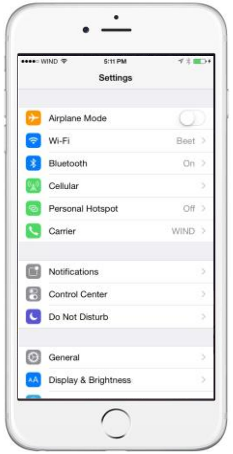
Navigation Patterns

Objectives

1. Progress through pages of data with stack-based navigation
2. Show different views of related data with tab navigation
3. Display hierarchical relationships with master/detail navigation
4. Organize pages of information with page navigation

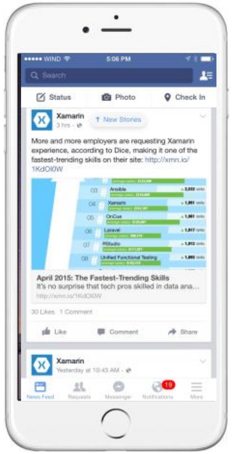
Navigation Patterns

- iOS provides several ways to structure navigation in your application – must decide the most effective way to present your information


Settings

Airplane Mode	
Wi-Fi	Beet
Bluetooth	On
Cellular	
Personal Hotspot	Off
Carrier	WIND
Notifications	
Control Center	
Do Not Disturb	
General	
Display & Brightness	

Stack



Xamarin

More and more employers are requesting Xamarin experience, according to Dice, making it one of the fastest-trending skills on their site: http://xmn.io/1KdOl0W

April 2015: The Fastest-Trending Skills

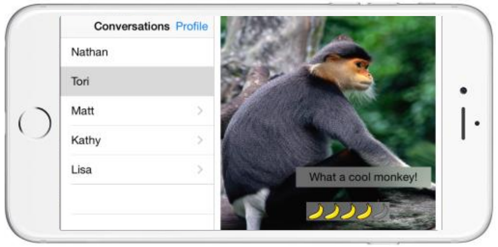
It's no surprise that tech pros skilled in data ana…

30 Likes 1 Comment

Xamarin

Yesterday at 10:43 AM

Tabs



Nathan

Tori

Matt

Kathy

Lisa

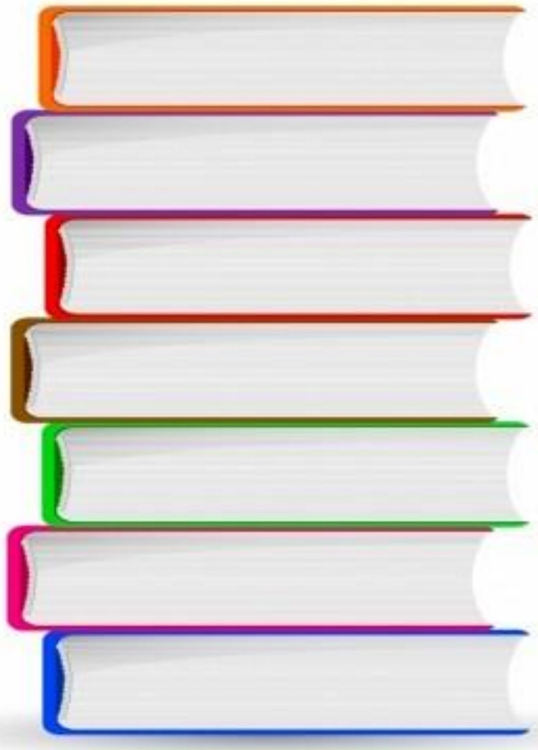
What a cool monkey!

