

Building CRUDdy iOS Apps with SQLite3

An Introduction to SQLite3 for your iOS Application



Don Miller



GroundSpeed™
rapid web + mobile software

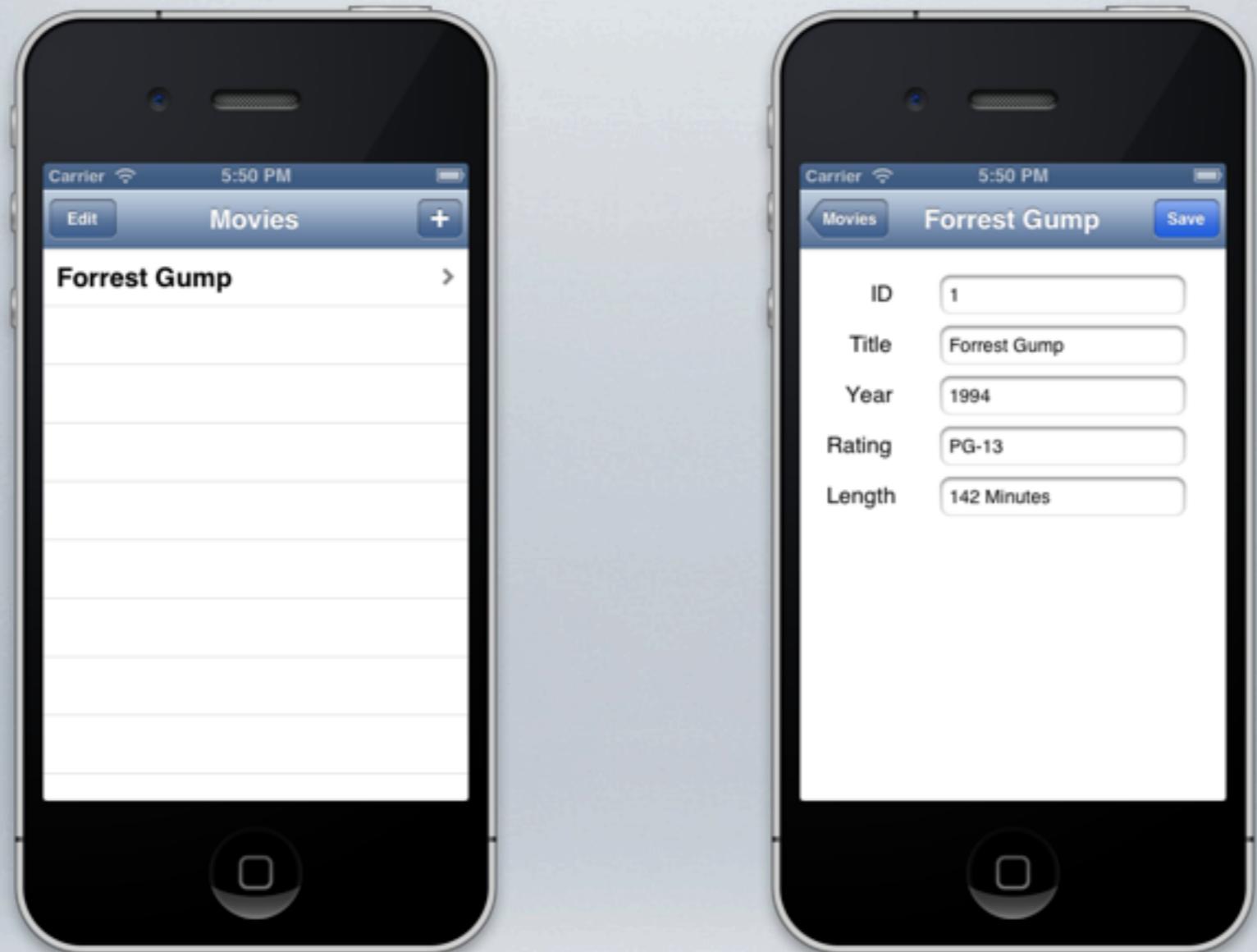
ABOUT SQLITE ON IPHONE/IPAD DEVICES

- Data Persistence
- libsqlite3.0.dylib

SQLite C Functions	Description:
sqlite3_open()	Opens specified database file. If the database file does not already exist, it is created.
sqlite3_close()	Closes a previously opened database file.
sqlite3_prepare_v2()	Prepares a SQL statement ready for execution.
sqlite3_step()	Executes a SQL statement previously prepared by the <code>sqlite3_prepare_v2()</code> function.
sqlite3_column_<type>()	Returns a data field from the results of a SQL retrieval operation where <type> is replaced by the data type of the data to be extracted (text, blob, bytes, int, int16 etc).
sqlite3_finalize()	Deletes a previously prepared SQL statement from memory.
sqlite3_exec()	Combines the functionality of <code>sqlite3_prepare_v2()</code> , <code>sqlite3_step()</code> and <code>sqlite3_finalize()</code> into a single function call.

OUR SIMPLE DATABASE

```
1
2
3 CREATE TABLE "movies" (
4   "id" INTEGER PRIMARY KEY AUTOINCREMENT,
5   "title" TEXT,
6   "year" INTEGER,
7   "rating" TEXT,
8   "length" TEXT
9 );
10
```



EXAMPLE APPLICATION

Choose a template for your new project

The screenshot shows the Xcode project template selection interface. On the left, there's a sidebar with categories for iOS (Application, Framework & Library, Other) and OS X (Application, Framework & Library, Application Plug-in, System Plug-in, Other). The 'Master-Detail Application' template is selected for iOS, indicated by a blue highlight. Other visible templates include OpenGL Game, Page-Based Application, Single View Application, Tabbed Application, Utility Application, and Empty Application. Below the selected template, its description is provided: "This template provides a starting point for a master-detail application. It provides a user interface configured with a navigation controller to display a list of items and also a split view on iPad." At the bottom are 'Cancel', 'Previous', and 'Next' buttons.

iOS

Application

Framework & Library

Other

OS X

Application

Framework & Library

Application Plug-in

System Plug-in

Other

Master-Detail Application

OpenGL Game

Page-Based Application

Single View Application

Tabbed Application

Utility Application

Empty Application

Master-Detail Application

This template provides a starting point for a master-detail application. It provides a user interface configured with a navigation controller to display a list of items and also a split view on iPad.

