**How to Save Photos to Core Data**

**Part 1: THE SETUP**

**Create a project using the Empty Application Template**

**Call it anything you’d like and check Core Data**

**Select your core date file (xcdatamodeld)**

**Click Add Entity at the bottom of the screen and call the new entity Pictures.**

**With it selected add a new attribute called title and make it of type String.**

**Now add another entity called desc and make it of type String.**

**Add a third attribute and call it smallPicture (note the case) and make it of type Binary Data. That’s all.**

**Choose New File.**

**On the left hand side choose Core Data and on the right choose NSManagedObject subclass.**

**After you click Create, Xcode creates a class for each of the entities we created**

**Choose New File.**

**On the left hand side, choose Cocoa Touch and on the right, choose Objective-C class.**

**Call this class CoreDataHelper and make it a subclass of NSObject.**

**CoreDataHelper.h should have: (note all the class methods +)**

#import <UIKit/UIKit.h>

#import <CoreData/CoreData.h>

@interface CoreDataHelper : NSObject

**// For retrieval of objects**

+(NSMutableArray \*)getObjectsForEntity:(NSString\*)entityName withSortKey:(NSString\*)sortKey andSortAscending:(BOOL)sortAscending andContext:(NSManagedObjectContext \*)managedObjectContext;

+(NSMutableArray \*)searchObjectsForEntity:(NSString\*)entityName withPredicate:(NSPredicate \*)predicate andSortKey:(NSString\*)sortKey andSortAscending:(BOOL)sortAscending andContext:(NSManagedObjectContext \*)managedObjectContext;

**// For deletion of objects**

+(BOOL)deleteAllObjectsForEntity:(NSString\*)entityName withPredicate:(NSPredicate\*)predicate andContext:(NSManagedObjectContext \*)managedObjectContext;

+(BOOL)deleteAllObjectsForEntity:(NSString\*)entityName andContext:(NSManagedObjectContext \*)managedObjectContext;

**// For counting objects**

+(NSUInteger)countForEntity:(NSString \*)entityName andContext:(NSManagedObjectContext \*)managedObjectContext;

+(NSUInteger)countForEntity:(NSString \*)entityName withPredicate:(NSPredicate \*)predicate andContext:(NSManagedObjectContext \*)managedObjectContext;

@end

**CoreDataHelper.m should have:**

#import "CoreDataHelper.h"

@implementation CoreDataHelper

#pragma mark - Retrieve objects

**// Fetch objects with a predicate**

+(NSMutableArray \*)searchObjectsForEntity:(NSString\*)entityName withPredicate:(NSPredicate \*)predicate andSortKey:(NSString\*)sortKey andSortAscending:(BOOL)sortAscending andContext:(NSManagedObjectContext \*)managedObjectContext

{

**// Create fetch request**

NSFetchRequest \*request = [[NSFetchRequest alloc] init];

NSEntityDescription \*entity = [NSEntityDescription entityForName:entityName inManagedObjectContext:managedObjectContext];

[request setEntity:entity];

**// If a predicate was specified then use it in the request**

if (predicate != nil)

[request setPredicate:predicate];

**// If a sort key was passed then use it in the request**

if (sortKey != nil) {

NSSortDescriptor \*sortDescriptor = [[NSSortDescriptor alloc] initWithKey:sortKey ascending:sortAscending];

NSArray \*sortDescriptors = [[NSArray alloc] initWithObjects:sortDescriptor, nil];

[request setSortDescriptors:sortDescriptors];

}

**// Execute the fetch request**

NSError \*error = nil;

NSMutableArray \*mutableFetchResults = [[managedObjectContext executeFetchRequest:request error:&error] mutableCopy];

**// If the returned array was nil then there was an error**

if (mutableFetchResults == nil)

NSLog(@"Couldn't get objects for entity %@", entityName);

**// Return the results**

return mutableFetchResults;

}

**// Fetch objects without a predicate**

+(NSMutableArray \*)getObjectsForEntity:(NSString\*)entityName withSortKey:(NSString\*)sortKey andSortAscending:(BOOL)sortAscending andContext:(NSManagedObjectContext \*)managedObjectContext