Master/Detail

Progress through pages of data with stack-based navigation

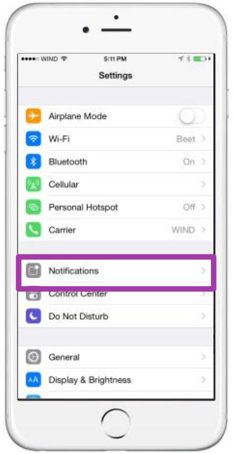
Tasks

1. Create a Navigation Controller programmatically
2. Utilize the designer to create a Navigation Controller
3. Customize the Navigation Controller



Stack Navigation

❖ When we have a hierarchy of data, it’s convenient to use stack navigation to browse and interact with the content


●●●●○ WIND 5:11 PM

Settings

Airplane Mode

Wi-Fi Beet

Bluetooth On

Cellular

Personal Hotspot Off

Carrier WIND

Notifications

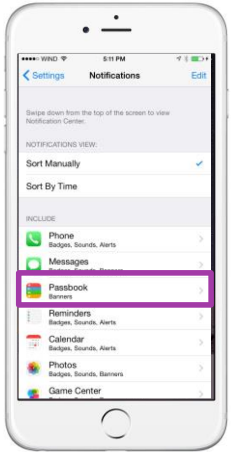
Control Center

Do Not Disturb

General

Display & Brightness

Settings


●●●●○ WIND 5:11 PM

Settings Notifications Edit

Swipe down from the top of the screen to view Notification Center.

NOTIFICATIONS VIEW:

Sort Manually ✓

Sort By Time

INCLUDE

Phone
Badges, Sounds, Alerts

Messages
Badges, Sounds, Banners

Passbook
Banners

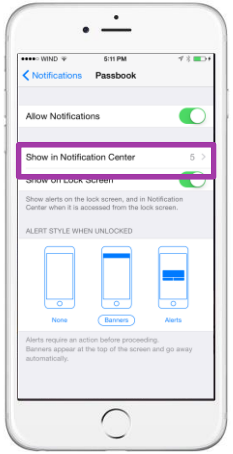
Reminders
Badges, Sounds, Alerts

Calendar
Badges, Sounds, Alerts

Photos
Badges, Sounds, Banners

Game Center

Notifications


●●●●○ WIND 5:11 PM

Notifications Passbook

Allow Notifications

Show in Notification Center 5

Show on Lock Screen

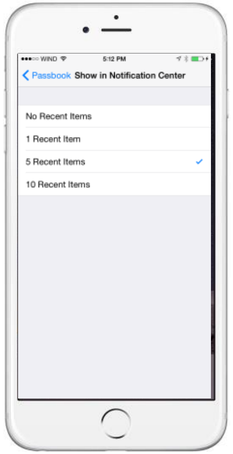
Show alerts on the lock screen, and in Notification Center when it is accessed from the lock screen.

ALERT STYLE WHEN UNLOCKED

None [✓ Banners] Alerts

Alerts require an action before proceeding. Banners appear at the top of the screen and go away automatically.

Passbook


●●●○○ WIND 5:12 PM

Passbook Show in Notification Center

No Recent Items

1 Recent Item

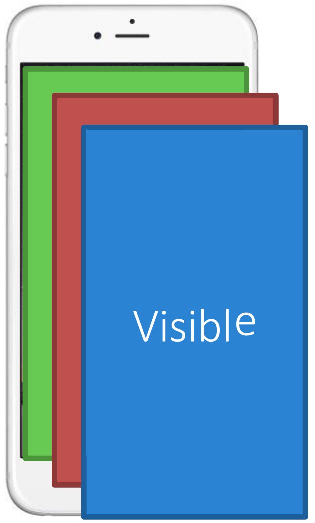
5 Recent Items ✓

10 Recent Items

Notification Center

Stack Navigation

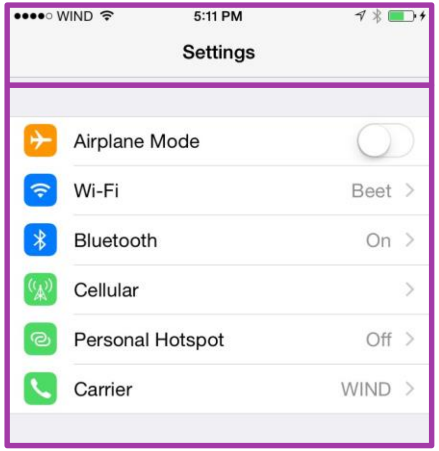
- When a new view controller is *pushed* onto the stack, it becomes visible and hides the previous screen
- Only one view controller is ever visible at a time (the last one added)
- Great for displaying multi-level relationships because it allows "drilling" into details



What is UINavigationController?