Cancel

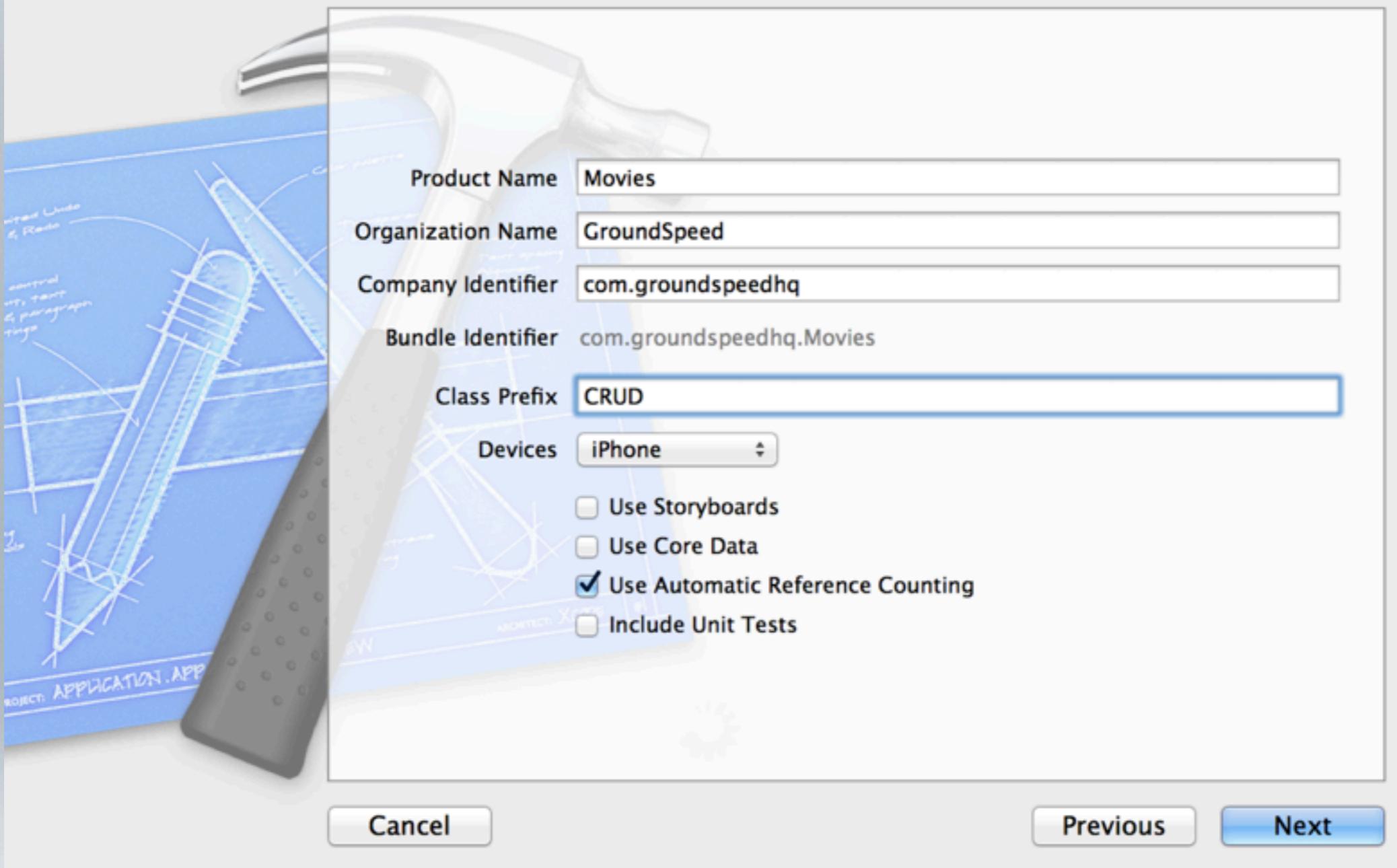
Previous

Next

START A MASTER-DETAIL APP



Choose options for your new project:



Product Name

Organization Name

Company Identifier

Bundle Identifier

Class Prefix

Devices

Use Storyboards

Use Core Data

Use Automatic Reference Counting

Include Unit Tests

NAME YOUR PROJECT



Movies
1 target, iOS SDK 6.0

- libssqlite3.0.dylib
- Movies
 - Models
 - Views
 - CRUDMasterViewController.xib
 - CRUDDetailViewController.xib
 - Controllers
 - CRUDMasterViewController.h
 - CRUDMasterViewController.m
 - CRUDDetailViewController.h
 - CRUDDetailViewController.m
 - CRUDAppDelegate.h
 - CRUDAppDelegate.m
 - Supporting Files
- Frameworks
- Products

PROJECT
Movies

TARGETS
Movies

Summary **Info** **Build Settings**
Retina Display

Launch Images

Retina (3.5-inch)

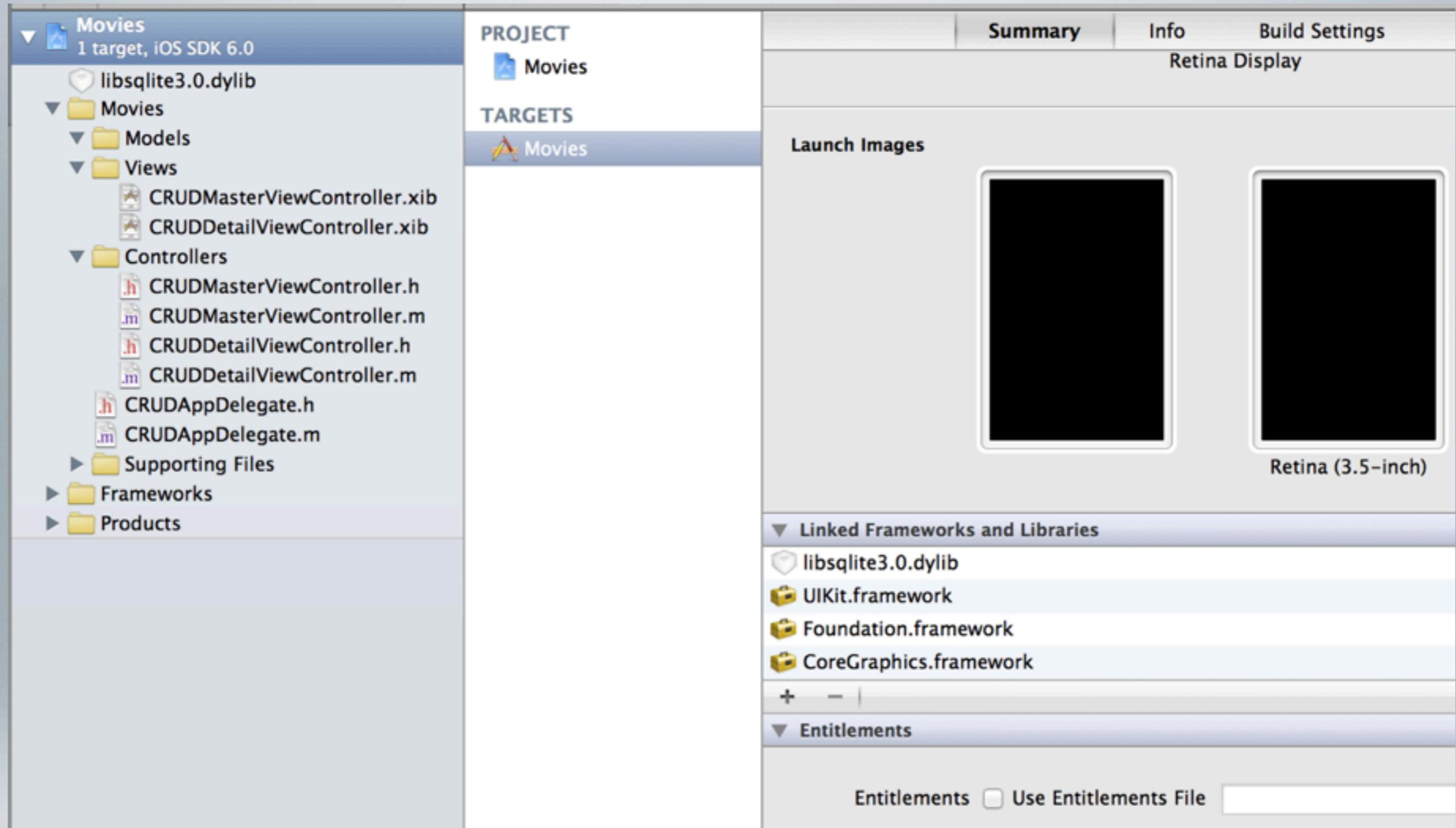
Linked Frameworks and Libraries

- libssqlite3.0.dylib
- UIKit.framework
- Foundation.framework
- CoreGraphics.framework