{

return [self searchObjectsForEntity:entityName withPredicate:nil andSortKey:sortKey andSortAscending:sortAscending andContext:managedObjectContext];

}

**//this is mostly for the jump bar**

#pragma mark - Count objects

**// Get a count for an entity with a predicate**

+(NSUInteger)countForEntity:(NSString \*)entityName withPredicate:(NSPredicate \*)predicate andContext:(NSManagedObjectContext \*)managedObjectContext

{

**// Create fetch request**

NSFetchRequest \*request = [[NSFetchRequest alloc] init];

NSEntityDescription \*entity = [NSEntityDescription entityForName:entityName inManagedObjectContext:managedObjectContext];

[request setEntity:entity];

[request setIncludesPropertyValues:NO];

**// If a predicate was specified then use it in the request**

if (predicate != nil)

[request setPredicate:predicate];

**// Execute the count request**

NSError \*error = nil;

NSUInteger count = [managedObjectContext countForFetchRequest:request error:&error];

**// If the count returned NSNotFound there was an error**

if (count == NSNotFound)

NSLog(@"Couldn't get count for entity %@", entityName);

**// Return the results**

return count;

}

**// Get a count for an entity without a predicate**

+(NSUInteger)countForEntity:(NSString \*)entityName andContext:(NSManagedObjectContext \*)managedObjectContext

{

return [self countForEntity:entityName withPredicate:nil andContext:managedObjectContext];

}

#pragma mark - Delete Objects

**// Delete all objects for a given entity**

+(BOOL)deleteAllObjectsForEntity:(NSString\*)entityName withPredicate:(NSPredicate\*)predicate andContext:(NSManagedObjectContext \*)managedObjectContext

{

**// Create fetch request**

NSFetchRequest \*request = [[NSFetchRequest alloc] init];

NSEntityDescription \*entity = [NSEntityDescription entityForName:entityName inManagedObjectContext:managedObjectContext];

[request setEntity:entity];

**// Ignore property values for maximum performance**

[request setIncludesPropertyValues:NO];

**// If a predicate was specified then use it in the request**

if (predicate != nil)

[request setPredicate:predicate];

**// Execute the count request**

NSError \*error = nil;

NSArray \*fetchResults = [managedObjectContext executeFetchRequest:request error:&error];

**// Delete the objects returned if the results weren't nil**

if (fetchResults != nil) {

for (NSManagedObject \*manObj in fetchResults) {

[managedObjectContext deleteObject:manObj];

}

} else {

NSLog(@"Couldn't delete objects for entity %@", entityName);

return NO;

}

return YES;

}

+(BOOL)deleteAllObjectsForEntity:(NSString\*)entityName andContext:(NSManagedObjectContext \*)managedObjectContext

{

return [self deleteAllObjectsForEntity:entityName withPredicate:nil andContext:managedObjectContext];

}

@end

**PART 2. CREATING THE STORYBOARD**

**Right-click on your Supporting Files group and choose New File.**

**From the left hand side, choose User Interface and from the right, choose Storyboard.**

**Make sure the device family is set to iPhone.**

**Call it PictureListStoryboard.**

**Click on your project icon at the top of your files list.**

**On the right hand side, from the Main Storyboard drop down box choose PictureListStoryboard.**

**Open up AppDelegate.m**

**Add an** #import “PictureListMainTable.h”

**Go the didFinishLaunchingWithOptions: method:**

UINavigationController \*navigationController = (UINavigationController \*)self.window.rootViewController;

PictureListMainTable \*controller = (PictureListMainTable \*)navigationController.topViewController;

controller.managedObjectContext = self.managedObjectContext;

return YES;

**PART 3: THE INITIAL TABLE VIEW**

**Drag in a Table View Controller**

**Embed this in a Navigation Controller (Editor menu → Embed In → Navigation Controller)**

**Drag a Bar Button Item to the nav bar in your Table View and Change the Identifier in the Attributes Inspector to Add (+)**