- Stack-based navigation is built into iOS through the use of the `UINavigationController` class

Displays a Navigation Bar above the currently displayed view controller



Acts as a parent to any number of child view controllers (stored in a stack)

Create a Navigation Controller

- We can create a **`UINavigationController`** programmatically, most often added as the root view controller for the app

```
public override bool FinishedLaunching(UIApplication application,
                                       NSDictionary launchOptions)
{
   window = new UIWindow (UIScreen.MainScreen.Bounds);
   var navVC = new UINavigationController(new FirstPageVC());

   window.RootViewController = navVC;
   window.MakeKeyAndVisible();
   return true;
}
```

Can pass in the initial view controller to display on the constructor

Forward Navigation [Programmatically]

❖ To navigate forward, we add (or "push") a child View Controller onto the Navigation Controller's stack

```
UINavigationController navVC = ...;
...
navVC.PushViewController(newViewController, animated:true);
```

↑

Displays the new View Controller and adds it to the navigation stack, if there was a view controller already shown, then it is hidden and a back button is added to the navigation bar

The NavigationController property

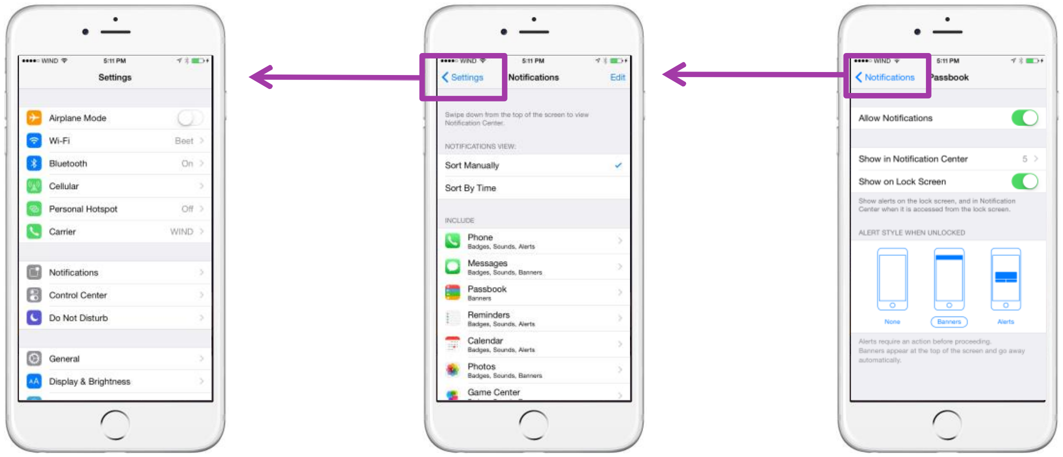
- View Controllers that have been added to the navigation controller can use the **`NavigationController`** property to access it

```
var navCon = myViewController.NavigationController;

navCon.PushViewController(newViewController, animated:true);
```

The **`NavigationController`** property is only valid when this view controller is owned by a navigation controller, otherwise it will be **`null`**!

Back Navigation [definition]

❖ *Back Navigation* removes the top viewcontroller and navigates back through the stack of child screens



Settings

- Airplane Mode
- Wi-Fi: Beet
- Bluetooth: On
- Cellular
- Personal Hotspot: Off
- Carrier: WIND
- Notifications
- Control Center
- Do Not Disturb
- General
- Display & Brightness

< Settings | Notifications | Edit

Swipe down from the top of the screen to view Notification Center.

NOTIFICATIONS VIEW:

- Sort Manually ✓
- Sort By Time

INCLUDE

- Phone: Badges, Sounds, Alerts
- Messages: Badges, Sounds, Banners
- Passbook: Banners
- Reminders: Badges, Sounds, Alerts
- Calendar: Badges, Sounds, Alerts
- Photos: Badges, Sounds, Banners
- Game Center

< Notifications | Passbook

- Allow Notifications: ✓
- Show in Notification Center: 5
- Show on Lock Screen: ✓

Show alerts on the lock screen, and in Notification Center when it is accessed from the lock screen.