+ - |

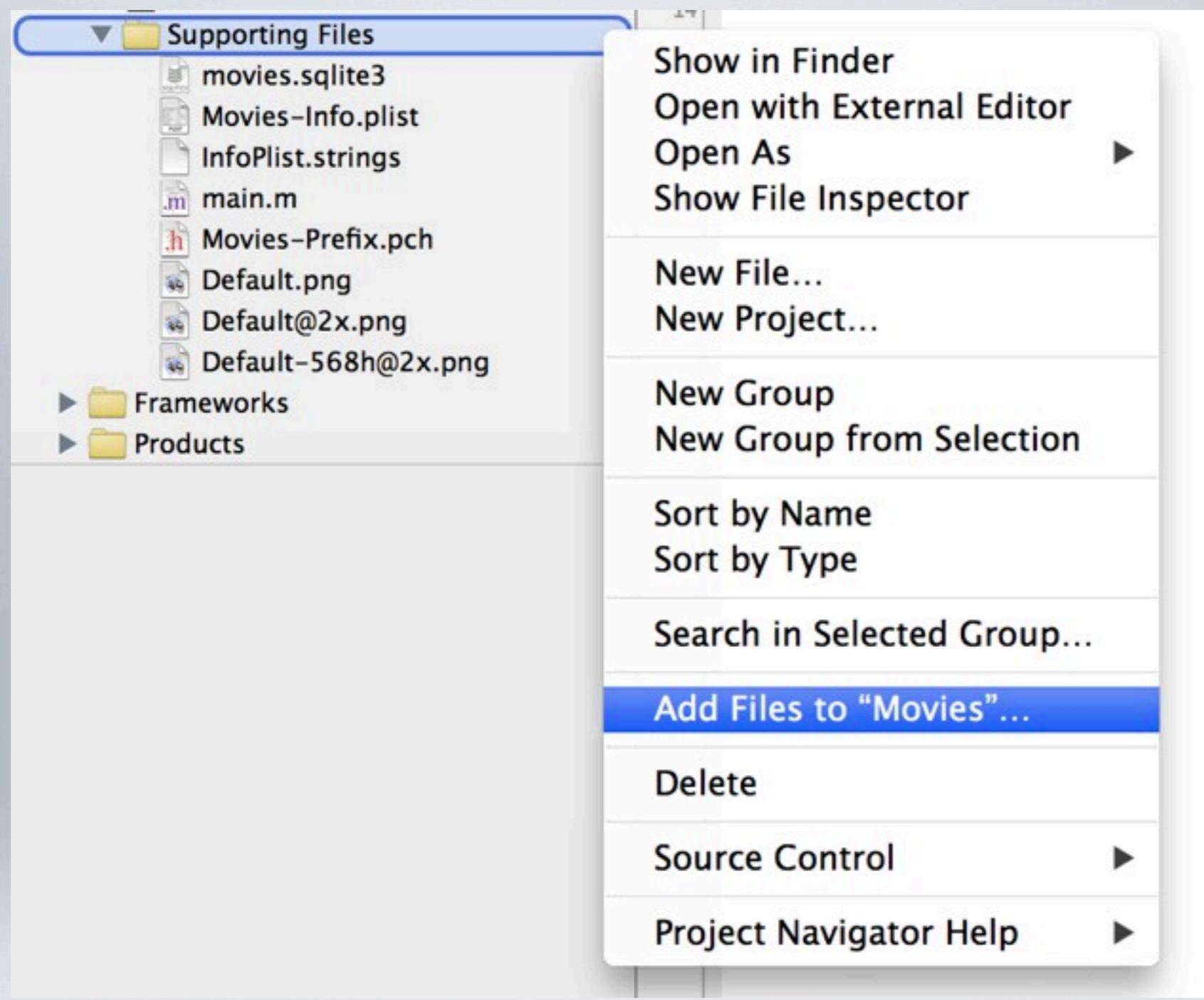
Entitlements

Entitlements Use Entitlements File

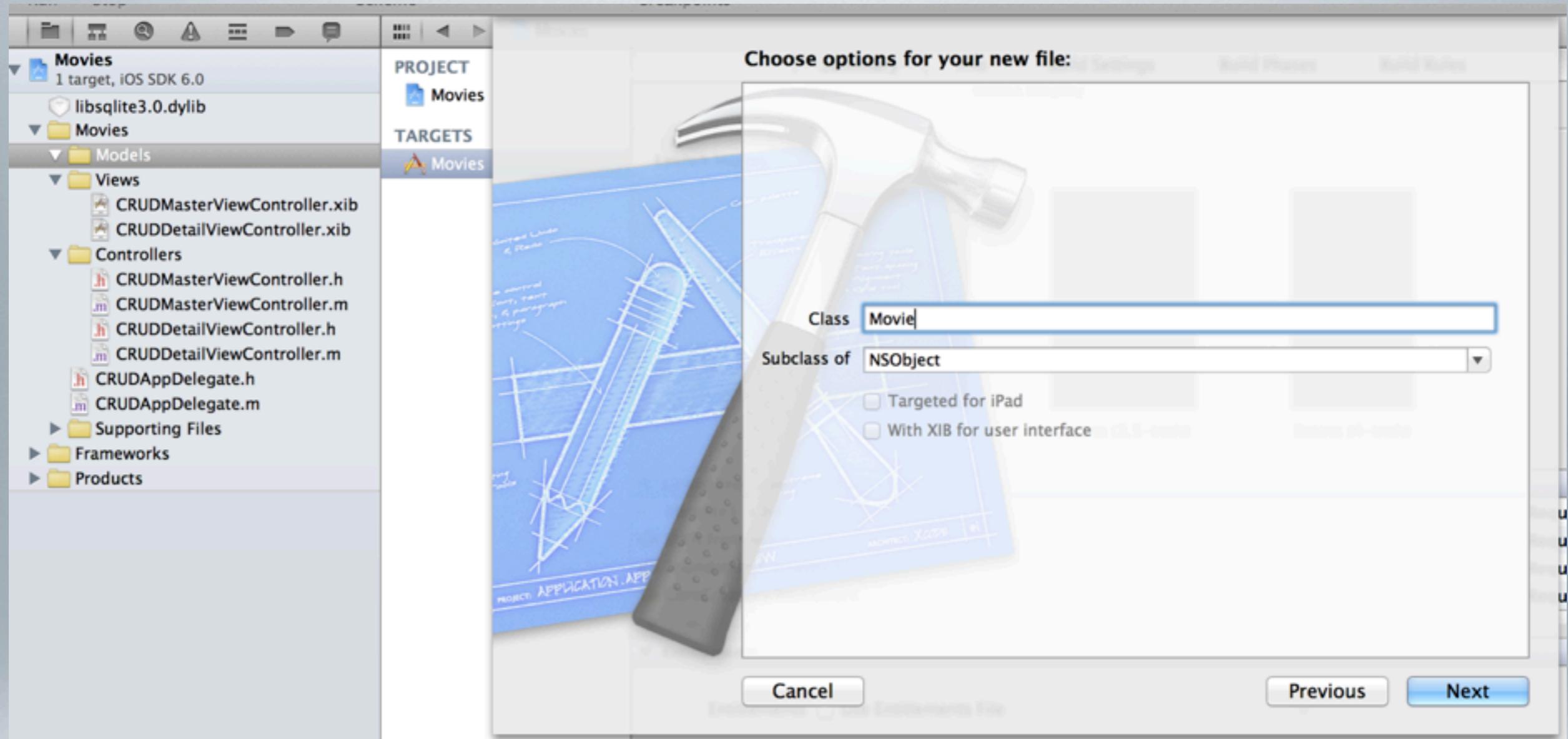


ADD SQLITE LIBRARY





ADD DATABASE TO PROJECT



CREATE MOVIES MODEL

```

9 #import <Foundation/Foundation.h>
10
11 @interface Movie : NSObject
12
13 @property(nonatomic) int intMovieId;
14 @property(nonatomic, strong) NSString *strTitle;
15 @property(nonatomic) int intYear;
16 @property(nonatomic, strong) NSString *strRating;
17 @property(nonatomic, strong) NSString *strLength;
18
19 -(id) initWithMovieId: (int) pMovieId
20     WithTitle: (NSString *) pTitle
21     WithYear: (int) pYear
22     WithRating: (NSString *) pRating
23     WithLength: (NSString *) pLength;
24
25 @end
26

