Building CRUDdy iOS Apps with SQLite3

An Introduction to SQLite3 for your iOS Application

devLink

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 sqlite3_prepare_v2() 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 sqlite3_prepare_v2(), sqlite3_step() and sqlite3_finalize() 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
```

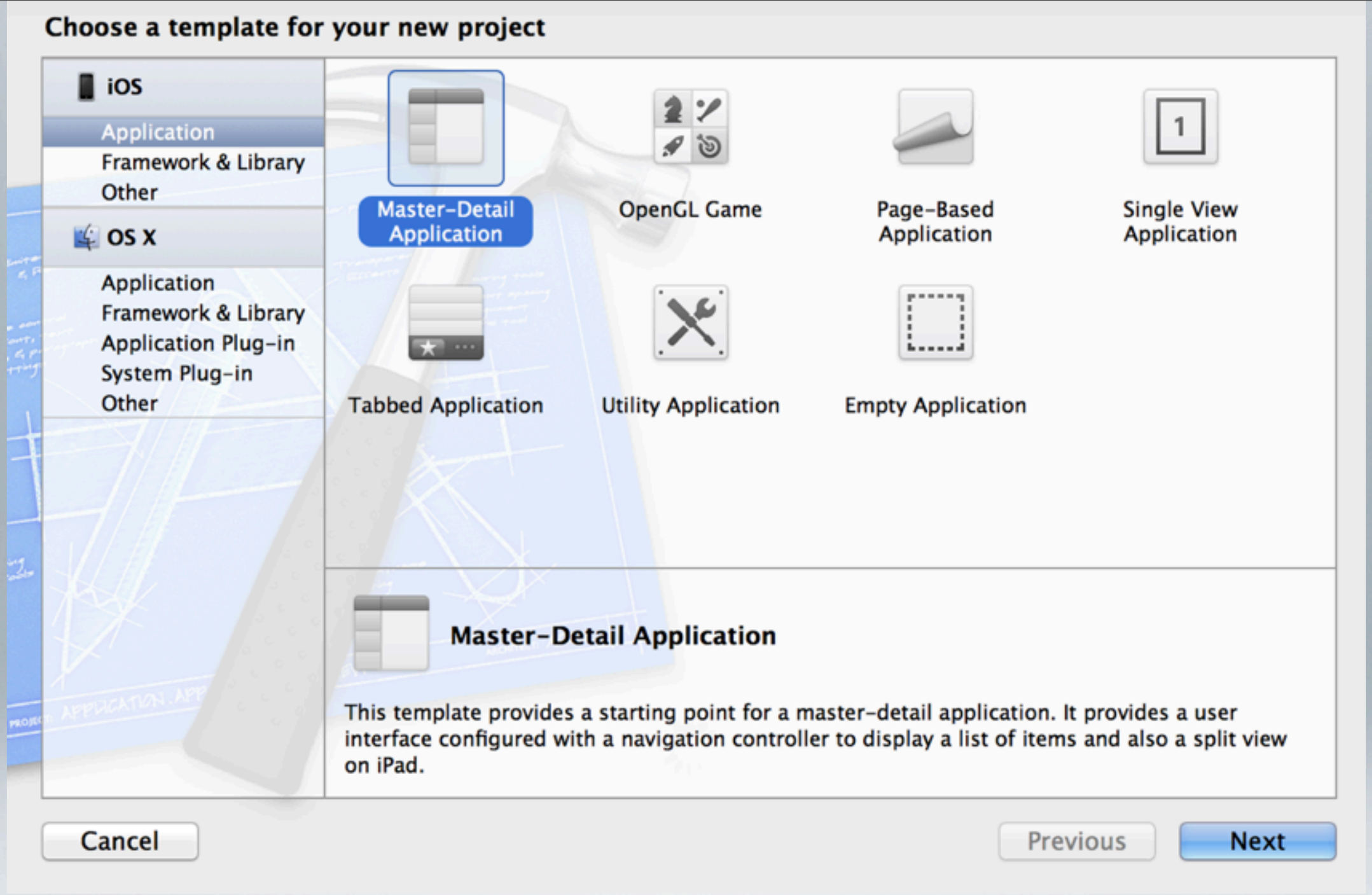
devLink

GroundSpeed
rapid web + mobile software