ALERT STYLE WHEN UNLOCKED

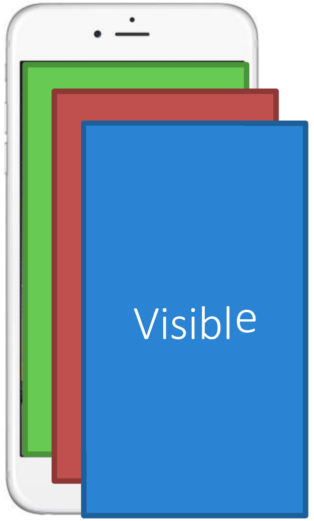
None [✓ Banners] Alerts

Alerts require an action before proceeding. Banners appear at the top of the screen and go away automatically.

Back Navigation

- When a screen is "popped" off the stack, it's removed from the Navigation Controller and the screen below becomes visible

- Must always have at least one view controller on the stack – popping off the last entry will result in an error



Back Navigation [programmatically]

- We can navigate back using the **`PopViewController`** method on the Navigation Controller

```
var navVC = new UINavigationController(clockVC);
...
navVC.PopViewController(animated:true);
```

```
navVC.PopToRootViewController(animated:true);
```

Fine-grained stack manipulation

- Navigation controller includes methods to influence the stack directly and properties to interrogate the current state of the navigation stack

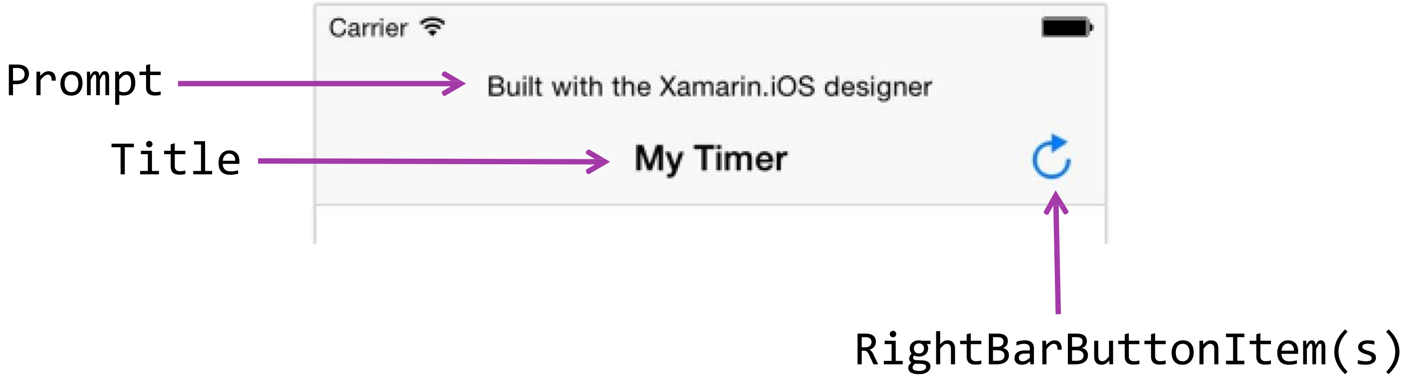
```
public class UINavigationController
{
   UIViewController TopViewController { get; }
   UIViewController VisibleViewController { get; }
   UIViewController[] ViewControllers { get; set; }

   UIViewController[] PopToRootViewController(bool animated);
   UIViewController[] PopToViewController(
                                UIViewController viewController, bool animated);
   void SetViewControllers(UIViewController[] controllers, bool animated);
}
```