```

GCODE MOVIE HEADER

movieheadercrud



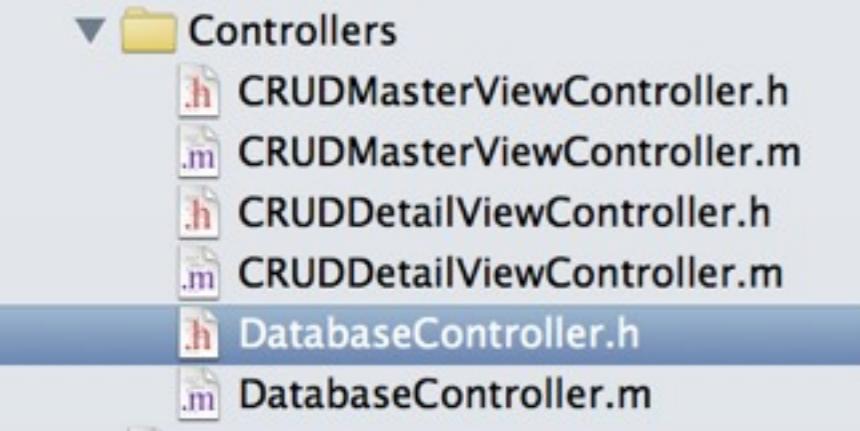
```

9 #import "Movie.h"
10
11 @implementation Movie
12
13 -(id) initWithMovieId: (int) pMovieId
14    WithTitle:(NSString *) pTitle
15     WithYear: (int) pYear
16     WithRating: (NSString *) pRating
17     WithLength: (NSString *) pLength
18 {
19     if ((self=[super init]))
20     {
21         self.intMovieId = pMovieId;
22         self.strTitle = pTitle;
23         self.intYear = pYear;
24         self.strRating = pRating;
25         self.strLength = pLength;
26         return self;
27     }
28     else
29     {
30         return nil;
31     }
32 }
33
34 @end

```

CODE MOVIE IMPLEMENTATION

movieimplementcrud



```
9 #import <Foundation/Foundation.h>
10
11 @interface DatabaseController : NSObject
12
13 @end
14
```

CREATE DATABASE CONTROLLER

```

9 #import <Foundation/Foundation.h>
10 #import <sqlite3.h> // Observe this import
11 #import "Movie.h" // Observe this import too
12
13 @interface DatabaseController : NSObject
14 {
15     NSString *dbName;
16 }
17
18 @property (nonatomic,retain) NSFileManager *fileMgr;
19 @property (nonatomic,retain) NSString *homeDir;
20 @property (nonatomic,retain) NSString *title;
21
22 -(void) initializeDatabase;
23 -(void) closeDatabase;
24 -(void) CopyDbToDocumentsFolder;
25 -(NSString *) GetDocumentDirectory;
26
27 -(NSMutableArray*) getAllMovies;
28
29 -(void) insertMovie:(NSString *) fldTitle
30             Year:(int) fldYear
31             Rating:(NSString *) fldRating
32             Length:(NSString *) fldLength;
33
34 -(void) updateMovie:(int) fldMovieId
35             Title:(NSString *) fldTitle
36             Year:(int) fldYear
37             Rating:(NSString *) fldRating
38             Length:(NSString *) fldLength;
39
40 -(void) deleteMovie:(int) fldMovieId;
41 @end

```

CREATE DB CONTROLLER HEADER

dbcontrollerheadercrud

```

13 sqlite3 *sqlDatabase;
14
15 -(id) init
16 { // Call super init to invoke superclass initiation code
17     if ((self = [super init]))
18     { // set the reference to the database
19         dbName = @"movies.sqlite3";
20         [self initializeDatabase];
21     }
22     return self;
23 }
24
25 - (void) initializeDatabase
26 {
27     [self CopyDbToDocumentsFolder];
28
29     //Get list of directories in Document path
30     NSArray *dirPath = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
31 NSUserDomainMask, YES);
32
33     //Define new path for database
34     NSString *path = [[dirPath objectAtIndex:0] stringByAppendingPathComponent:dbName];
35
36     // Get the database from the application bundle.
37     NSLog(@"%@", path);
38     // Open the database.
39     if (sqlite3_open([path UTF8String], &sqlDatabase) == SQLITE_OK)
40     {
41         NSLog(@"Opening Database");
42     }
43     else
44     {
45         // Call close to properly clean up
46         sqlite3_close(sqlDatabase);
47         NSAssert1(0, @"Failed to open database: '%s'.",
48                 sqlite3_errmsg(sqlDatabase));
49     }
50 }

```

Don't forget to Import the Movie.h file.

IMPLEMENT INITIALIZER AND INITIALIZE

dbcontrollerinitcrud



```

50 -(void) CopyDbToDocumentsFolder
51 {
52     NSError *err=nil;
53
54     _fileMgr = [NSFileManager defaultManager];
55
56     NSString *dbpath = [[[NSBundle mainBundle] resourcePath]
57                           stringByAppendingPathComponent:dbName];
58
59     NSString *copydbpath = [self.GetDocumentDirectory
60                           stringByAppendingPathComponent:dbName];
61
62     // lets only copy the db if it does not exist
63     if (![_fileMgr fileExistsAtPath:copydbpath])
64     {
65         if(![_fileMgr copyItemAtPath:dbpath toPath:copydbpath error:&err])
66         {
67             UIAlertView *tellErr = [[UIAlertView alloc] initWithTitle:_title
68                                     message:@"Unable to copy database." delegate:self
69                                     cancelButtonTitle:@"OK" otherButtonTitles:nil];
70             [tellErr show];
71         }
72     }
73
74     -(NSString *) GetDocumentDirectory
75     {
76         _fileMgr = [NSFileManager defaultManager];
77         _homeDir = [NSHomeDirectory()
78                     stringByAppendingPathComponent:@"Documents/"];
79
80         return _homeDir;
81     }

```

COPY DB TO DOCUMENTS FOLDER

dbcontrollercopydbcrud



```
79 -(void) closeDatabase
80 {
81     // Close the database.
82     if (sqlite3_close(sqlDatabase) != SQLITE_OK) {
83         NSAssert1(0, @"Error: failed to close database: '%s'.",
84                 sqlite3_errmsg(sqlDatabase));
85     }
86 }
87
```

CODE THE CLOSE DATABASE METHOD

dbcontrollerclosedberud



```

89 -(NSMutableArray*) getAllMovies
90 {
91     NSMutableArray *arrayMovies = [[NSMutableArray alloc] init];
92
93     const char *mySql = "SELECT id, title, year, rating, length FROM movies";
94     const int movieId = 0;
95     const int movieTitle = 1;
96     const int movieYear = 2;
97     const int movieRating = 3;
98     const int movieLength = 4;
99
100    // Declare the SQLite statement object that will hold our result set
101    sqlite3_stmt *myStatement;
102
103    int sqlResult = sqlite3_prepare_v2(sqlDatabase, mySql, -1, &myStatement, NULL);
104    if (sqlResult== SQLITE_OK)
105    {
106        while (sqlite3_step(myStatement) == SQLITE_ROW)
107        {
108            char *name = (char *)sqlite3_column_text(myStatement, movieTitle);
109            char *rating = (char *)sqlite3_column_text(myStatement, movieRating);
110            char *length = (char *)sqlite3_column_text(myStatement, movieLength);
111
112            Movie *movies = [[Movie alloc]
113                            initWithMovieId:sqlite3_column_int(myStatement, movieId)
114                            WithTitle:(name) ? [NSString stringWithUTF8String:name] : @"""
115                            WithYear:sqlite3_column_double(myStatement, movieYear)
116                            WithRating:(rating) ? [NSString stringWithUTF8String:rating] : @"""
117                            WithLength:(length) ? [NSString stringWithUTF8String:length] : @""];
118            [arrayMovies addObject:movies];
119            movies = nil;
120        }
121        sqlite3_finalize(myStatement);
122    }
123    else {
124        NSLog(@"Problem with the database:");
125        NSLog(@"%@",sqlResult);
126    }
127    return arrayMovies;
128}

```

PUT ALL THE MOVIES INTO AN ARRAY

dbcontrollergetmoviescrud



```

130 -(void) insertMovie:(NSString *) fldTitle
131     Year:(int) fldYear
132     Rating:(NSString *) fldRating
133     Length:(NSString *) fldLength
134 {
135     NSString *documentPath = [self.GetDocumentDirectory stringByAppendingPathComponent:dbName];
136
137     if(!(sqlite3_open([documentPath UTF8String], &sqlDatabase) == SQLITE_OK))
138     {
139         NSLog(@"An error has occurred.");
140         return;
141     }
142     else
143     {
144         NSString *insertSQL = [NSString stringWithFormat:
145             @"INSERT INTO Movie(title, year, rating, length) VALUES ('%@','%i',
146             %@, '%@')", fldTitle, fldYear, fldRating, fldLength];
147
148         const char *sql = [insertSQL UTF8String];
149
150         sqlite3_stmt *sqlStatement;
151         if(sqlite3_prepare_v2(sqlDatabase, sql, -1, &sqlStatement, NULL) != SQLITE_OK)
152         {
153             NSLog(@"Problem with prepare statement");
154             return;
155         }
156         else
157         {
158             if(sqlite3_step(sqlStatement)==SQLITE_DONE)
159             {
160                 sqlite3_finalize(sqlStatement);
161                 sqlite3_close(sqlDatabase);
162             }
163         }
164     }
}

