Tutorial: Simple iOS App

By the iOS team: Eli Evans, Molly Flaccavento, Dana Germano, Lenny Magill, Cristiane Maia, Katie Pritchard

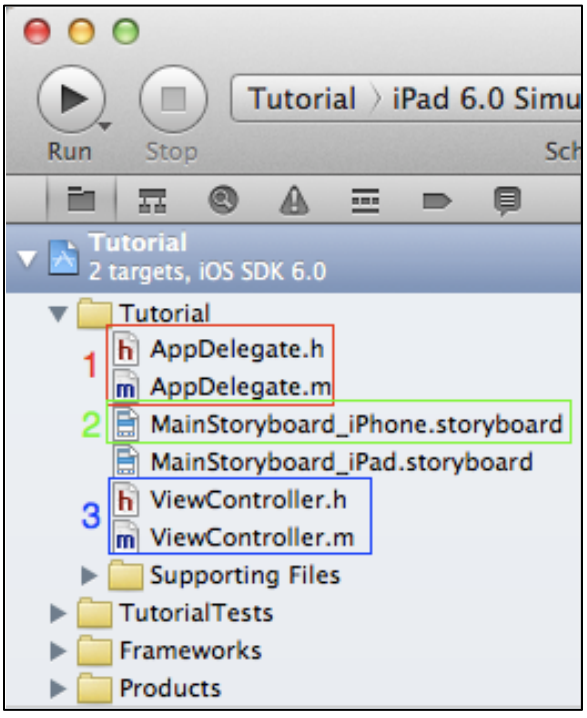
The final product of this tutorial is a basic multi-view app that allows the user to take a picture and attach it in an email.

Introduction - Setting up Xcode, basic methods, basic setup

Creating a new project:

1. Open *Xcode.*
2. Click *Create a New Xcode Project.*
3. Under the iOS category in the left sidebar, click *Application*, and choose *Single View Application*.
4. Enter a *Product Name* and a *Company Identifier* of your choosing for your project. For this tutorial, use:
   - Devices: *Universal*
   - Check *Use Storyboards*
   - Check *Use Automatic Reference Counting*
   - Uncheck *Include Unit Tests*
5. Hit *Next* and choose a location to save your project, then hit *Create.*

What you should see:



In the left sidebar is the *Project Navigator* (if not visible, click the gray file folder tab at the top):

1. *AppDelegate:* You should see two AppDelegate files, a header (.h) and a main (.m), similar to C programs. The AppDelegate is already created within every new project you create. The application delegate (AppDelegate for short) aids in the tasks related to the application. Typically it handles startup and startdown, transitions, and similar application-wide tasks.

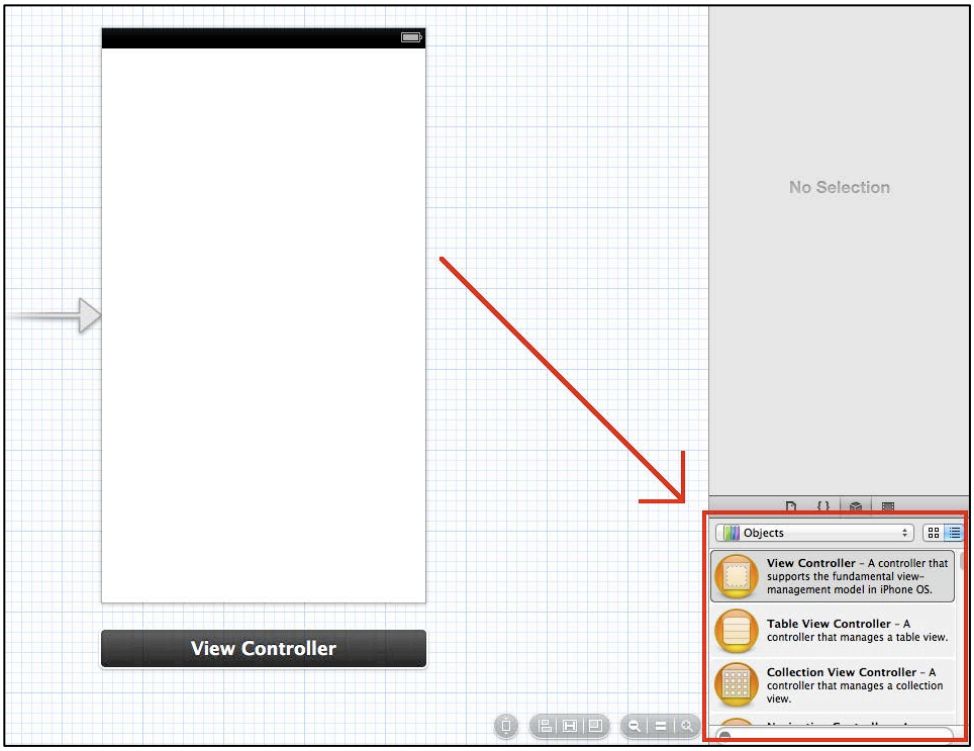
More information can be found in the Apple documentation (Help >> Documentation and API Reference >> Search "UIApplicationDelegate" >> click one of the results in the left sidebar), or use the online version found at https://developer.apple.com/library/ios/navigation/.