Demonstration

Stack Navigation programmatically

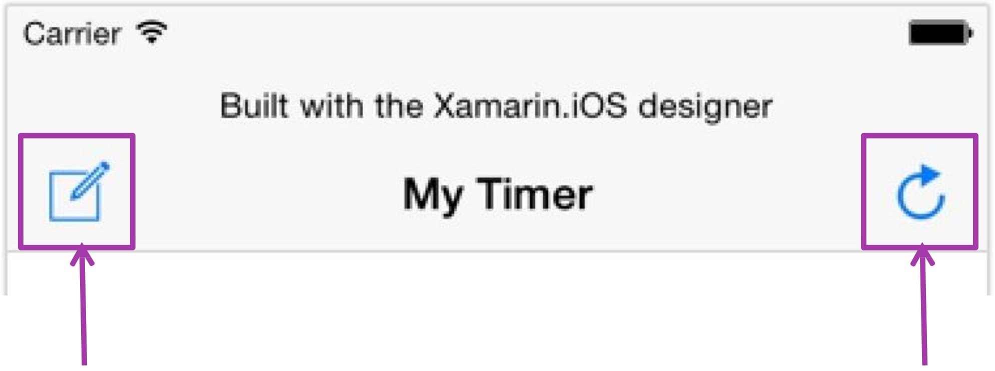
The NavigationItem property

- Every View Controller has a **`NavigationItem`** property that can be used to change the behavior and appearance of the Navigation Controller



UIBarButtonItem

❖ `UIBarButtonItem` objects can be used to add buttons to the Navigation Bar



`LeftBarButtonItem(s)` `RightBarButtonItem(s)`

UIBarButtonItems

❖ The **`UIBarButtonItems`** are available in the **`UINavigationItem`** property

`LeftBarButtonItem`	`RightBarButtonItem`	`BackBarButtonItem`
Appears to the left of the title – note this replaces the default back button	Appears to the right of the title on the navigation bar	Appears to the left of the title when navigating one level deeper

You can add multiple buttons on the left and right sides using the plural forms which take an array of buttons – **`LeftBarButtonItems`** and **`RightBarButtonItems`**

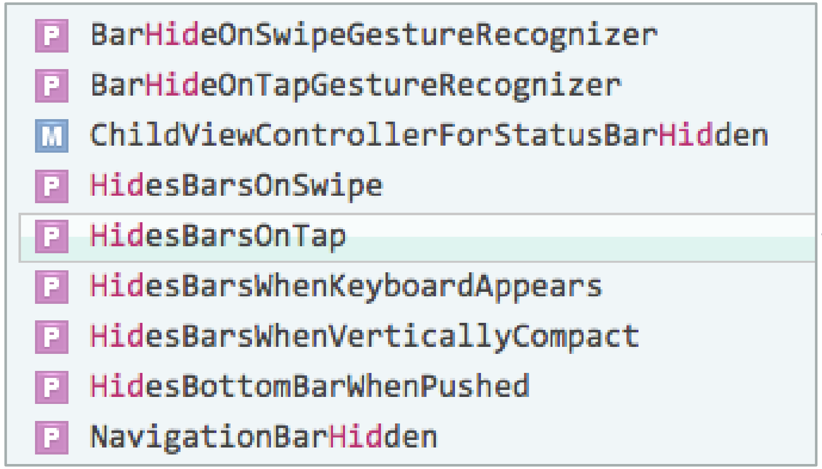
Adding BarButtonItems programmatically

- To add bar button items programmatically, set the properties in the currently active `UIViewController` using the `NavigationItem` property

```
this.NavigationItem.LeftBarButtonItem = new UIBarButtonItem(...);

this.NavigationItem.RightBarButtonItem = new UIBarButtonItem(...);

this.NavigationItem.BackBarButtonItem = new UIBarButtonItem(...);
```

Hiding the Navigation Bar

- If targeting iOS8 or above, the visibility of the navigation bar can be changed by setting properties on the `UINavigationController`