```

CREATE METHOD TO INSERT MOVIE

dbcontrollerinsertmoviecrud



```

166 -(void) updateMovie:(int) fldMovieId
167     Title:(NSString *) fldTitle
168     Year:(int) fldYear
169     Rating:(NSString *) fldRating
170     Length:(NSString *) fldLength
171 {
172     NSString *documentPath = [self.GetDocumentDirectory stringByAppendingPathComponent:dbName];
173
174     if(!(sqlite3_open([documentPath UTF8String], &sqlDatabase) == SQLITE_OK))
175     {
176         NSLog(@"An error has occurred.");
177         return;
178     }
179     else
180     {
181         NSString *updateSQL = [NSString stringWithFormat:
182             @"UPDATE movies SET Title='%@', year=%i, rating=%@, length='@'
183             WHERE id=%i", fldTitle, fldYear, fldRating, fldLength,
184             fldMovieId];
185
186         const char *sql = [updateSQL UTF8String];
187
188         sqlite3_stmt *sqlStatement;
189         if(sqlite3_prepare_v2(sqlDatabase, sql, -1, &sqlStatement, NULL) != SQLITE_OK)
190         {
191             NSLog(@"Problem with prepare statement");
192             return;
193         }
194         else
195         {
196             if(sqlite3_step(sqlStatement)==SQLITE_DONE)
197             {
198                 sqlite3_finalize(sqlStatement);
199                 sqlite3_close(sqlDatabase);
200             }
201         }
202     }
203 }

```

CREATE METHOD TO UPDATE MOVIE

dbcontrollerupdatemoviecrud



```

203 -(void) deleteMovie:(int) fldMovieId
204 {
205     NSString *documentPath = [self.GetDocumentDirectory stringByAppendingPathComponent:dbName];
206
207     if(!(sqlite3_open([documentPath UTF8String], &sqlDatabase) == SQLITE_OK))
208     {
209         NSLog(@"An error has occurred.");
210         return;
211     }
212     else
213     {
214         NSString *deleteSQL = [NSString stringWithFormat:@"DELETE FROM movies WHERE id = '%i'", fldMovieId];
215         const char *sql = [deleteSQL UTF8String];
216
217         sqlite3_stmt *sqlStatement;
218         if(sqlite3_prepare_v2(sqlDatabase, sql, -1, &sqlStatement, NULL) != SQLITE_OK)
219         {
220             NSLog(@"Problem with prepare statement");
221             return;
222         }
223         else
224         {
225             if(sqlite3_step(sqlStatement)==SQLITE_DONE)
226             {
227                 sqlite3_finalize(sqlStatement);
228                 sqlite3_close(sqlDatabase);
229             }
230         }
231     }
232 }

```

CREATE FINAL METHOD TO DELETE MOVIE

dbcontrollerdeletemoviecrud



```
9 #import <UIKit/UIKit.h>
10 #import "Movie.h"
11 #import "DatabaseController.h"
12
13 @class CRUDDetailViewController;
14
15 @interface CRUDMasterViewController : UITableViewController
16
17 @property (strong, nonatomic) CRUDDetailViewController *detailViewController;
18 @property (strong, nonatomic) NSMutableArray *arrayMovies;
19
20 -(void)loadMovieObject;
21
22
23 @end
```

UPDATE MASTERVIEW CONTROLLER HEADER

masterviewheadercrud

```
20 - (id)initWithNibName:(NSString *)NibNameOrNil bundle:(NSBundle *)NibNameOrNil  
21 {  
22     self = [super initWithNibNameNibNameOrNil bundleNibNameOrNil];  
23     if (self) {  
24         self.title = NSLocalizedString(@"Movies", @"Movies");  
25     }  
26     return self;  
27 }
```



ADD THE TITLE TO THE NAVIGATION

```

29 -(void)viewDidAppear:(BOOL)animated
30 {
31     [self loadMovieObject];
32     [self.tableView reloadData];
33 }
34
35 -(void)viewDidLoad
36 {
37     [super viewDidLoad];
38     // Do any additional setup after loading the view, typically from a nib.
39     self.navigationItem.leftBarButtonItem = self.editButtonItem;
40
41     [self loadMovieObject];
42
43     UIBarButtonItem * addButton = [[UIBarButtonItem alloc] initWithBarButtonSystemItem:
44         UIBarButtonSystemItemAdd target:self action:@selector(insertNewObject:)];
45     self.navigationItem.rightBarButtonItem = addButton;
46 }
47

```

ADD VIEWDIDAPPEAR AND UPDATE VIEWDIDLLOAD

masterviewappearcrud



```
135 #pragma mark User Methods  
136  
137 -(void) loadMovieObject  
{  
    DatabaseController *dbController = [[DatabaseController alloc] init];  
    // Fire the dbController getAllCustomers method to fill our array  
    _arrayMovies = [dbController getAllMovies];  
    // Release the dbAccess object to free its memory  
    dbController=nil;  
}
```