EXAMPLE APPLICATION

devLink

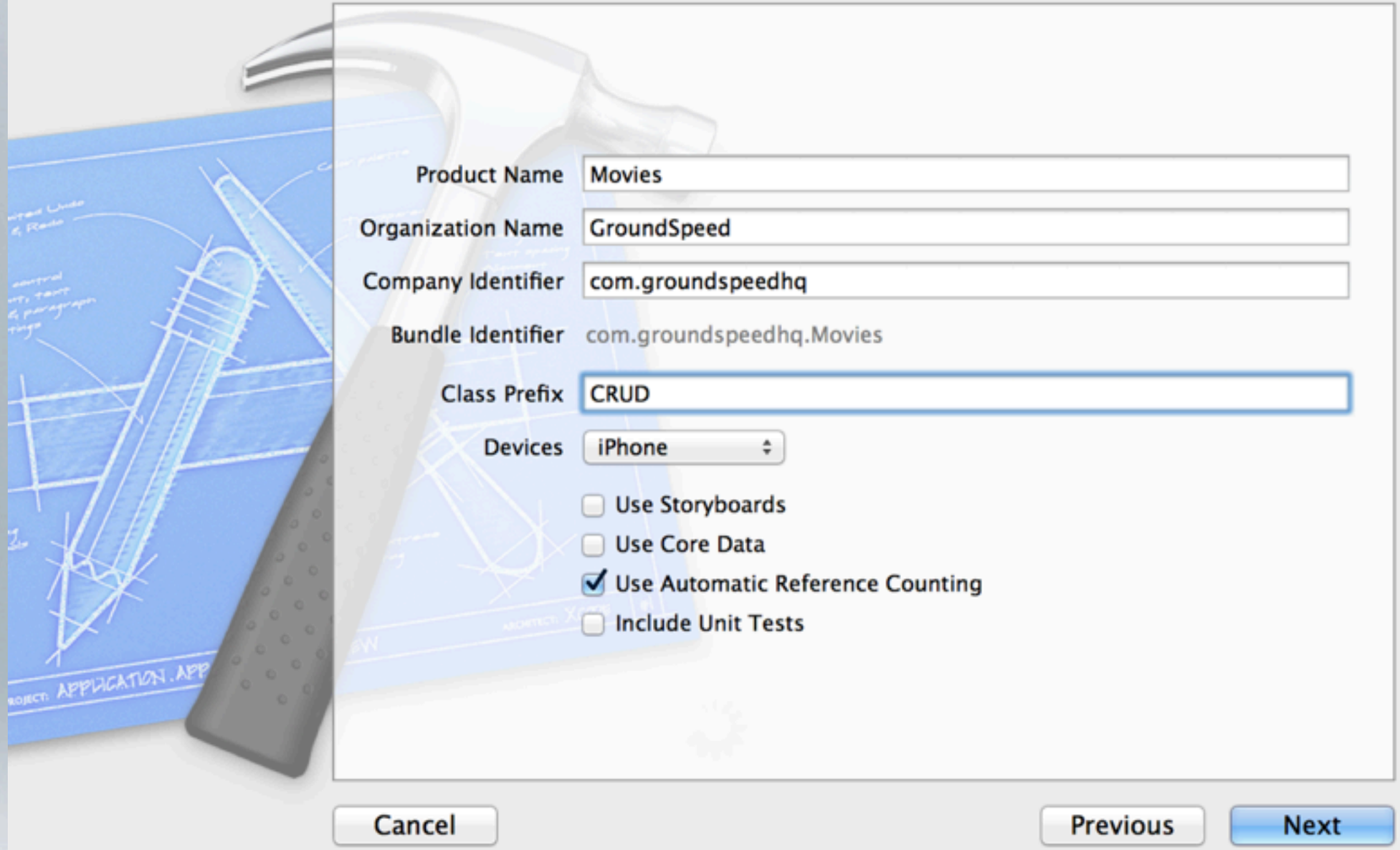
GroundSpeed



START A MASTER-DETAIL APP

devLink

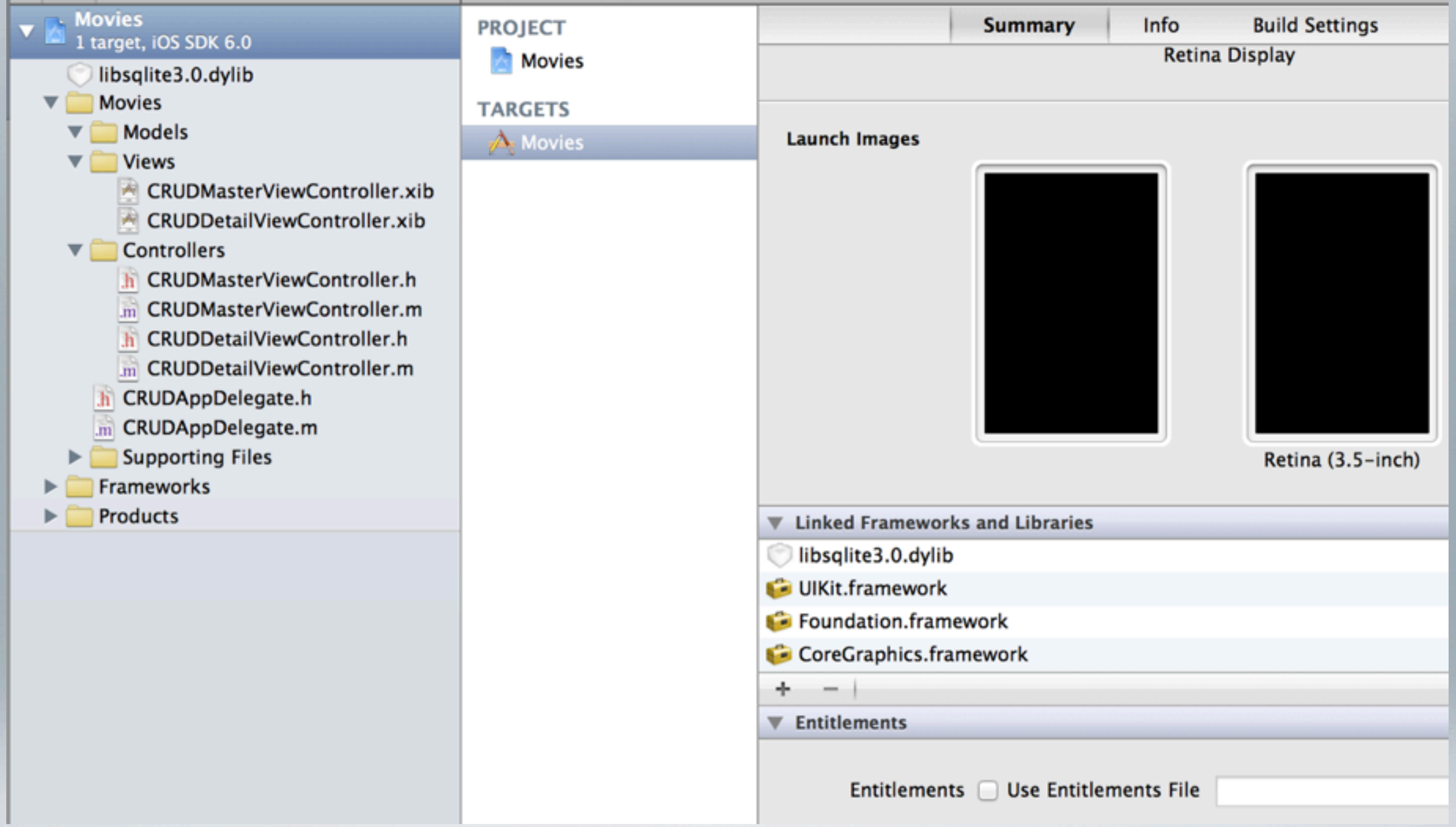
GroundSpeed



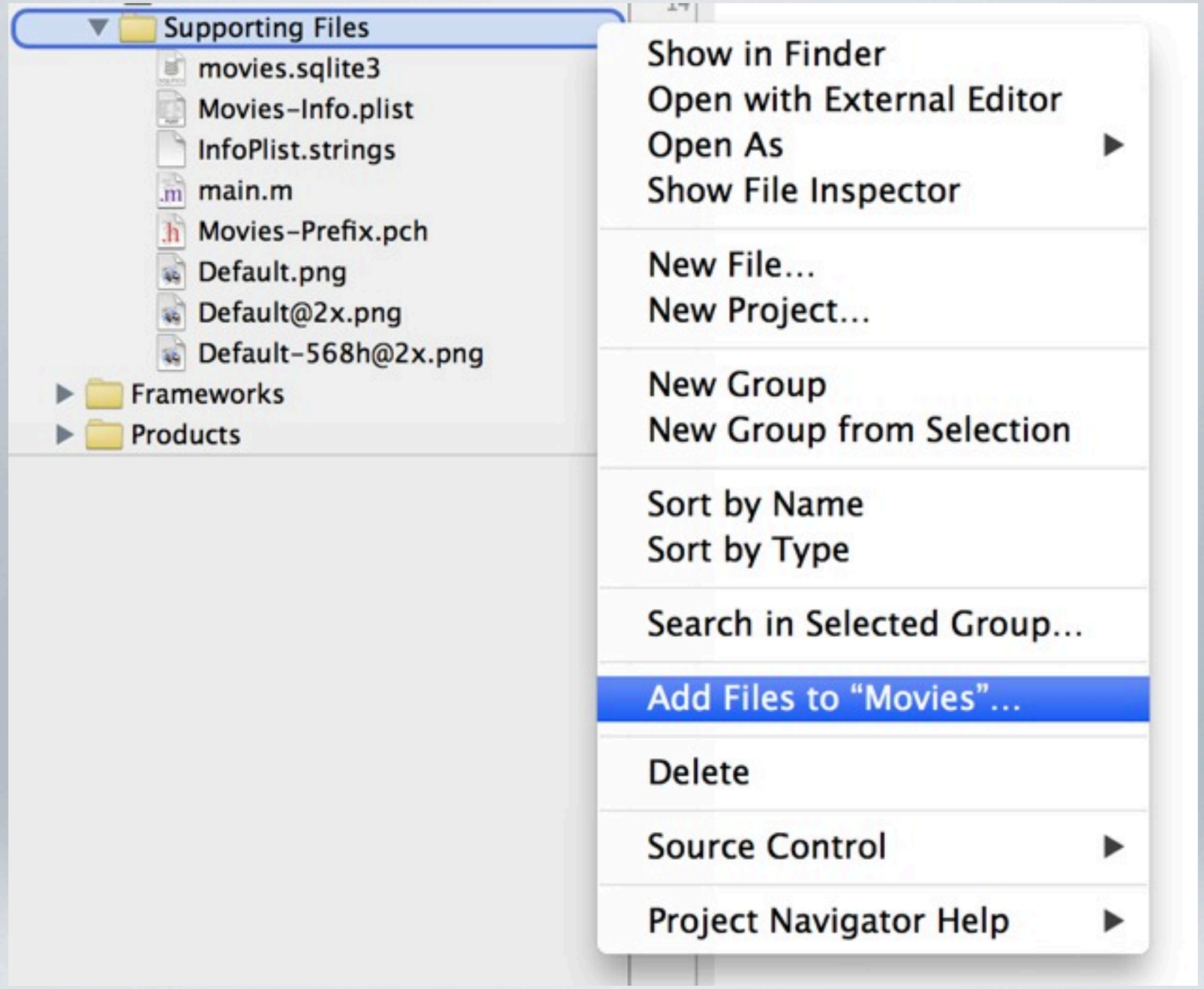
NAME YOUR PROJECT

devLink

GroundSpeed



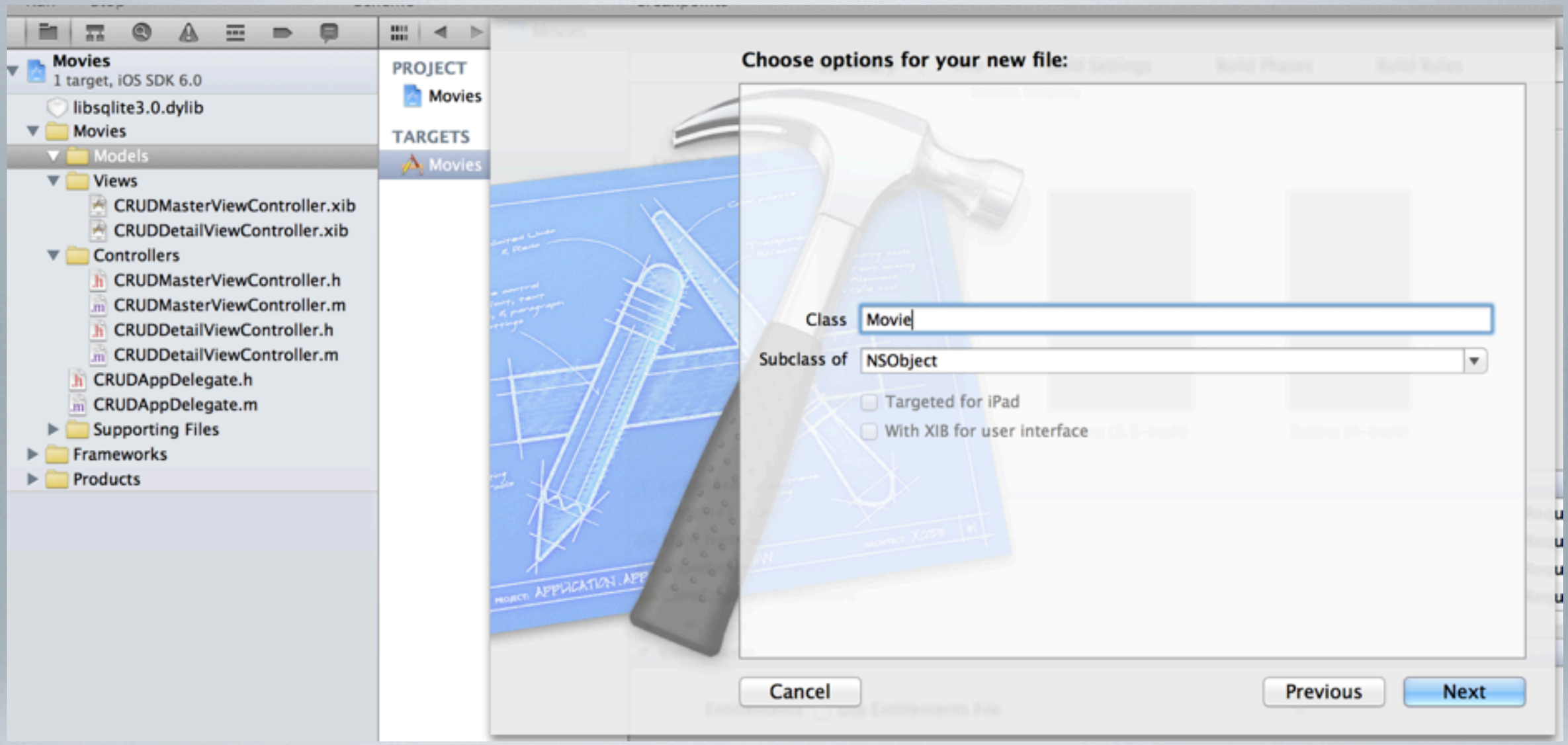
ADD SQLITE LIBRARY



ADD DATABASE TO PROJECT

devLink

GroundSpeed



CREATE MOVIES MODEL

devLink

GroundSpeed