**Choose New File. On the left, choose Cocoa Touch and on the right choose UIViewController subclass.**

**Call this new class PictureListMainTable and make it a subclass of UITableViewController.**

**PictureListMainTable.h should have:**

#import <UIKit/UIKit.h>

@interface PictureListMainTable : UITableViewController

@property (strong, nonatomic) NSManagedObjectContext \*managedObjectContext;

@property (strong, nonatomic) NSMutableArray \*pictureListData;

- (void)readDataForTable;

@end

**PictureListMainTable.m should have:**

#import "PictureListMainTable.h"

#import "CoreDataHelper.h"

#import "Pictures.h"

@implementation PictureListMainTable

@synthesize managedObjectContext, pictureListData;

**// When the view reappears, read new data for table**

- (void)viewWillAppear:(BOOL)animated

{

**// Repopulate the array with new table data**

[self readDataForTable];

}

**// Grab data for table - this will be used whenever the list appears or reappears after an add/edit**

- (void)readDataForTable

{

**// Grab the data**

pictureListData = [CoreDataHelper getObjectsForEntity:@"Pictures" withSortKey:@"title" andSortAscending:YES andContext:managedObjectContext];

**// Force table refresh**

[self.tableView reloadData];

}

#pragma mark - Table view data source

**// Return the number of sections in the table**

- (NSInteger)numberOfSectionsInTableView:(UITableView \*)tableView

{

return 1;

}

**// Return the number of rows in the section (the amount of items in our array)**

- (NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section

{

return [pictureListData count];

}

**// Create / reuse a table cell and configure it for display**

- (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];

}

**// Get the core data object we need to use to populate this table cell**

Pictures \*currentCell = [pictureListData objectAtIndex:indexPath.row];

**// Fill in the cell contents**

cell.textLabel.text = [currentCell title];

cell.detailTextLabel.text = [currentCell desc];

**// If a picture exists then use it**

if ([currentCell smallPicture])

{

cell.imageView.contentMode = UIViewContentModeScaleAspectFit;

cell.imageView.image = [UIImage imageWithData:[currentCell smallPicture]];

}

return cell;

}

**// Swipe to delete has been used. Remove the table item**

- (void)tableView:(UITableView \*)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath \*)indexPath

{

if (editingStyle == UITableViewCellEditingStyleDelete)

{

**// Get a reference to the table item in our data array**

Pictures \*itemToDelete = [self.pictureListData objectAtIndex:indexPath.row];

**// Delete the item in Core Data**

[self.managedObjectContext deleteObject:itemToDelete];

**// Remove the item from our array**

[pictureListData removeObjectAtIndex:indexPath.row];

**// Commit the deletion in core data**

NSError \*error;

if (![self.managedObjectContext save:&error])

NSLog(@"Failed to delete picture item with error: %@", [error domain]);

**// Delete the row from the data source**

[tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath] withRowAnimation:UITableViewRowAnimationFade];

}

}

@end

**Select the Table View Controller icon on the black bar under the Table View and in the Identity Inspector**

**Change the class to PictureListMainTable.**

**Then go to the Attributes Inspector and select the Table View itself.**

**Change the style to Grouped and the Separator to Single Line.**

**Now select the prototype cell and in the attributes inspector, change the style to Subtitle.**

**This allows us to use a title, subtitle and an image without having to create those things ourselves.**

**Also set the Identifier to ‘Cell’ and change the height of the cell to 84 (in the size inspector).**

**PART 4. ADDING THE DETAIL VIEW:**

**Right-click the PictureList group and choose New File.**

**Select Cocoa Touch on the left and Objective-C Class. Call the new class PictureListDetail and make it a subclass of UITableViewController.**

**PictureListDetail.h**

#import <UIKit/UIKit.h>

#import "Pictures.h"

@interface PictureListDetail : UITableViewController <UINavigationControllerDelegate, UIImagePickerControllerDelegate>

@property (strong, nonatomic) NSManagedObjectContext \*managedObjectContext;

@property (strong, nonatomic) Pictures \*currentPicture;

@property (strong, nonatomic) IBOutlet UITextField \*titleField;

@property (strong, nonatomic) IBOutlet UITextField \*descriptionField;