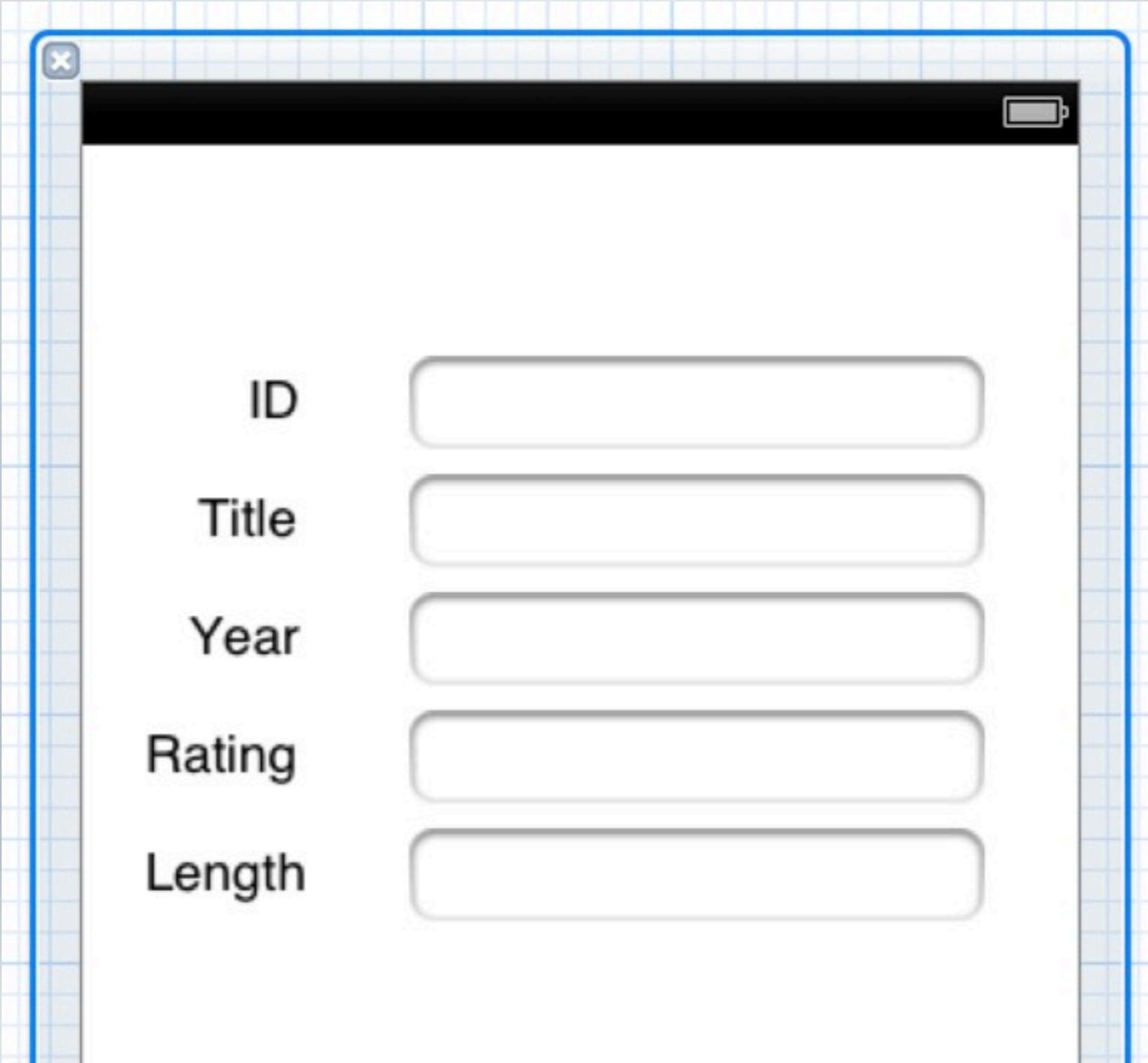
CODE THE LOADMOVIEOBJECT METHOD

masterviewloadmoviescrud



```
66 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
67 {
68     return 1;
69 }
70
71 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
72 {
73     return _arrayMovies.count; ←
74 }
75
76 // Customize the appearance of table view cells.
77 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
78 {
79     static NSString *CellIdentifier = @"Cell";
80
81     UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
82     if (cell == nil) {
83         cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier];
84         cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;
85     }
86
87     Movie *movies = [_arrayMovies objectAtIndex:[indexPath row]];
88     cell.textLabel.text = movies.strTitle; ←
89
90     return cell;
91 }
92
93 }
```

MODIFY TABLEVIEW METHODS TO RETURN MOVIES



MODIFY THE DETAIL VIEW TO HANDLE FIELDS

```

9 #import <UIKit/UIKit.h>
10 #import "Movie.h"
11 #import "DatabaseController.h"
12
13
14 @interface CRUDDetailViewController : UIViewController
15
16 @property (strong, nonatomic) id detailItem;
17 @property (weak, nonatomic) IBOutlet UILabel *detailDescriptionLabel;
18
19 @property (nonatomic, strong) IBOutlet UITextField *txtMovieID;
20 @property (nonatomic, strong) IBOutlet UITextField *txtTitle;
21 @property (nonatomic, strong) IBOutlet UITextField *txtYear;
22 @property (nonatomic, strong) IBOutlet UITextField *txtRating;
23 @property (nonatomic, strong) IBOutlet UITextField *txtLength;
24
25 -(void)setLabelsForMovie: (Movie *) theMovie;
26 -(void)updateMovie:(id)sender;

```

ADD PROPERTIES TO THE DETAIL VIEW CONTROLLER

detailviewpropertiescrud



```
38 - (void)viewDidLoad
39 {
40     [super viewDidLoad];
41     // Do any additional setup after loading the view, typically from a nib.
42     [self configureView];
43
44     UIBarButtonItem *saveButton = [[UIBarButtonItem alloc]
45         initWithBarButtonSystemItem:UIBarButtonSystemItemSave target:self
46         action:@selector(updateMovie:)];
47     self.navigationItem.rightBarButtonItem = saveButton;
48 }
```

ADD SAVE BUTTON TO TOP RIGHT OF DETAILVIEW

saveButton



```

64 #pragma mark User Methods
65
66 -(void) setLabelsForMovie:(Movie *)theMovie
67 {
68     // Set the labels to the values passed of the passed customer object
69     _txtMovieID.text=[NSString stringWithFormat:@"%i", theMovie.intMovieId];
70     _txtTitle.text=theMovie.strTitle;
71     _txtYear.text=[NSString stringWithFormat:@"%i", theMovie.intYear];
72     _txtRating.text=theMovie.strRating;
73     _txtLength.text=theMovie.strLength;
74 }
75
76 -(void) updateMovie:(id)sender
77 {
78     // Create an instance of DBServer class.
79     DatabaseController *dbController = [[DatabaseController alloc] init];
80
81     [dbController updateMovie:[_txtMovieID.text intValue]
82                             Title:_txtTitle.text
83                             Year:[_txtYear.text intValue]
84                             Rating:_txtRating.text
85                             Length:_txtLength.text];
86
87     // Release the dbAccess object to free its memory
88     dbController=nil;
89 }

```

CREATE LABEL AND UPDATE METHODS FOR DETAIL

detailviewusermethodscrud



```

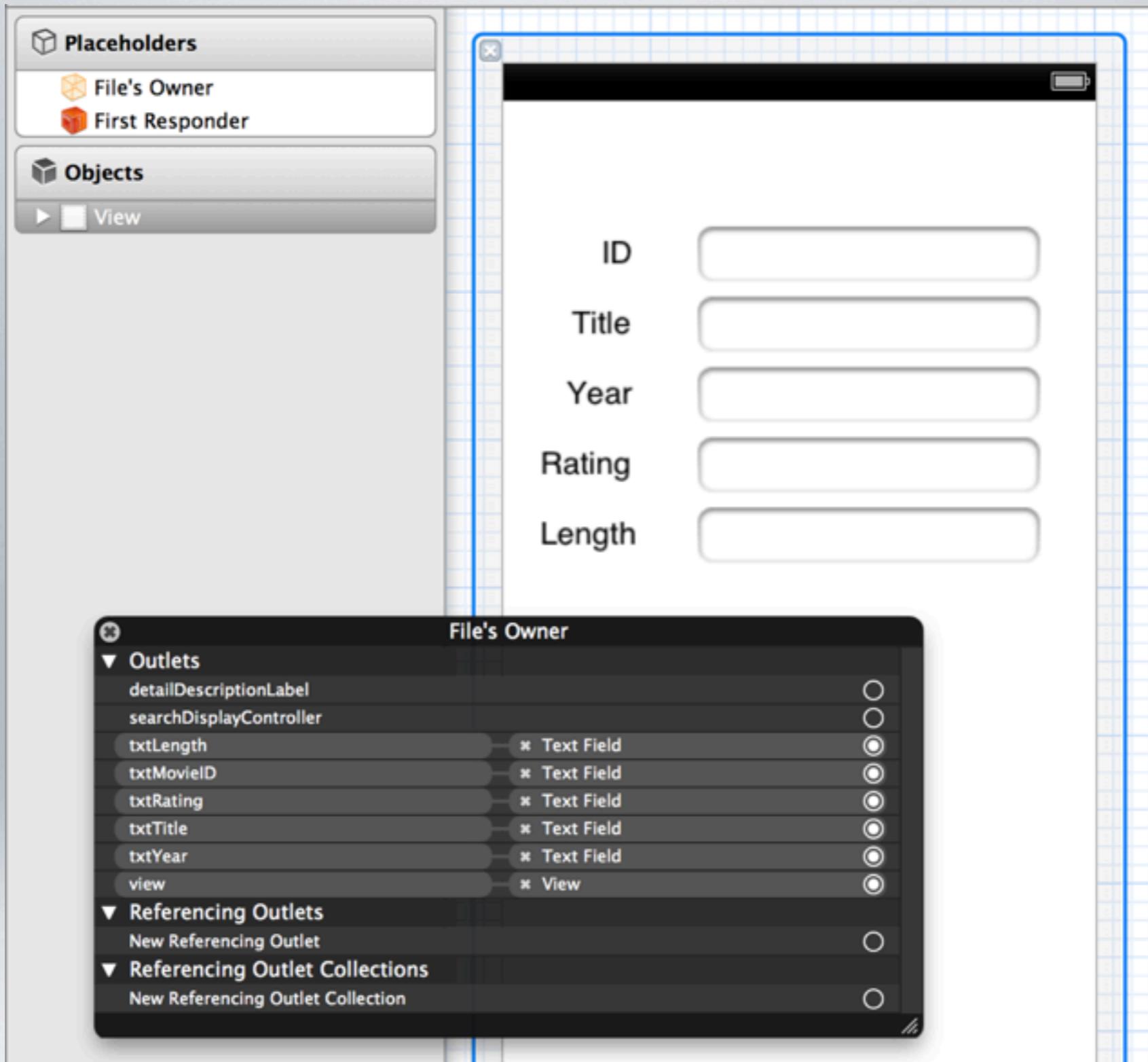
126 - (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
127 {
128     if (!self.detailViewController) {
129         self.detailViewController = [[CRUDDetailViewController alloc] initWithNibName:@"CRUDDetailViewController" bundle:nil];
130         ;
131     }
132 
133 //     NSDate *object = _objects[indexPath.row];
134 //     self.detailViewController.detailItem = object;
135 //     [self.navigationController pushViewController:self.detailViewController animated:YES];
136 
137     Movie *theMovie = [_arrayMovies objectAtIndex:[indexPath row]];
138 
139     // Set the title of the detail page
140     [self.detailViewController setTitle:theMovie.strTitle];
141 
142     // Push the detail controller on to the stack
143     [self.navigationController pushViewController:self.detailViewController animated:YES];
144 
145     // Populate the detail view by calling its setLabelsForMovie method
146     [self.detailViewController setLabelsForMovie:theMovie];
147     self.detailViewController=nil;
148 }
149