```objc
#import <Foundation/Foundation.h>

@interface Movie : NSObject

@property(nonatomic) int       intMovieId;
@property(nonatomic, strong)  NSString *strTitle;
@property(nonatomic) int       intYear;
@property(nonatomic, strong)  NSString *strRating;
@property(nonatomic, strong)  NSString *strLength;

-(id)  initWithMovieId: (int) pMovieId
             WithTitle:(NSString *) pTitle
              WithYear: (int) pYear
            WithRating: (NSString *) pRating
            WithLength: (NSString *) pLength;

@end
```

GCODE MOVIE HEADER

movieheadercrud

devLink

GroundSpeed

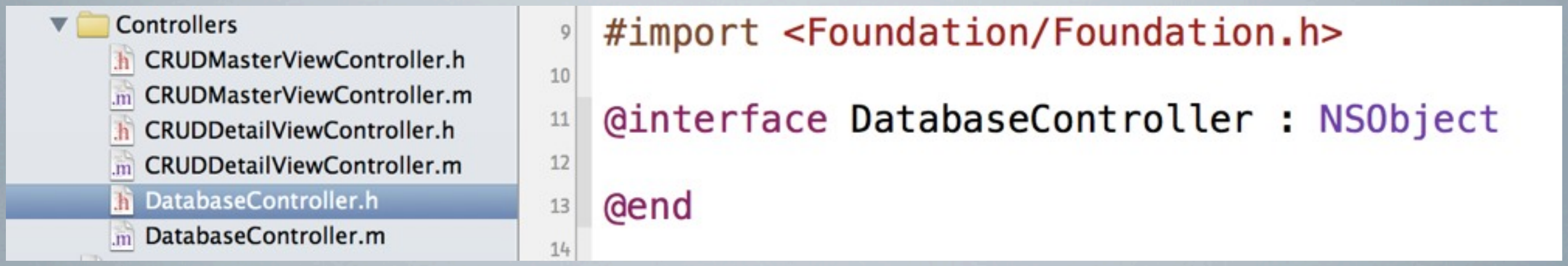
```objc
#import "Movie.h"

@implementation Movie

-(id)  initWithMovieId: (int) pMovieId
             WithTitle:(NSString *) pTitle
              WithYear: (int) pYear
            WithRating: (NSString *) pRating
            WithLength: (NSString *) pLength
{
    if ((self=[super init]))
    {
        self.intMovieId = pMovieId;
        self.strTitle = pTitle;
        self.intYear = pYear;
        self.strRating = pRating;
        self.strLength = pLength;
        return self;
    }
    else
    {
        return nil;
    }
}