@property (strong, nonatomic) IBOutlet UIImageView \*imageField;

@property (strong, nonatomic) UIImagePickerController \*imagePicker;

@end

**PictureListDetail.m**

#import "PictureListDetail.h"

@implementation PictureListDetail

@synthesize managedObjectContext;

@synthesize currentPicture;

@synthesize titleField, descriptionField, imageField;

@synthesize imagePicker;

#pragma mark - View lifecycle

- (void)viewDidLoad

{

[super viewDidLoad];

**// If we are editing an existing picture, then put the details from Core Data into the text fields for displaying**

if (currentPicture)

{

[titleField setText:[currentPicture title]];

[descriptionField setText:[currentPicture desc]];

if ([currentPicture smallPicture])

[imageField setImage:[UIImage imageWithData:[currentPicture smallPicture]]];

}

}

#pragma mark - Button actions

- (IBAction)editSaveButtonPressed:(id)sender

{

**// If we are adding a new picture (because we didnt pass one from the table) then create an entry**

if (!currentPicture)

self.currentPicture = (Pictures \*)[NSEntityDescription insertNewObjectForEntityForName:@"Pictures" inManagedObjectContext:self.managedObjectContext];

**// For both new and existing pictures, fill in the details from the form**

[self.currentPicture setTitle:[titleField text]];

[self.currentPicture setDesc:[descriptionField text]];

if (imageField.image)

{

**// Resize and save a smaller version for the table**

float resize = 74.0;

float actualWidth = imageField.image.size.width;

float actualHeight = imageField.image.size.height;

float divBy, newWidth, newHeight;

if (actualWidth > actualHeight) {

divBy = (actualWidth / resize);

newWidth = resize;

newHeight = (actualHeight / divBy);

} else {

divBy = (actualHeight / resize);

newWidth = (actualWidth / divBy);

newHeight = resize;

}

CGRect rect = CGRectMake(0.0, 0.0, newWidth, newHeight);

UIGraphicsBeginImageContext(rect.size);

[imageField.image drawInRect:rect];

UIImage \*smallImage = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

**// Save the small image version**

NSData \*smallImageData = UIImageJPEGRepresentation(smallImage, 1.0);

[self.currentPicture setSmallPicture:smallImageData];

}

**// Commit item to core data**

NSError \*error;

if (![self.managedObjectContext save:&error])

NSLog(@"Failed to add new picture with error: %@", [error domain]);

**// Automatically pop to previous view now we're done adding**

[self.navigationController popViewControllerAnimated:YES];

}

**// Pick an image from album**

- (IBAction)imageFromAlbum:(id)sender

{

imagePicker = [[UIImagePickerController alloc] init];

imagePicker.delegate = self;

imagePicker.sourceType = UIImagePickerControllerSourceTypeSavedPhotosAlbum;

[self presentViewController:imagePicker animated:YES completion:nil];

}

**// Take an image with camera this will not work in simulator**

- (IBAction)imageFromCamera:(id)sender

{

imagePicker = [[UIImagePickerController alloc] init];

imagePicker.delegate = self;

imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;

imagePicker.cameraDevice = UIImagePickerControllerCameraDeviceRear;

[self presentViewController:imagePicker animated:YES completion:nil];

}

**// Resign the keyboard after Done is pressed when editing text fields**

- (IBAction)resignKeyboard:(id)sender

{

[sender resignFirstResponder];

}

#pragma mark - Image Picker Delegate Methods

**// Dismiss the image picker on selection and use the resulting image in our ImageView**

- (void)imagePickerController:(UIImagePickerController \*)picker didFinishPickingImage:(UIImage \*)image editingInfo:(NSDictionary \*)editingInfo

{

[imagePicker dismissModalViewControllerAnimated:YES];

[imageField setImage:image];

}

**// On cancel, only dismiss the picker controller**

- (void)imagePickerControllerDidCancel:(UIImagePickerController \*)picker

{

[imagePicker dismissModalViewControllerAnimated:YES];

}

@end

**Open up PictureListMainTable.m and add the following method to the bottom…and add #import "PictureListDetail.h" to the top.**

