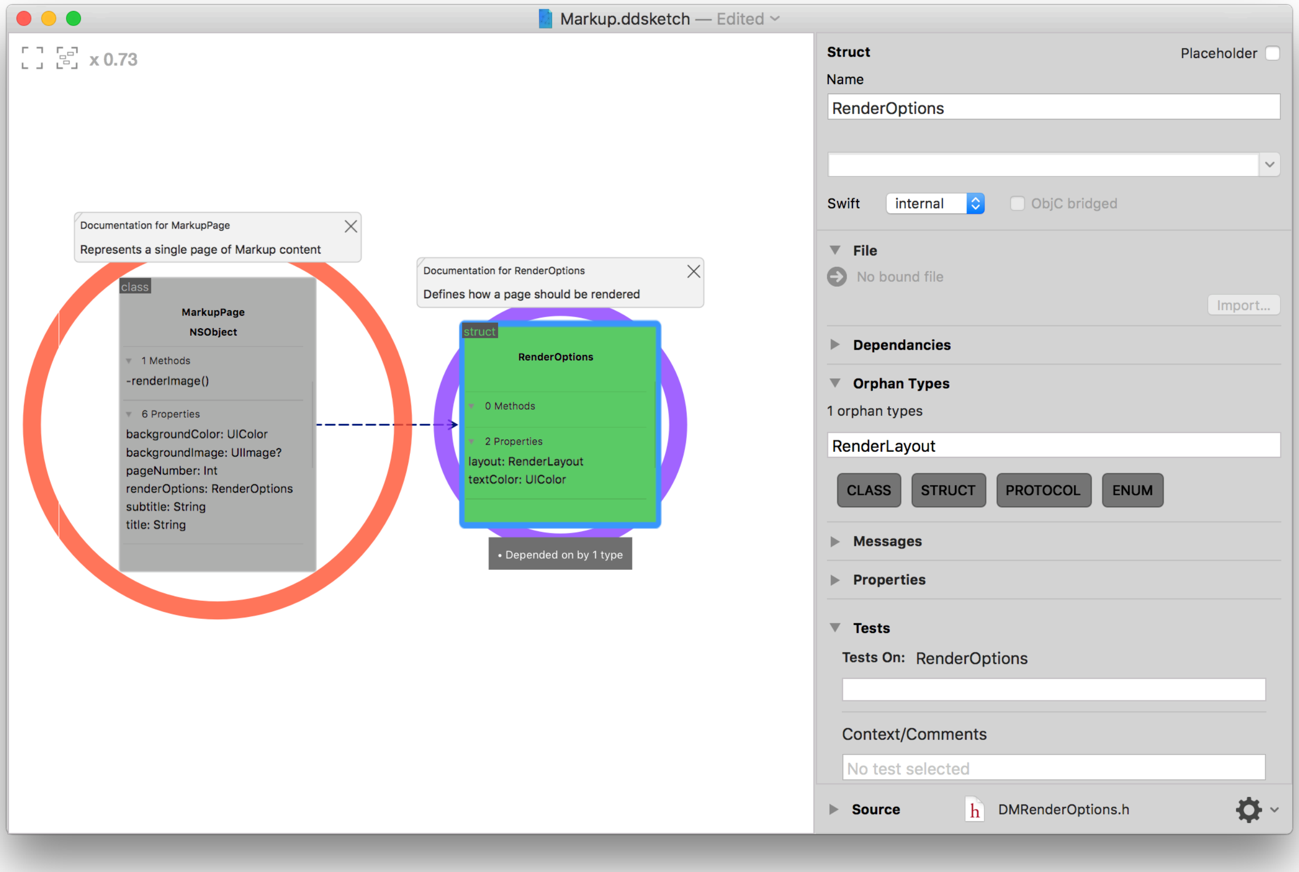
Tests On: MarkupPage

Context/Comments

No test selected

Source: DMMarkupPage.h

- Start defining what *RenderOptions* should do and what attributes it should have.