@end
```

CODE MOVIE IMPLEMENTATION

movieimplementcrud

devLink

GroundSpeed



CREATE DATABASE CONTROLLER

devLink

GroundSpeed

```objc
#import <Foundation/Foundation.h>
#import <sqlite3.h>  // Observe this import
#import "Movie.h"    // Observe this import too

@interface DatabaseController : NSObject
{
    NSString *dbName;
}

@property (nonatomic,retain) NSFileManager *fileMgr;
@property (nonatomic,retain) NSString *homeDir;
@property (nonatomic,retain) NSString *title;

-(void) initializeDatabase;
-(void) closeDatabase;
-(void) CopyDbToDocumentsFolder;
-(NSString *) GetDocumentDirectory;

-(NSMutableArray*) getAllMovies;

-(void) insertMovie:(NSString *) fldTitle
               Year:(int) fldYear
             Rating:(NSString *) fldRating
             Length:(NSString *) fldLength;

-(void) updateMovie:(int) fldMovieId
              Title:(NSString *) fldTitle
               Year:(int) fldYear
             Rating:(NSString *) fldRating
             Length:(NSString *) fldLength;

-(void) deleteMovie:(int) fldMovieId;
@end
```

CREATE DB CONTROLLER HEADER

dbcontrollerheadercrud

```objc
sqlite3 *sqlDatabase;

-(id) init
{   //  Call super init to invoke superclass initiation code
    if ((self = [super init]))
    {   //  set the reference to the database
        dbName = @"movies.sqlite3";
        [self initializeDatabase];
    }
    return self;
}

- (void) initializeDatabase
{
    [self CopyDbToDocumentsFolder];

    //Get list of directories in Document path
    NSArray *dirPath = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES);

    //Define new path for database
    NSString *path = [[dirPath objectAtIndex:0] stringByAppendingPathComponent:dbName];

    // Get the database from the application bundle.
    NSLog(@"%@", path);
    // Open the database.
    if (sqlite3_open([path UTF8String], &sqlDatabase) == SQLITE_OK)
    {
        NSLog(@"Opening Database");
    }
    else
    {   // Call close to properly clean up
        sqlite3_close(sqlDatabase);
        NSAssert1(0, @"Failed to open database: '%s'.",
                  sqlite3_errmsg(sqlDatabase));
    }
}
```

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

IMPLEMENT INITIALIZER AND INITIALIZE

dbcontrollerinitcrud

```objc
-(void) CopyDbToDocumentsFolder
{
    NSError *err=nil;

    _fileMgr = [NSFileManager defaultManager];

    NSString *dbpath = [[[NSBundle mainBundle] resourcePath]
        stringByAppendingPathComponent:dbName];

    NSString *copydbpath = [self.GetDocumentDirectory
        stringByAppendingPathComponent:dbName];

    // lets only copy the db if it does not exist
    if (![_fileMgr fileExistsAtPath:copydbpath])
    {
        if(![_fileMgr copyItemAtPath:dbpath toPath:copydbpath error:&err])
        {
            UIAlertView *tellErr = [[UIAlertView alloc] initWithTitle:_title
                message:@"Unable to copy database." delegate:self
                cancelButtonTitle:@"OK" otherButtonTitles:nil];
            [tellErr show];
        }
    }
}

-(NSString *) GetDocumentDirectory
{
    _fileMgr = [NSFileManager defaultManager];
    _homeDir = [NSHomeDirectory()
        stringByAppendingPathComponent:@"Documents/"];

    return _homeDir;
}
```

COPY DB TO DOCUMENTS FOLDER

dbcontrollercopydbcrud

```objc
-(void) closeDatabase
{
    // Close the database.
    if (sqlite3_close(sqlDatabase) != SQLITE_OK) {
        NSAssert1(0, @"Error: failed to close database: '%s'.",
                  sqlite3_errmsg(sqlDatabase));
    }
}
```

CODE THE CLOSE DATABASE METHOD

dbcontrollerclosedbcrud

devLink

GroundSpeed

```objc
-(NSMutableArray*) getAllMovies
{
    NSMutableArray *arrayMovies = [[NSMutableArray alloc] init];

    const char *mySql = "SELECT id, title, year, rating, length FROM movies";
    const int movieId = 0;
    const int movieTitle = 1;
    const int movieYear = 2;
    const int movieRating = 3;
    const int movieLength = 4;

    //  Declare the SQLite statement object that will hold our result set
    sqlite3_stmt *myStatement;

    int sqlResult = sqlite3_prepare_v2(sqlDatabase, mySql, -1, &myStatement, NULL);
    if ( sqlResult== SQLITE_OK)
    {
        while (sqlite3_step(myStatement) == SQLITE_ROW)
        {
            char *name = (char *)sqlite3_column_text(myStatement, movieTitle);
            char *rating = (char *)sqlite3_column_text(myStatement, movieRating);
            char *length = (char *)sqlite3_column_text(myStatement, movieLength);

            Movie *movies = [[Movie alloc]
                 initWithMovieId:sqlite3_column_int(myStatement, movieId)
                       WithTitle:(name) ? [NSString stringWithUTF8String:name] : @""
                        WithYear:sqlite3_column_double(myStatement, movieYear)
                      WithRating:(rating) ? [NSString stringWithUTF8String:rating] : @""
                      WithLength:(length) ? [NSString stringWithUTF8String:length] : @""];
            [arrayMovies addObject:movies];
            movies = nil;
        }
        sqlite3_finalize(myStatement);
    }
    else {
        NSLog(@"Problem with the database:");
        NSLog(@"%d",sqlResult);
    }
    return arrayMovies;
}
```

PUT ALL THE MOVIES INTO AN ARRAY

dbcontrollergetmoviescrud

```objc
-(void) insertMovie:(NSString *) fldTitle
               Year:(int) fldYear
             Rating:(NSString *) fldRating
             Length:(NSString *) fldLength
{
    NSString *documentPath = [self.GetDocumentDirectory stringByAppendingPathComponent:dbName];

    if(!(sqlite3_open([documentPath UTF8String], &sqlDatabase) == SQLITE_OK))
    {
        NSLog(@"An error has occured.");
        return;
    }
    else
    {
        NSString *insertSQL = [NSString stringWithFormat:
                               @"INSERT INTO Movie(title, year, rating, length) VALUES ('%@','%i',
                                   %@, '%@')", fldTitle, fldYear, fldRating, fldLength];

        const char *sql = [insertSQL UTF8String];

        sqlite3_stmt *sqlStatement;
        if(sqlite3_prepare_v2(sqlDatabase, sql, -1, &sqlStatement, NULL) != SQLITE_OK)
        {
            NSLog(@"Problem with prepare statement");
            return;
        }
        else
        {
            if(sqlite3_step(sqlStatement)==SQLITE_DONE)
            {
                sqlite3_finalize(sqlStatement);
                sqlite3_close(sqlDatabase);
            }
        }
    }
}
```

CREATE METHOD TO INSERT MOVIE

dbcontrollerinsertmoviecrud