2. *MainStoryboard*: The new project, because we selected "Use Storyboards" in creating it, includes two main Storyboards: MainStoryboard_iPhone.storyboard and MainStoryboard_iPad.storyboard. For the purposes of this tutorial we will focus on the iPhone storyboard. The iPad storyboard can be implemented separately, but developing for the iPhone still allows iPad users to use iPhone apps in actual or doubled size.

3. *ViewController*: simply named, this file is the main view controller of your application. If you click the MainStoryboard_iPhone.storyboard file, you should see a single blank screen with a bottom label of "*ViewController*". This is the main view of your application currently, and if you click the *Run* button in the upper left-hand corner of the *XCode* window, the single screen should appear in the simulator. Make sure the menu next to the *Stop* button reads *iPhone Simulator* (as shown below). If it does not, click the bar and select it from the drop-down.

Text on the Screen:

Click on the *MainStoryboard_iPhone.storyboard* file and you will see the single view controller. In order to add text onto this screen, we need to add a *Label* (text box). In the bottom right-hand corner of the right *Organizer* view, there should be a library of objects including things like *View Controller*, *Object*, *Rounded Rect Button*, etc. Scroll down to *Label*, and click + drag that onto the white *View Controller* screen.

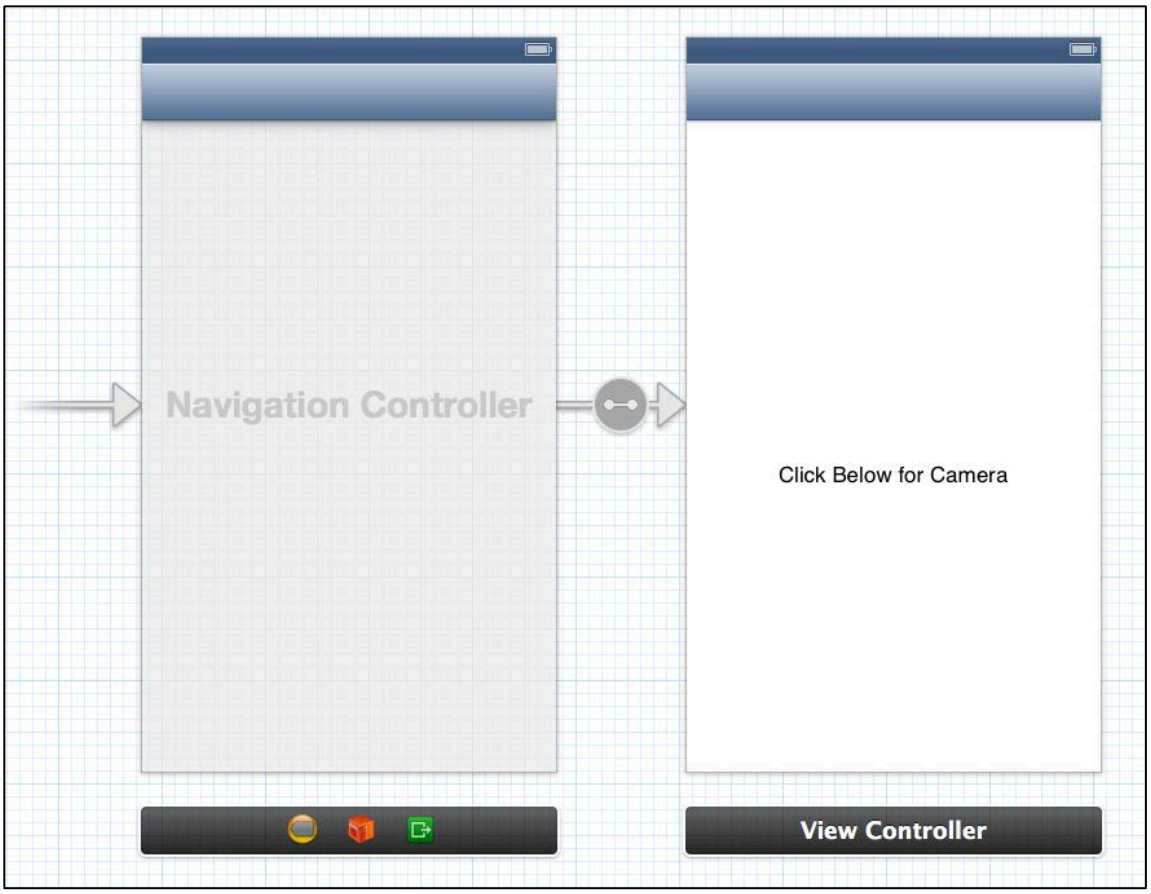


A *Label* should now be on your screen. Drag it to the center of your screen or wherever you would like it, and double click the Label to edit the displayed text. Change the text to your own message, for this tutorial we will use the phrase “Click below for Camera”.

If you click the *Run* button in the upper left-hand corner of *Xcode*, your text should now be displayed on the screen.

Create Button to link to new screen:

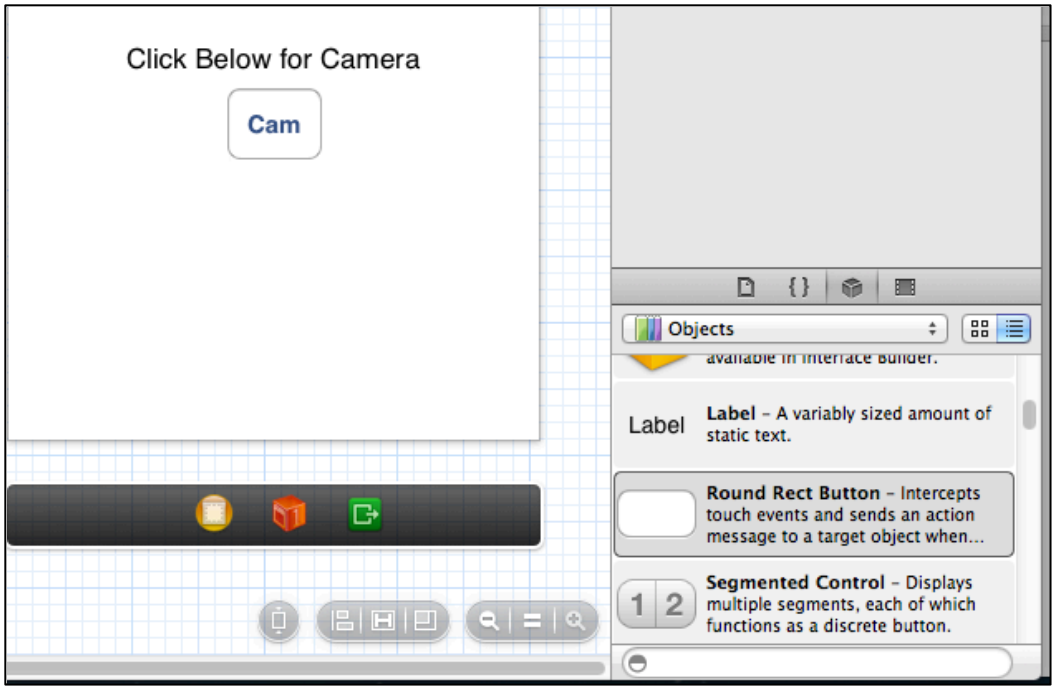
First we want to make the application have a *Navigation Controller* to manage the interaction of different screens. To do this, open the *Storyboard* and highlight the *View Controller* from within the storyboard sidebar to the left (the *View Controller* should become outlined in blue), then choose Editor >> Embed in >> Navigation Controller. You should see a grayed Navigation Controller pointing to the View Controller screen, and blue Navigation Bar at the top of the View Controller, as shown here:



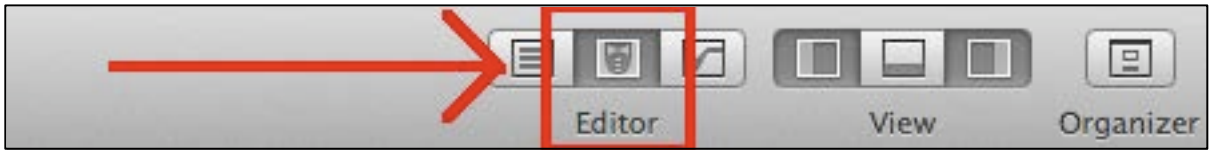
Now to add a *Button*, find and drag the *Rounded Rect Button* from the *Objects Library* (same list where *Label* was found) onto the *View Controller* screen below the label.

To change the text on the button, double click it and change the text of the button to the text you want on the button. For this tutorial, we

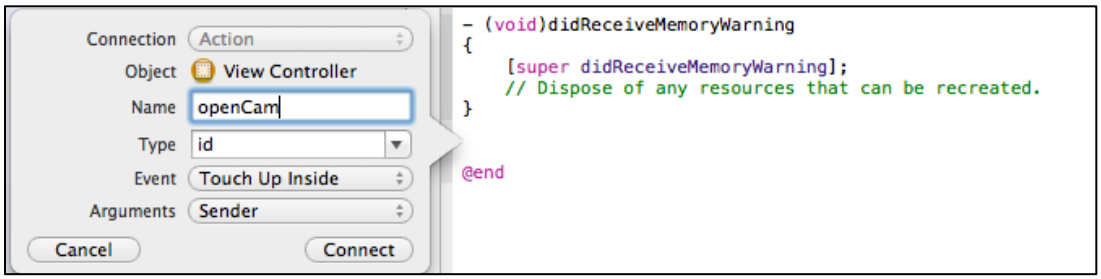
will use “Cam”.



If you click Run now, the button should appear and though it is clickable, nothing will happen. We need to link the button to an action so something happens when it is clicked. To do this, it is helpful to open the dual-screen editor, which you do by clicking the “tuxedo-like” button in the upper right corner. This should open two windows, so make sure the storyboard is open in one window, and the *ViewController.m* open in the other.



Now, hold down the control button and use the mouse to click on the *Cam* button and drag into the *ViewController.m* file. This should open a dialog that asks about Connection, name, type, etc.



The connection should be *Action*, and fill in the *Name* field with "openCam" then hit *Connect*. This should create an empty function that looks like the following:

```
- (IBAction)openCam:(id)sender {
}
```