Markup.ddsketch — Edited

Documentation for MarkupPage
Represents a single page of Markup content

Documentation for RenderOptions
Defines how a page should be rendered

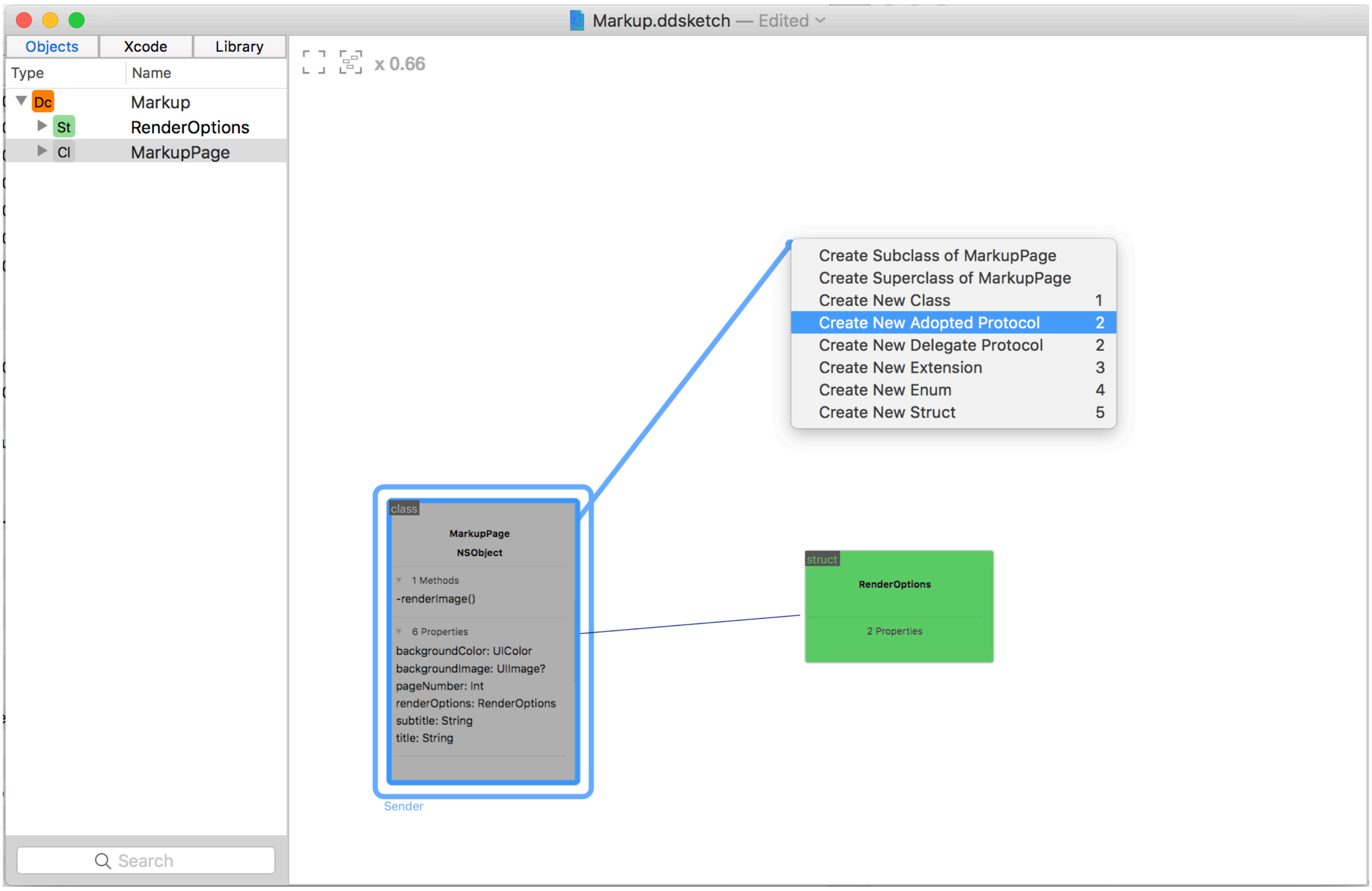
Struct
Name: RenderOptions
Swift: internal
No bound file
1 orphan types
RenderLayout
Tests On: RenderOptions
No test selected
Source: DMRenderOptions.h

The **Orphan Inspector** allows you to be flexible in your design and naming. If you are not sure what an object should be named or if it should even exist you can delay creation till you are ready.

Refactoring

While you are working you will inevitably discover points where you want to change the design.

- Decide that a page should be represented by a protocol. You need refactor some of the attributes of **MarkupPage** out.



1. Hold down *ALT* and drag away from the origin object.
2. Release the mouse where you want to place your new object.
3. Select what object type you want to create from the menu.

- Create a protocol called **PageDescription**

1. Select the attributes you want to shift to the protocol.
2.