```objc
-(void) updateMovie:(int) fldMovieId
              Title:(NSString *) fldTitle
               Year:(int) fldYear
             Rating:(NSString *) fldRating
             Length:(NSString *) fldLength
{
    NSString *documentPath = [self.GetDocumentDirectory stringByAppendingPathComponent:dbName];

    if(!(sqlite3_open([documentPath UTF8String], &sqlDatabase) == SQLITE_OK))
    {
        NSLog(@"An error has occured.");
        return;
    }
    else
    {
        NSString *updateSQL = [NSString stringWithFormat:
                               @"UPDATE movies SET Title='%@', year='%i', rating=%@, length='%@'
                                   WHERE id=%i", fldTitle, fldYear, fldRating, fldLength,
                                   fldMovieId];

        const char *sql = [updateSQL UTF8String];

        sqlite3_stmt *sqlStatement;
        if(sqlite3_prepare_v2(sqlDatabase, sql, -1, &sqlStatement, NULL) != SQLITE_OK)
        {
            NSLog(@"Problem with prepare statement");
            return;
        }
        else
        {
            if(sqlite3_step(sqlStatement)==SQLITE_DONE)
            {
                sqlite3_finalize(sqlStatement);
                sqlite3_close(sqlDatabase);
            }
        }
    }
}
```

CREATE METHOD TO UPDATE MOVIE

dbcontrollerupdatemoviecrud

```objc
-(void) deleteMovie:(int) fldMovieId
{
    NSString *documentPath = [self.GetDocumentDirectory stringByAppendingPathComponent:dbName];

    if(!(sqlite3_open([documentPath UTF8String], &sqlDatabase) == SQLITE_OK))
    {
        NSLog(@"An error has occured.");
        return;
    }
    else
    {
        NSString *deleteSQL = [NSString stringWithFormat:@"DELETE FROM movies WHERE id = '%i'", fldMovieId];
        const char *sql = [deleteSQL UTF8String];

        sqlite3_stmt *sqlStatement;
        if(sqlite3_prepare_v2(sqlDatabase, sql, -1, &sqlStatement, NULL) != SQLITE_OK)
        {
            NSLog(@"Problem with prepare statement");
            return;
        }
        else
        {
            if(sqlite3_step(sqlStatement)==SQLITE_DONE)
            {
                sqlite3_finalize(sqlStatement);
                sqlite3_close(sqlDatabase);
            }
        }
    }
}
```

CREATE FINAL METHOD TO DELETE MOVIE

dbcontrollerdeletemoviecrud

devLink

GroundSpeed

```objc
#import <UIKit/UIKit.h>
#import "Movie.h"
#import "DatabaseController.h"

@class CRUDDetailViewController;

@interface CRUDMasterViewController : UITableViewController

@property (strong, nonatomic) CRUDDetailViewController *detailViewController;
@property (strong, nonatomic) NSMutableArray *arrayMovies;

-(void)loadMovieObject;

@end
```

UPDATE MASTERVIEW CONTROLLER HEADER

masterviewheadercrud

devLink

GroundSpeed

```objc
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        self.title = NSLocalizedString(@"Movies", @"Movies");
    }
    return self;
}
```

ADD THE TITLE TO THE NAVIGATION

devLink

GroundSpeed

```objc
-(void)viewDidAppear:(BOOL)animated
{
    [self loadMovieObject];
    [self.tableView reloadData];
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    self.navigationItem.leftBarButtonItem = self.editButtonItem;

    [self loadMovieObject];

    UIBarButtonItem *addButton = [[UIBarButtonItem alloc] initWithBarButtonSystemItem:
        UIBarButtonSystemItemAdd target:self action:@selector(insertNewObject:)];
    self.navigationItem.rightBarButtonItem = addButton;
}
```

ADD VIEWDIDAPPEAR AND UPDATE VIEWDIDLOAD

masterviewappearcrud

devLink

GroundSpeed

```
135 #pragma mark User Methods
136
137 -(void) loadMovieObject
138 {
139     DatabaseController *dbController = [[DatabaseController alloc] init];
140
141     //  Fire the dbController getAllCustomers method to fill our array
142     _arrayMovies = [dbController getAllMovies];
143
144     //  Release the dbAccess object to free its memory
145     dbController=nil;
146 }
```

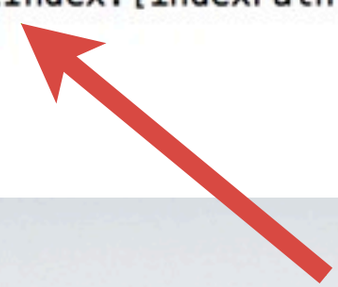
CODE THE LOADMOVIEOBJECT METHOD

masterviewloadmoviescrud

devLink

GroundSpeed

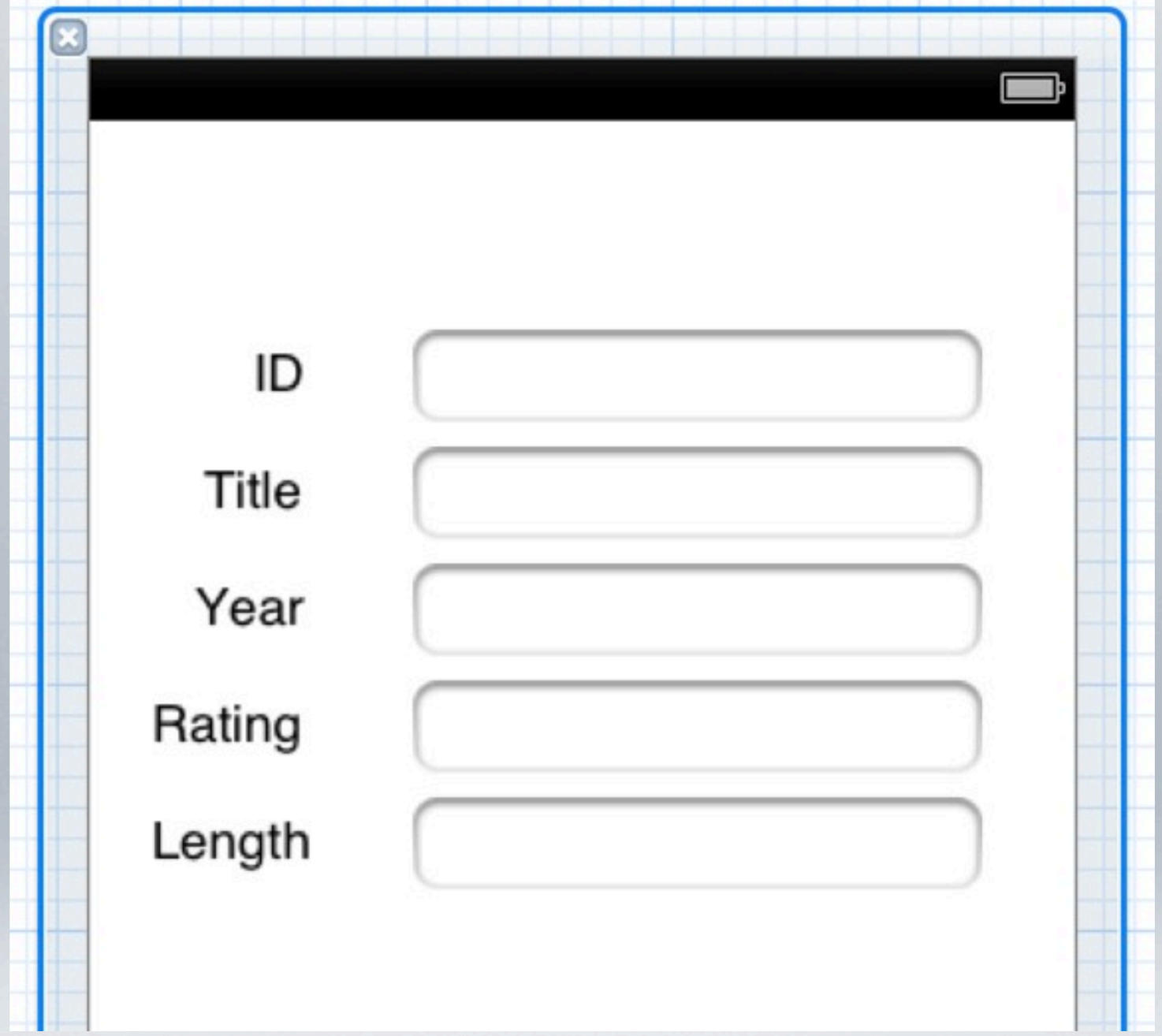
```objc
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return _arrayMovies.count;
}

// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier];
        cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;
    }

    Movie *movies = [_arrayMovies objectAtIndex:[indexPath row]];
    cell.textLabel.text = movies.strTitle;

    return cell;
}
```



MODIFY TABLEVIEW METHODS TO RETURN MOVIES

devLink

GroundSpeed



MODIFY THE DETAIL VIEW TO HANDLE FIELDS

```objc
#import <UIKit/UIKit.h>
#import "Movie.h"
#import "DatabaseController.h"

@interface CRUDDetailViewController : UIViewController

@property (strong, nonatomic) id detailItem;
@property (weak, nonatomic) IBOutlet UILabel *detailDescriptionLabel;

@property (nonatomic, strong) IBOutlet UITextField *txtMovieID;
@property (nonatomic, strong) IBOutlet UITextField *txtTitle;
@property (nonatomic, strong) IBOutlet UITextField *txtYear;
@property (nonatomic, strong) IBOutlet UITextField *txtRating;
@property (nonatomic, strong) IBOutlet UITextField *txtLength;

-(void)setLabelsForMovie: (Movie *) theMovie;
-(void)updateMovie:(id)sender;
```

ADD PROPERTIES TO THE DETAIL VIEW CONTROLLER

detailviewpropertiescrud

devLink

GroundSpeed

```
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]
           initWithBarButtonSystemItem:UIBarButtonSystemItemSave target:self
           action:@selector(updateMovie:)];
46     self.navigationItem.rightBarButtonItem = saveButton;
47 }
48
```

ADD SAVE BUTTON TO TOP RIGHT OF DETAILVIEW

saveButton

```objc
#pragma mark User Methods

-(void) setLabelsForMovie:(Movie *)theMovie
{
    //  Set the labels to the values passed of the passed customer object
    _txtMovieID.text=[NSString stringWithFormat:@"%i", theMovie.intMovieId];
    _txtTitle.text=theMovie.strTitle;
    _txtYear.text=[NSString stringWithFormat:@"%i", theMovie.intYear];
    _txtRating.text=theMovie.strRating;
    _txtLength.text=theMovie.strLength;
}

-(void) updateMovie:(id)sender
{
    //  Create an instance of DBServer class.
    DatabaseController *dbController = [[DatabaseController alloc] init];

    [dbController updateMovie:[_txtMovieID.text intValue]
                        Title:_txtTitle.text
                         Year:[_txtYear.text intValue]
                       Rating:_txtRating.text
                       Length:_txtLength.text];

    //  Release the dbAccess object to free its memory
    dbController=nil;
}
```

CREATE LABEL AND UPDATE METHODS FOR DETAIL

detailviewusermethodscrud

devLink

GroundSpeed

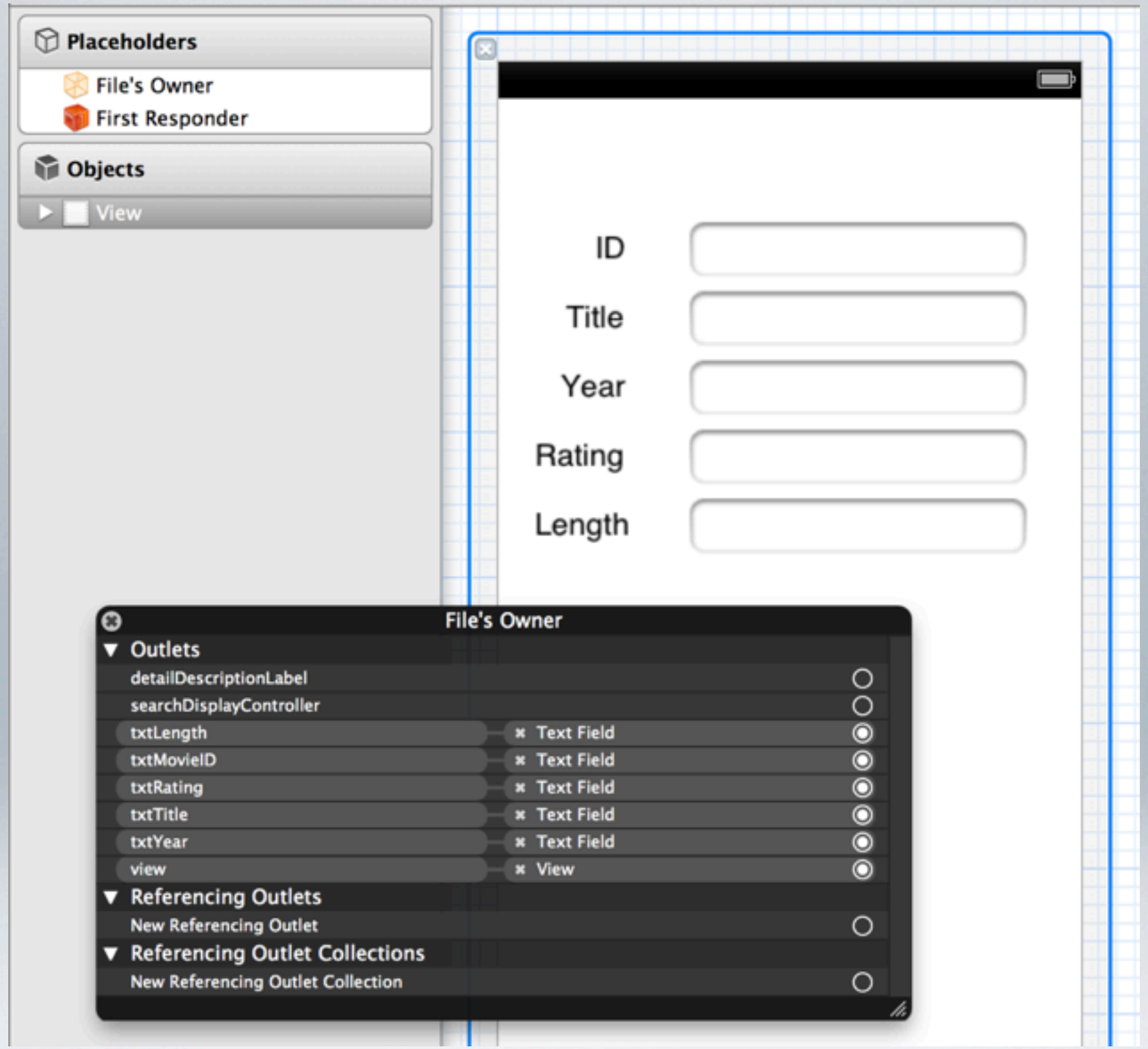
```
126 - (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
127 {
128     if (!self.detailViewController) {
129         self.detailViewController = [[CRUDDetailViewController alloc] initWithNibName:@"CRUDDetailViewController" bundle:nil]
                    ;
130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