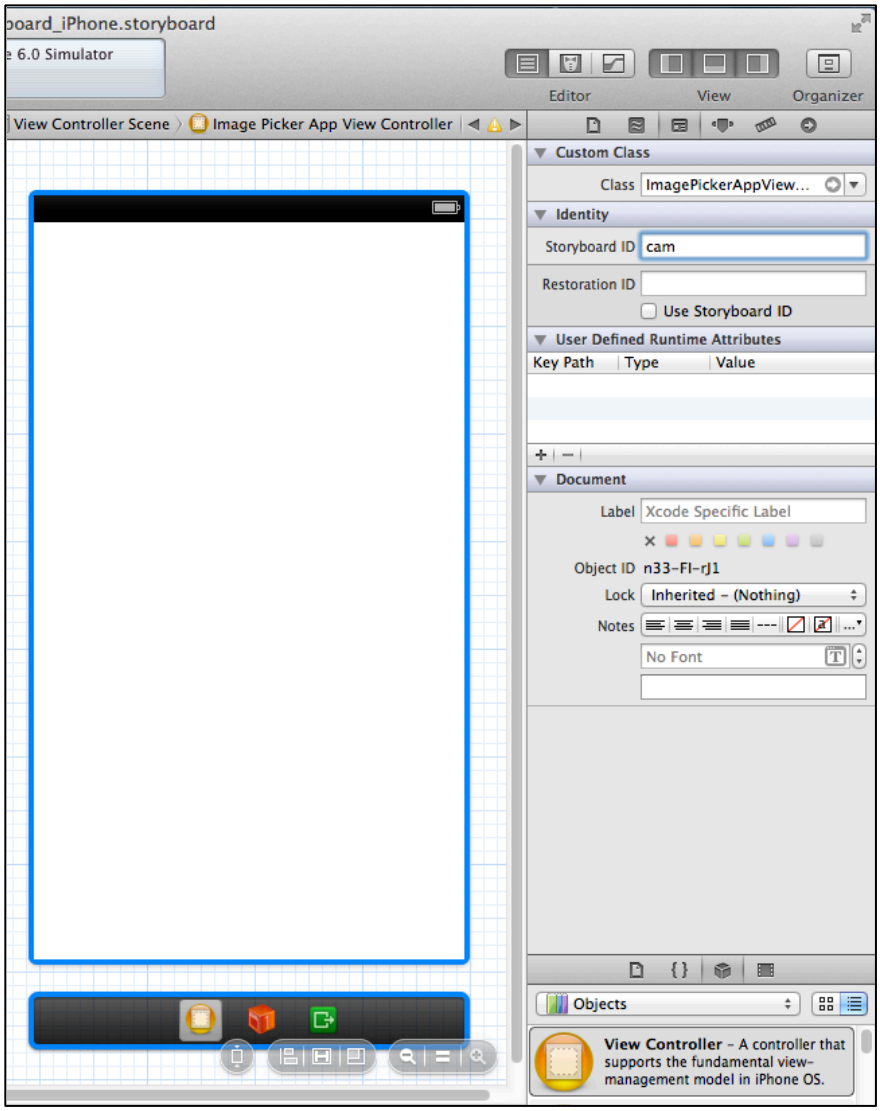
The IBAction indicates an action that happens upon some event, in this case the pressing of a button, and the sender is the button itself. Add "NSLog(@"Cam button pressed");" to the function so it looks like the following.

```
- (IBAction)openCam:(id)sender {
    NSLog(@"Cam button pressed");
}
```

If you save and *Run* at this point, and press the *Cam* button, the message "Cam button pressed" should appear in the bottom console section of *Xcode*.

Now we will add a new class via File >> New >> File... Select Objective-C class and name the class "ImagePickerAppViewController". Make sure it is a subclass of *UIViewController*, then hit *Next,* and *Create* the file in the current project. This should have created an *ImagePickerAppViewController.m* file and an *ImagePickerAppViewController.h* file. Now to add this new screen to the storyboard, drag a *View Controller* onto the storyboard from *Object Library* (bottom right of window). Select the new *View Controller* and choose the *Identity Inspector* tab from the top of the right sidebar.

Under the *Custom Class* header, choose the *ImagePickerAppViewController* from the drop down menu. This should update the title bar in the left sidebar of the storyboard. Next, give the *ImagePickerAppViewController* a *Storyboard ID*. We will use “cam”.