**// When add is pressed or a table row is selected**

- (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender

{

**// Get a reference to our detail view**

PictureListDetail \*pld = (PictureListDetail \*)[segue destinationViewController];

**// Pass the managed object context to the destination view controller**

pld.managedObjectContext = managedObjectContext;

if ([[segue identifier] isEqualToString:@"EditPicture"])

{

**// Get the row we selected to view**

NSInteger selectedIndex = [[self.tableView indexPathForSelectedRow] row];

// Pass the picture object from the table that we want to view

pld.currentPicture = [pictureListData objectAtIndex:selectedIndex];

}

}

**PART 5. FINAL ALTERATIONS**

**Open up your storyboard again and drag in a TableView Controller to the right of the existing one.**

**Select the Table View Controller icon on the black bar underneath it and in the Identity Inspector, change the class to PictureListDetail.**

**Now select the table view itself and in the attributes inspector, change the content dropdown to Static Cells. Also change the sections to 2 and the style to Grouped.**

**Right-click drag from the Add button to the Picture List Detail table view and choose Push from the context menu.**

**Single-click the Segue and change the identifier to AddPicture (note the case!).**

**Next, select the single cell in our Picture List Main Table and right-click drag it to our Picture List Detail table and select Push from the menu.**

**This has wired up both the add and cell selection to our new table (remember, the code will take care of figuring out which one was selected using the identifier).**

**Select the segue for the second segue and change the identifier to EditPicture. You can tell which segue is which by selecting it. It will highlight what the segue connects by outlining it.**

**Now – over to our new table view. Delete two of the cells in the top section leaving only one and change the height of it to around 95. You can do this either by dragging the cell edge at the bottom or using the Size Inspector, checking the Custom box and entering 95 in the Row Height box.**

**Next, drag an Image View into the cell and put it on the left hand side. Make it’s size around 80 x 80 pixels (in the size inspector). Then drag two buttons to the right of it. Name one Pick from album and the other Take a photo.**

**For the second section, delete one of the cells leaving two. Drag a label into the left of the first cell and change it to Title:.**

**Do the same for the second cell and change the label to Description**

**Next, drag a Text Field into the first cell to the right of the label. Under the Attributes Inspector, change the text field properties so that it has an invisible border and give it a place holder description. Do the same for the Description cell.**

**Now drag a Bar Button Item onto the right hand side of the navigation bar. In the Attributes Inspector, change the Identifier to Save. Also double click the navigation bar and change the title to Picture Details.**

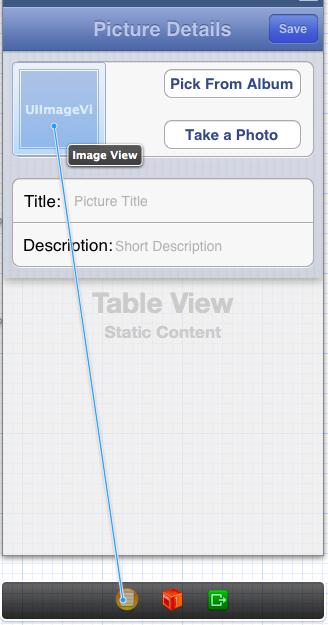
**Now we will reverse wire out elements:**

**Now lets do the final wiring and we can test this out.**

**First the Save button. Right-click drag it to the Picture List Detail icon in the black bar and choose editSaveButtonPressed**

****

**Now right-click drag from the Picture List Detail icon in the black bar, to the Image View and select imageField from the menu.**

****

**Do the same again for the two Text Fields and select the appropriate property (titleField and descriptionField).**

**Right-click the first Text Field and drag the circle to the right of Did end on exit to the Picture List Detail icon and select resignKeyboard.**

**Do the same for the second Text Field again choosing resignKeyboard.**

**Finally, right-click drag from the Pick from Album button to the Picture List Detail icon and choose imageFromAlbum.**

**Do the same for the Take a photo button and choose imageFromCamera.**

**NOW WE CAN RUN AND TROUBLE SHOOT**