UPDATE DIDSELECTROW METHOD ON MASTERVIEW

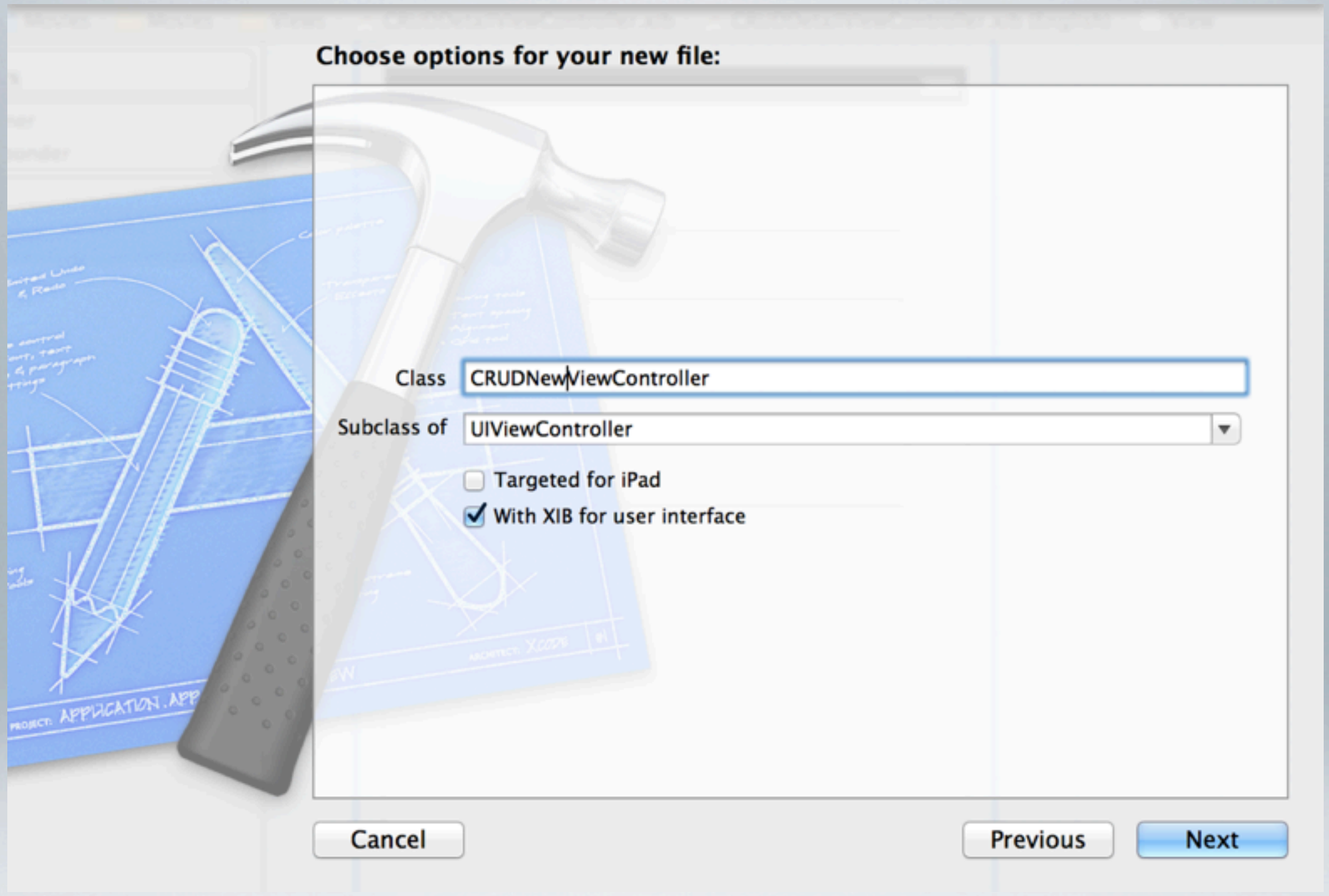
masterviewdidselectcrud

devLink

GroundSpeed



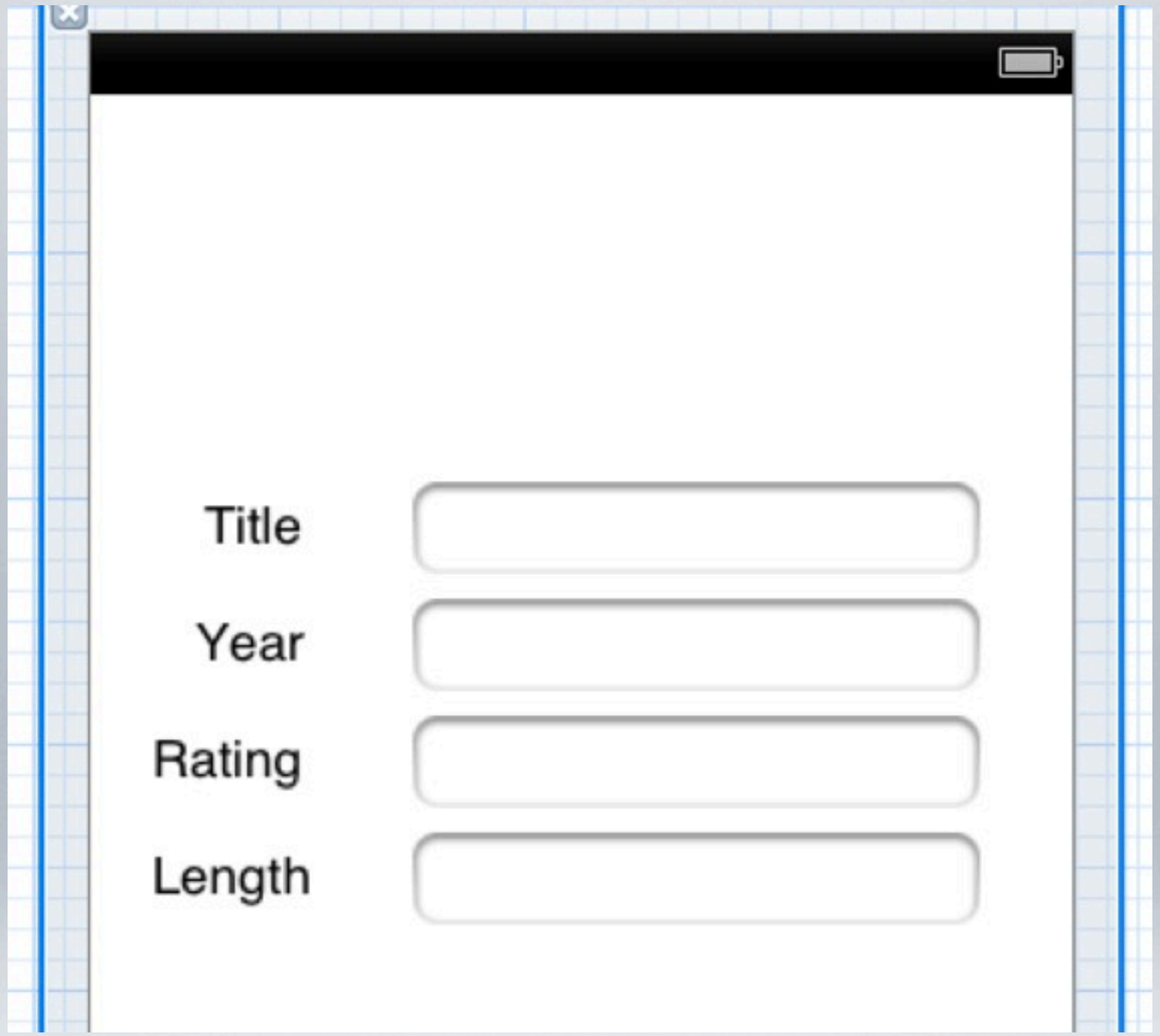
LINK YOUR PROPERTIES TO YOUR DETAIL IB LABELS



CREATE A NEW MOVIE VIEW CONTROLLER

devLink

GroundSpeed



CREATE LABELS AND TEXT BOXES FOR NEW FORM

devLink

GroundSpeed

```objc
#import <UIKit/UIKit.h>
#import "DatabaseController.h"

@interface CRUDNewViewController : UIViewController

@property (nonatomic, strong) IBOutlet UITextField *txtTitle;
@property (nonatomic, strong) IBOutlet UITextField *txtYear;
@property (nonatomic, strong) IBOutlet UITextField *txtRating;
@property (nonatomic, strong) IBOutlet UITextField *txtLength;

-(void)saveMovie:(id)sender;

@end
```

CREATE PROPERTIES IN THE NEW HEADER CLASS

devLink

GroundSpeed

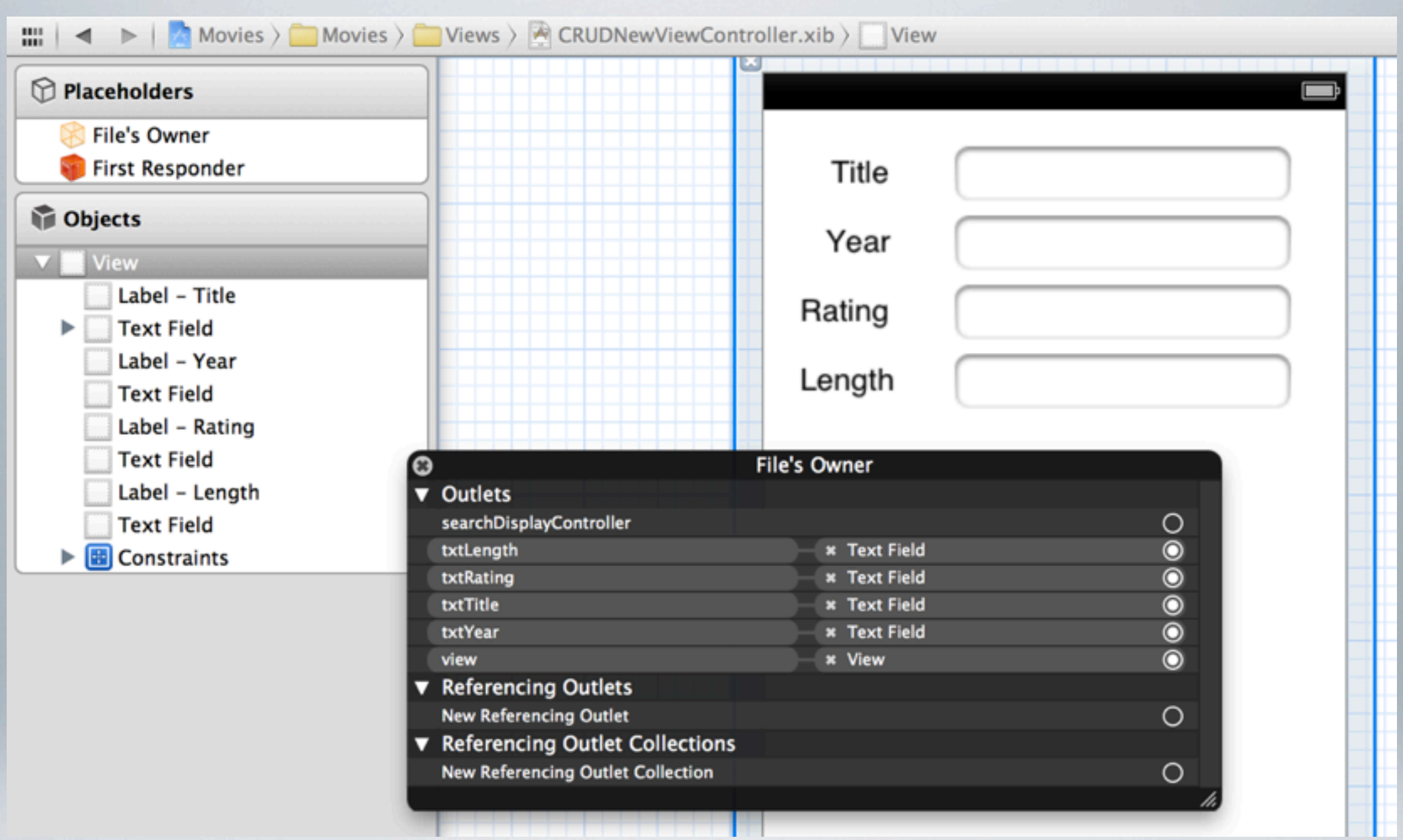
```objc
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view from its nib.
    UIBarButtonItem *saveButton = [[UIBarButtonItem alloc]
        initWithBarButtonSystemItem:UIBarButtonSystemItemSave target:self
        action:@selector(saveMovie:)];
    self.navigationItem.rightBarButtonItem = saveButton;
}

-(void)saveMovie:(id)sender
{
    //  Create an instance of DBServer class.
    DatabaseController *myDB = [[DatabaseController alloc] init];

    [myDB insertMovie:txtTitle.text
                 Year:[txtYear.text integerValue]
               Rating:txtRating.text
               Length:txtLength.text];

    //  Release the dbAccess object to free its memory