Now that the storyboard is linked to the *ImagePickerAppViewController* file, we need to implement the method to push on a new screen for the camera. Return to the ViewController.m file. First, import the *ImagePickerAppViewController* by adding the following line below the “import ViewController.h” line at the top of the file:

```
#import "ImagePickerAppViewController.h"
```

Then within the openCam: method, change the code to the following:

```
- (IBAction)openCam:(id)sender {
    ImagePickerAppViewController *imagePicker = [[self
storyboard]instantiateViewControllerWithIdentifier:@"cam"];
    [imagePicker setTitle:@"Camera"];
    [[self navigationController]pushViewController:imagePicker animated:YES];
}
```

This makes creates an instance of *ImagePickerAppViewController* and signals the *Navigation Controller* to push on the new view. Hit Run now, and make sure that when the “Cam” button is pressed, the new *Camera* View Controller is pushed onto the screen.

Camera - Implementing camera

To use the camera on an iPad or iPhone, *Xcode* offers of some simple framework that will let us take pictures, save pictures, or choose a picture from the camera roll to display. We will learn how to implement all of these functionalities. Most of the heavy lifting for this will be done by the *ImagePickerController*, which can accomplish all of these tasks.

To start we need to declare the class in the *ImagePickerAppViewController.h* file and designate it as a delegate:

```
@interface ImagePickerAppViewController : UIViewController
<UINavigationControllerDelegate, UIImagePickerControllerDelegate> {
    UIImagePickerController *picker;
    IBOutlet UIImageView *selectedImage;
}
```

We will let the image picker know that it is needed through the use of a button, so we need to have a way of handling the click of the button to communicate the event.

In *ImagePickerAppViewController.h*, add the following property under the class declaration (but before the @end tag):

```
@property(nonatomic, retain)UIImageView *selectedImage;
-(IBAction) buttonClicked;
```

In the *ImagePickerAppViewController.m* file, add the following tag below the "*@implementation*" tag.

```
@synthesize selectedImage;
```

This will generate setters and getters for the property we just declared in the header file. Next, create a new action in the body of the file (immediately before the @end tag):

```
-(IBAction) buttonClicked
{
    myPicker = [[UIImagePickerController alloc] init];
    myPicker.delegate = self;
    if([UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerSourceTypeCamera])
    {
        myPicker.sourceType = UIImagePickerControllerSourceTypeCamera;
    }else{
        myPicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    }
    [self presentViewController:myPicker animated:YES completion:NULL];
}
```

This action will check to make sure the application has a camera and choose the proper source for the image picker once our button has been clicked. Once the button has been clicked, we will want to give the user the option to cancel out of this screen, so we need to make sure a cancel method is implemented. We can use *imagePickerControllerDidCancel*.

```
- (void) imagePickerControllerDidCancel:(UIImagePickerController *) picker {
    [[picker parentViewController] dismissViewControllerAnimated:YES completion:NULL];
}
```

We will also need to acknowledge when an image has been selected. We can do this with a method called i*magePickerControllerdidFinishPickingMediaWithInfo*.

```
- (void)imagePickerController:(UIImagePickerController *) picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {
    selectedImage.image = [info objectForKey:UIImagePickerControllerOriginalImage];
    [[picker parentViewController] dismissViewControllerAnimated:YES
completion:NULL];
}
```

To display the photo on the current screen we first need to add an ImageView. Using the storyboard builder, drag an Image View from the *Object Library* object onto the Image Picker App view and link the UIImageView to selectedImage. You can do this by opening ImagePickerAppViewController.h in the assistant view that we used earlier (opened by clicking the "tuxedo-like" button). Then, select the UIImageView object. While holding the control key, drag from the UIImageView onto the "IBOutlet UIImageView *selectedImage;" line in the header file (a blue box should appear as you drag your cursor over the line). Now you can resize the object to occupy roughly three quarters of the iPhone's screen (see below).

Next, create and connect a *Pick An Image* button to the button clicked method. You can do this by dragging a button object onto the Image Picker App view and renaming it to "Pick An Image". Then, in the same way that you did for the UIImageView, hold control and drag from the *Pick an Image* button to the line "-(IBAction) buttonClicked;" in the header file. You can test that these feature are working by running the iPhone simulator. Since the simulator does not have a camera available, the photo gallery should automatically appear when *Pick An Image* is clicked.