```

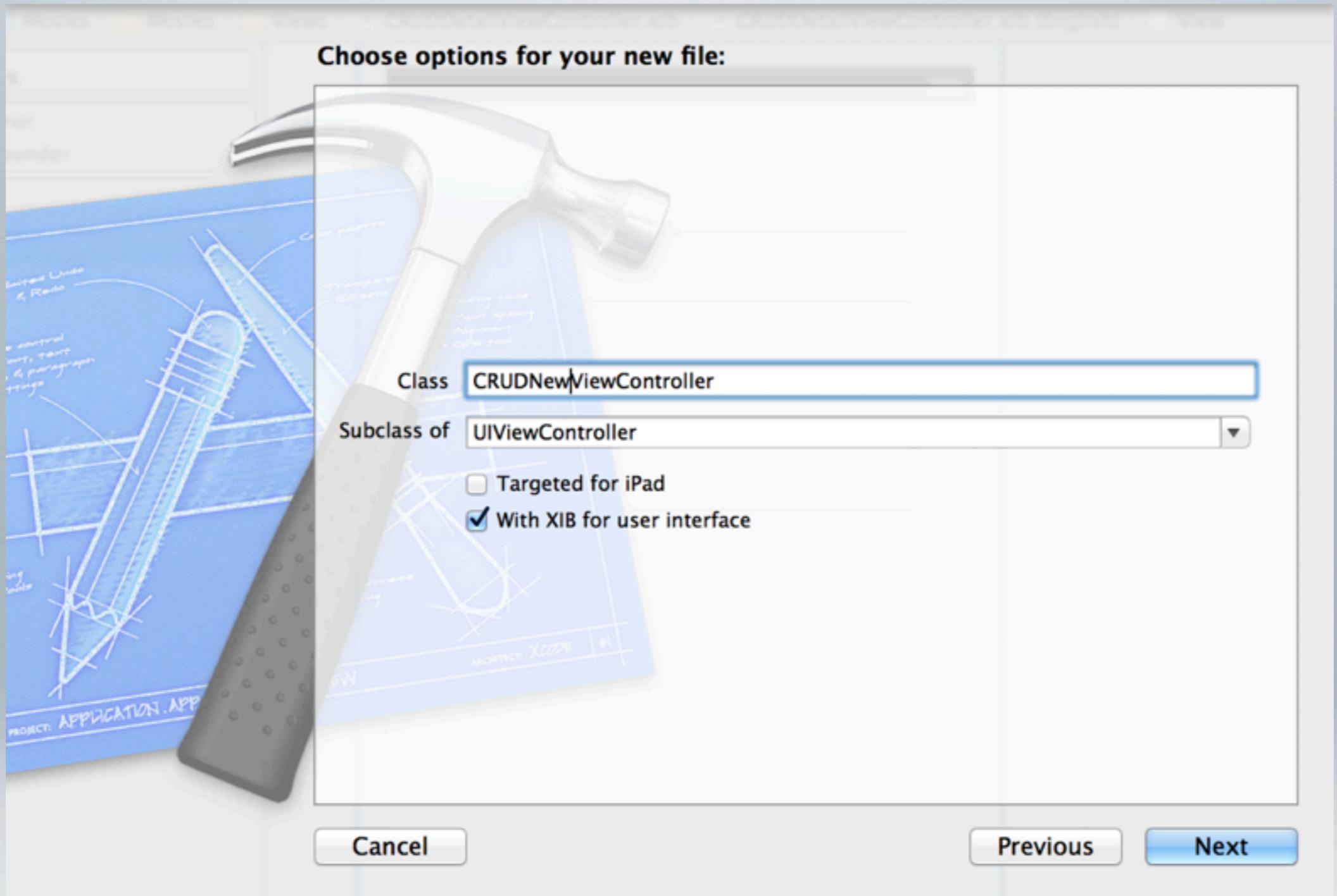
UPDATE DIDSELECTROW METHOD ON MASTERVIEW

masterviewdidselectcrud

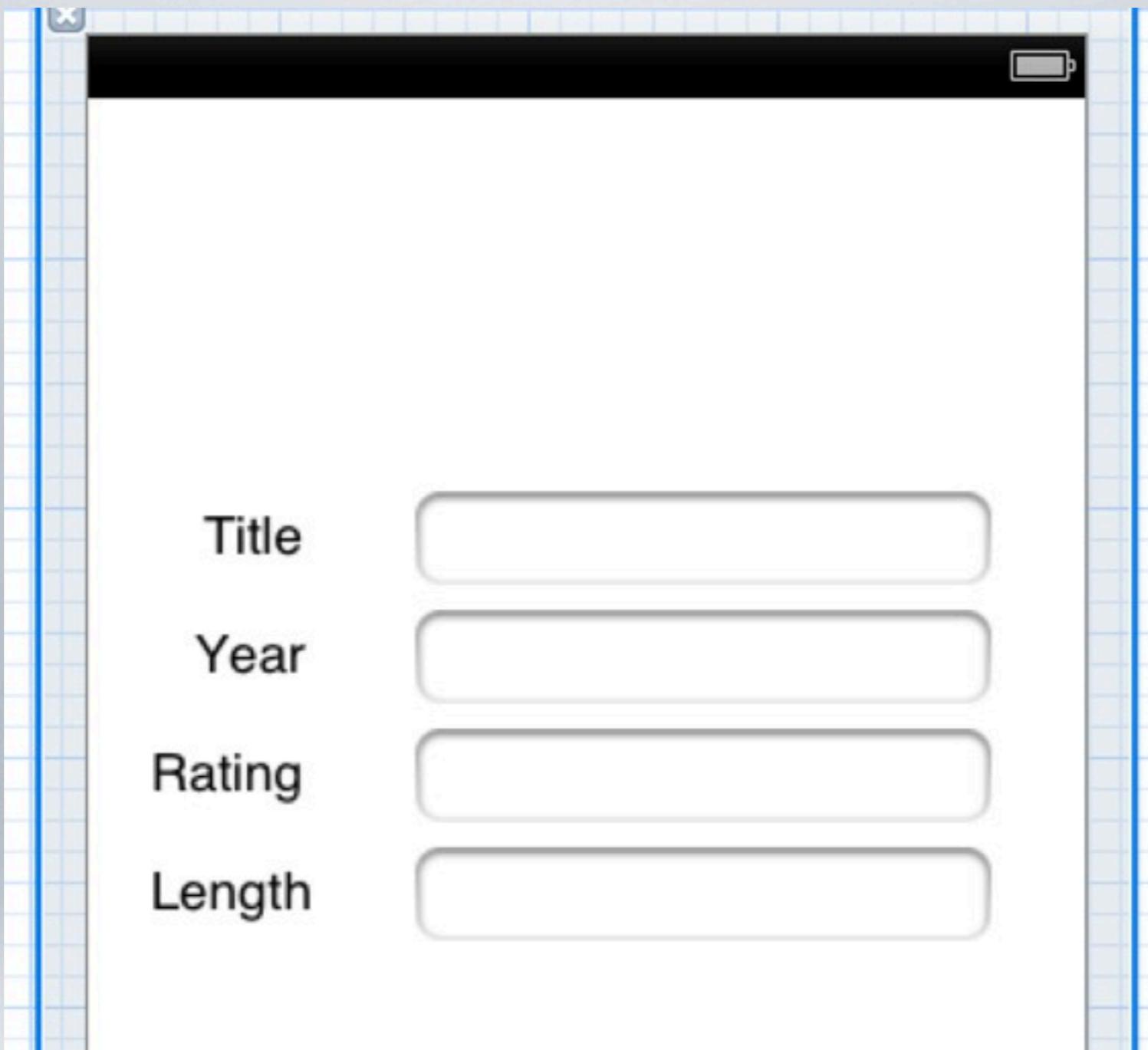




LINK YOUR PROPERTIES TO YOUR DETAIL IB LABELS



CREATE A NEW MOVIEVIEW CONTROLLER



CREATE LABELS AND TEXT BOXES FOR NEW FORM

```
9 #import <UIKit/UIKit.h>
10 #import "DatabaseController.h"
11
12 @interface CRUDNewViewController : UIViewController
13
14 @property (nonatomic, strong) IBOutlet UITextField *txtTitle;
15 @property (nonatomic, strong) IBOutlet UITextField *txtYear;
16 @property (nonatomic, strong) IBOutlet UITextField *txtRating;
17 @property (nonatomic, strong) IBOutlet UITextField *txtLength;
18
19 -(void)saveMovie:(id)sender;
20
21
22 @end
```

CREATE PROPERTIES IN THE NEW HEADER CLASS

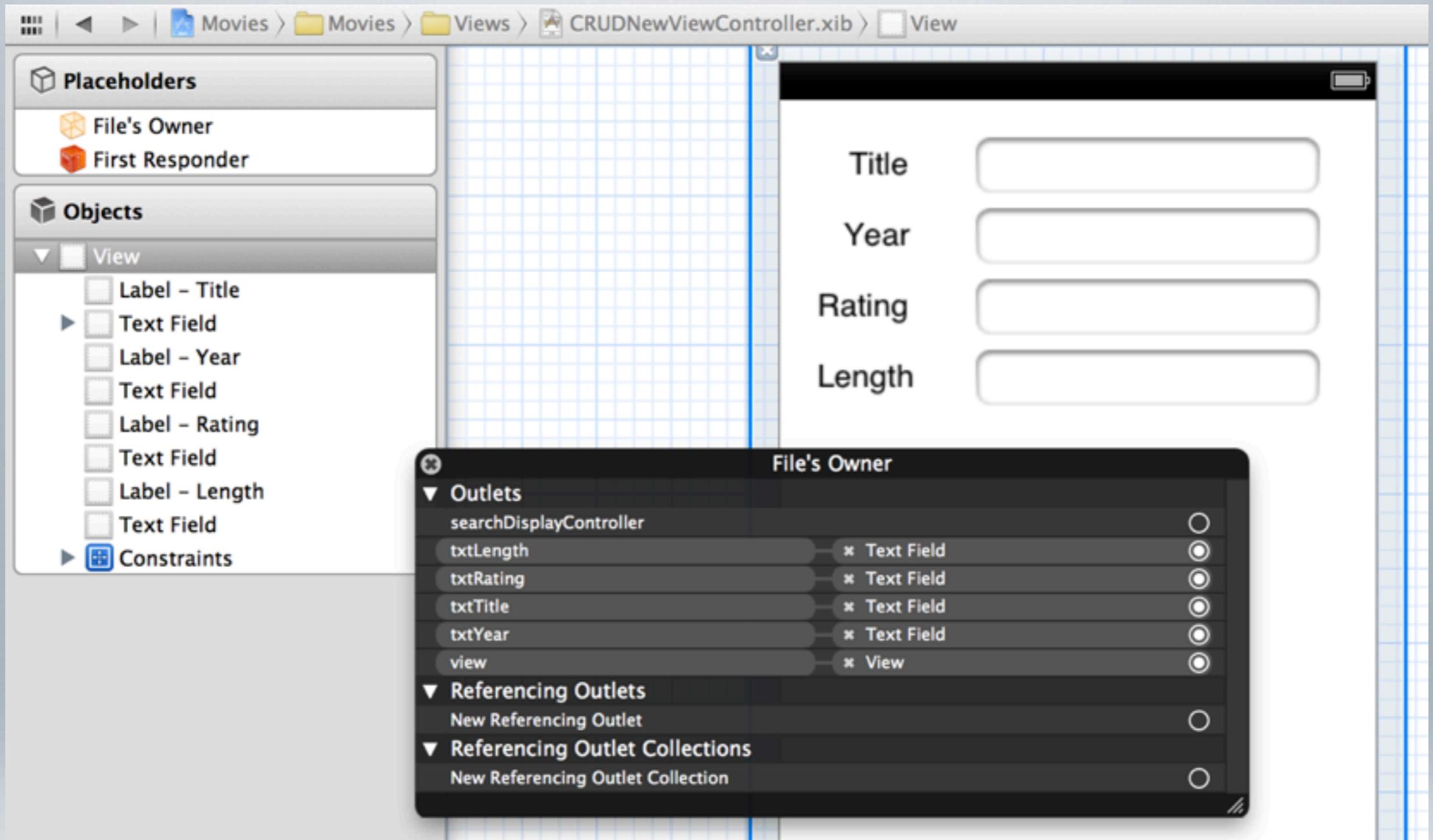
```

27 - (void)viewDidLoad
28 {
29     [super viewDidLoad];
30     // Do any additional setup after loading the view from its nib.
31     UIBarButtonItem *saveButton = [[UIBarButtonItem alloc]
32                                     initWithBarButtonSystemItem:UIBarButtonSystemItemSave target:self
33                                     action:@selector(saveMovie:)];
34     self.navigationItem.rightBarButtonItem = saveButton;
35 }
36 -(void)saveMovie:(id)sender
37 {
38     // Create an instance of DBServer class.
39     DatabaseController *myDB = [[DatabaseController alloc] init];
40
41     [myDB insertMovie:txtTitle.text
42                 Year:[txtYear.text integerValue]
43                 Rating:txtRating.text
44                 Length:txtLength.text];
45
46     // Release the dbAccess object to free its memory
47     myDB=nil;
48 }
49 }
```