    myDB=nil;

}
```

ADD A SAVE BUTTON TO THE NEW VIEW PROGRAMMATICALLY

newViewDidLoad

devLink

GroundSpeed



MAKE SURE TO LINK UP OUR PROPERTIES TO THE VIEW

```objc
- (void)insertNewObject:(id)sender
{
    CRUDNewViewController *newCustomerView = [[CRUDNewViewController alloc]
        initWithNibName:@"CRUDNewViewController" bundle:nil];

    //  Set the title of the detail page
    [newCustomerView setTitle:@"New Movie"];
    [self.navigationController pushViewController:newCustomerView animated:YES];

    newCustomerView=nil;
}
```

UPDATE THE INSERTNEWOBJECT METHOD IN THE MASTERVIEWCONTROLLER TO CALL OUR NEW VIEW

masterinsertnewcrud

devLink

GroundSpeed

```objc
- (void)tableView:(UITableView *)tableView commitEditingStyle:
    (UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)
    indexPath
{
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        DatabaseController *dbController = [[DatabaseController alloc] init];
        Movie *theMovie = [_arrayMovies objectAtIndex:[indexPath row]];
        [dbController deleteMovie:theMovie.intMovieId];

        [_arrayMovies removeObjectAtIndex:indexPath.row];
        [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:
            UITableViewRowAnimationFade];
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
        // Create a new instance of the appropriate class, insert it into the
            array, and add a new row to the table view.
    }
}
```

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