E-mailing photo - E-mailing photo to yourself

In the Storyboard we are going to add the buttons for the view we have created. In the *Image Picker App View Controller* that the *UIImageView* is now in, add a navigation bar with a share button:

1. Find a *Navigation Bar* in the *Object Library.* Drag it onto the view.
2. Change the title of the bar to “Image Picker” by double clicking the text and typing.
3. Find a *Bar Button Item* in the *Object Library.* Drag it onto the Navigation Bar.
4. Rename the button to “Share” by double clicking and typing.

Next we need to import the framework into our project that will contain the methods needed for emailing. To do this, click the project name (“Tutorial”) so it is highlighted in the *Project Navigator*. Click the *Build Phases* tab at the top of the middle window. Then expand the *Link Binary With Libraries* menu and click the “+” button that appears. Search for “MessageUI.framework” and click *Add* once you have found it. Drag newly appeared framework into *Frameworks Folder* of the *Project Navigator*.

Next we will be adding the code to our view controller files to generate an email with our picture attached. In *ImagePickerAppViewController.h*:

1. Add an Import statement for <MessageUI/MessageUI.h>.
2. Make the following connections to the share button by clicking ctrl and dragging:
    - From the *Share* button:
        - A. Make an outlet connection that has UIBarButtonItem type, strong storage and is named “email”.
        - B. Make an IBAction connection of type *id* named “share”.
    - From the *UIImageView* object:
        - A. Add the following line to designate the picture as the data being emailed.

```
-(void) emailItem :(NSData*)data;
```

B. Make an outlet connection that has *UIImageView* type, strong storage and named “picture”.

This is what your class should look like once you have completed the previous steps…

```
#import <UIKit/UIKit.h>
#import <MessageUI/MessageUI.h>

@interface ImagePickerAppViewController : UIViewController <UINavigation
    UIImagePickerController *myPicker;
    IBOutlet UIImageView *selectedImage;
}

@property(nonatomic, retain)UIImageView *selectedImage;
-(IBAction) buttonClicked;

@property (strong, nonatomic) IBOutlet UIBarButtonItem *email;
-(IBAction)share:(id)sender;

-(void) emailItem :(NSData*)data;

@property (strong, nonatomic) IBOutlet UIImageView *picture;
@end
```

In *ImagePickerAppViewController.m* insert these two methods:

Share Action Method:

(This method will already have auto-generated. Add code to body)

```
- (IBAction)share:(id)sender {
    NSData *imageData = UIImagePNGRepresentation(self.picture.image);
    [self emailItem:imageData];
}
```

This method sends a signal to the *emailItem* method to create the email using the “picture” stored in the *UIImageView*. The “picture” is assigned to the *NSData* object and passed as a parameter to *emailItem.*

EmailItem Method:

```
-(void) emailItem :(NSData*)data{
    Class mailClass=(NSClassFromString(@"MFMailComposeViewController"));
    if([mailClass canSendMail]){
        // Creates a view that can compose and send mail
       MFMailComposeViewController *emailUI=[[MFMailComposeViewController
alloc] init];
        // Sets the subject line as “picture”.
```

```
        [emailUI setSubject:@"picture"];
            // Attaches the image to the email
        [emailUI addAttachmentData:data mimeType:@"image/jpeg" fileName:@"pic"];
        [self presentViewController:emailUI animated:YES completion:NULL];
        [emailUI setModalTransitionStyle:UIModalTransitionStyleCoverVertical];
    }else{
            // If the phone cannot send mail…
        UIAlertView *noEmail = [[UIAlertView alloc] initWithTitle:@"no email"
message:@"there is not e-mail address associated with this phone" delegate:self
cancelButtonTitle:@"OK" otherButtonTitles:nil, nil] ;
        [noEmail show];
    }
}
```

This method pushes the view controller that has a mail composer. It allows the user to send the image as an attachment and automatically fills in the subject line if the device is email enabled.

Congrats! You finished your first iOS application.