ADD A SAVE BUTTON TO THE NEWVIEW PROGRAMMATICALLY

newViewDidLoad





MAKE SURE TO LINK UP OUR PROPERTIES TO THE
VIEW

```
162 - (void)insertNewObject:(id)sender
163 {
164     CRUDNewViewController *newCustomerView = [[CRUDNewViewController alloc]
165         initWithNibName:@"CRUDNewViewController" bundle:nil];
166
167     // Set the title of the detail page
168     [newCustomerView setTitle:@"New Movie"];
169     [self.navigationController pushViewController:newCustomerView animated:YES];
170
171     newCustomerView=nil;
172 }
```

UPDATE THE INSERTNEWOBJECT METHOD IN THE MASTERVIEWCONTROLLER TO CALL OUR NEWVIEW

masterinsertnewcrud



```

100 - (void)tableView:(UITableView *)tableView commitEditingStyle:
101   (UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath
102 {
103   if (editingStyle == UITableViewCellEditingStyleDelete) {
104     DatabaseController *dbController = [[DatabaseController alloc] init];
105     Movie *theMovie = [_arrayMovies objectAtIndex:[indexPath row]];
106     [dbController deleteMovie:theMovie.intMovieId];
107
108     [_arrayMovies removeObjectAtIndex:indexPath.row];
109     [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:
110      UITableViewRowAnimationFade];
111   } else if (editingStyle == UITableViewCellEditingStyleInsert) {
112     // Create a new instance of the appropriate class, insert it into the
113     // array, and add a new row to the table view.
114   }
115 }

```

RUN THE DELETE METHOD WHEN THE MOVIES ARE DELETED

masterdeletecrud



<https://github.com/GroundSpeed/CRUDdySQLite3>

Don Miller

don@groundspeedhq.com

@donmiller

Or follow @groundspeedhq
groundspeedhq.com



Don Miller



GroundSpeed™
rapid web + mobile software