# Table views

## Table views

* Iconic and common element of IOS
* Many stock apps are built around table views
* Actually is hard find a example of stock app does not has table view.

## Table view data

* Managed/displayed by UITableViewController
* Information requested as needed
  + When the TableViewController is displayed, the system asks the controller for information for its cells only when it needs it. Opposite of loading all the data at once.

## Table view navigation

* When a cell is tapped, a new screen is revealed.
* Stack of cards
* New cards can be added to the top, also you can remove the cards on the top to reveal what’s undernearth.
* Mail app
* Managed by a UINavigationViewController

# Choosing Images

## Setting up the Storyboard

* start up a new “single view application” project
* A button
* A UIImageView to show our loaded image
* Add an outlet to our image UIImageView
* Add an action to button to load the UIImagePickerController

## Loading the UIImagePickerController

* protocol to properly use UIImagePickerController
* an instance of UIImagePickerController

// ViewController.swift

// IOS Camera App

//

// Created by 诗迪杨 on 26/03/2016.

// Copyright © 2016 group 11. All rights reserved.

//

import UIKit

class ViewController: UIViewController, UIImagePickerControllerDelegate, UINavigationBarDelegate {

@IBOutlet weak var PhotoLibrary: UIButton!

@IBOutlet weak var Camera: UIButton!

@IBOutlet weak var ImageDisplay: UIImageView!

override func viewDidLoad() {

super.viewDidLoad()

// Do any additional setup after loading the view, typically from a nib.

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

@IBAction func PhotoLibraryAction(sender: UIButton) {

let picker = UIImagePickerController()

//constant picker

picker.delegate = self

//Set our view controller as the delegate for the UIImagePickerController

picker.sourceType = .PhotoLibrary

//PhotoLibrary: gives the user access to any of their photos available on the devices with albums, overall view, and cameral roll

//SavedPhotosAlbum is pretty much just that, the images saved locally to your device, but no access to albums or any of the other views

//present a view controller modally

presentViewController(picker, animated: true, completion: nil)

//completion: The block to execute after the presentation finishes.

}

@IBAction func CameraAction(sender: UIButton) {

}

}

# UIStackView()

Provide a way to lay out a series of views horizontally or vertically. By configuring a few simple properties such as alignment, distribution, and spacing.

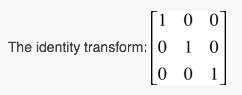
# CGGeometry

* CGRectZero: a x = 0, y = 0, width = 0, height = 0 rect

# CGAffineTransform

represents a matrix used for affine transformations. A transformation specifies how points in one coordinate system map to points in another coordinate system.

* CGAffineTransformIdentity



* CGAffineTransformTranslate

Return an affine transformation matrix constructed by translating an existing affine transform

func CGAffineTransformTranslate(t, tx, ty)

* + t: existing affine transform
  + tx: the value by which to move x values with the affine transform
  + ty: the value by which to move y values with the affine transform
* CGAffineTransformRotate

Returns an affine transformation matrix constructed by rotating an existing affine transform

func CGAffineTransformRotate(t, angle)

* t: an existing affine transform
* angle: in radius, in iOS positive value specifies counterclockwise rotation. In OS X, a positive value specifies clockwise rotation.

# Using Objective-C and Swift in the same project

## Bridging-Header.h:

#import “MyDataModel.h”

## MyDataModel.h:

@interface MyDataModel : NSObject

//set properties

@property (strong) NSString\* stringProperty;

@property int intProperty;

-(int) intPropertySquared;

## MyDataModel.m

@implementation MyDataModel

//set method

* (int) intPropertySquare{

return self.intProperty \* self.intProperty;

}

## ViewController.swift

override func viewDidLoad(){

let a = MyDataModel()

a.stringProperty = “String data”

a.intProperty = 45

print(“Result: \(a.intPropertySquared())”)

}

# Classes

## NSObject

Base class for all Objective-C classes

Some advanced feature will require you to subclass from NSObject

## NSNumber

Generic number-holding class

Let n = NSNumber(35.5)

Let intversion = n.intValue //also doubleValue, boolValue, etc

## NSDate

* Used to find out the date and time right now or to store past or future dates.
* NSCalendar, NSDateFormatter, NSDateComponents
* If you are displaying a date in your UI, there are localization remification

## NSData

Used to save/restore/transmit raw data throughout the Ios SDK.

# Debug

## Thread 1: breakpoint 1.1

you've set a breakpoint by mistake. Press CMD(?)+7 to select all breakpoints in Xcode. In the breakpoint navigator on the left you can now select and delete your breakpoints with backspace. Alternatively, if you see one of these (blue) breakpoints in the gutter you can simply drag it away for deletion.

## Xcode process launch failed: Security

As from iOS 9, it is required to go to Settings → General → DeviceManagement → DeveloperApp → Trust.

## Unlike the button and the code

Right click view controller-> check the outlet and the action

# Glossary

1. Navigation controller: a view controller subclass that manages transitions backward and forward through a serial of view controllers.