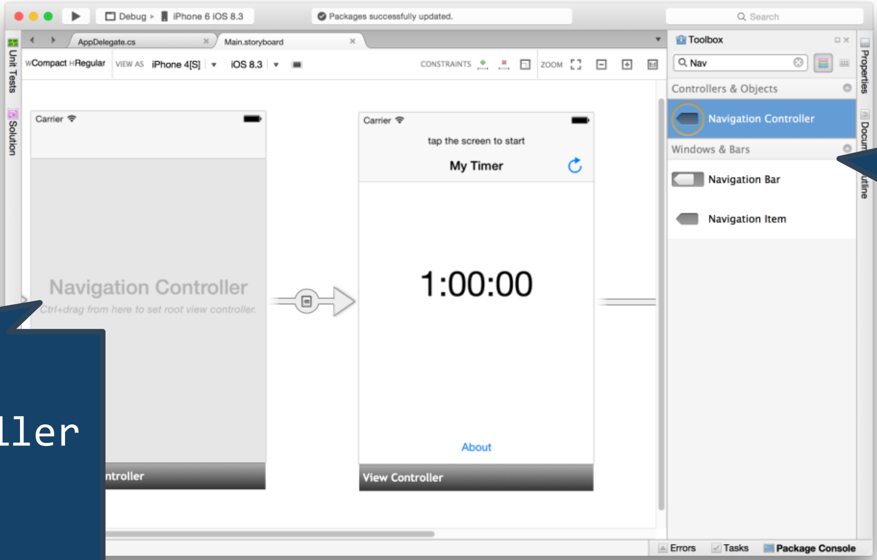


```
P BarHideOnSwipeGestureRecognizer
P BarHideOnTapGestureRecognizer
M ChildViewControllerForStatusBarHidden
P HidesBarsOnSwipe
P HidesBarsOnTap
P HidesBarsWhenKeyboardAppears
P HidesBarsWhenVerticallyCompact
P HidesBottomBarWhenPushed
P NavigationBarHidden
```

These properties are accessed directly from an instance of a `UINavigationController`

UINavigationController in the Designer

- We can add a **`UINavigationController`** to a Storyboard using the iOS Designer

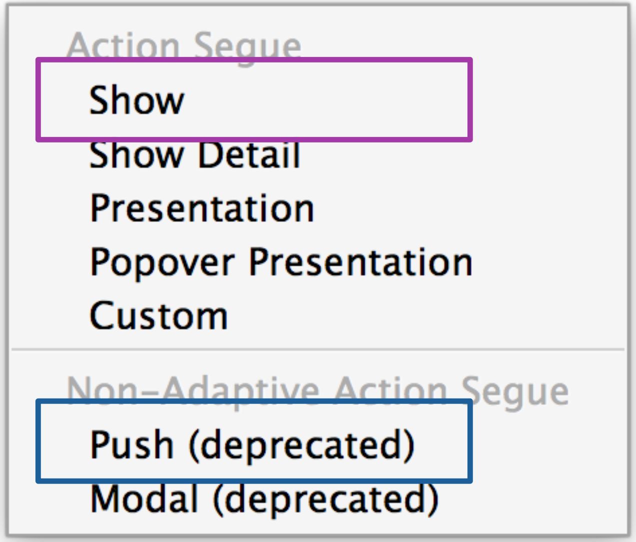


Use the Toolbox and search for Navigation Controller

Design provides a `UINavigationController` and one child `UIViewController`

Navigation using Segues

- To add view controllers to the stack using the designer on all versions of iOS, we use a Push Segue, on iOS8+ we use the Show Segue



Prepare for Segue

- To interact with a View Controller before it's displayed via a Segue, override the **PrepareForSegue** method on the source View Controller

```
public override void PrepareForSegue(
    UIStoryboardSegue segue, NSObject sender)
{
    base.PrepareForSegue (segue, sender);

    var aboutVC = segue.DestinationViewController as
        AboutViewController;
    ...
}
```

Must cast the destination View Controller to access custom properties and methods

Instantiating a View Controller

❖ To instantiate a View Controller defined in a Storyboard programmatically, use the `InstantiateViewController` method

```
UIViewController controller = ...;
var sb = controller.Storyboard;

var newVC = sb.InstantiateViewController ("myViewController");

controller.PushViewController (newVC, true);
```

Can then push the new controller onto the navigation stack

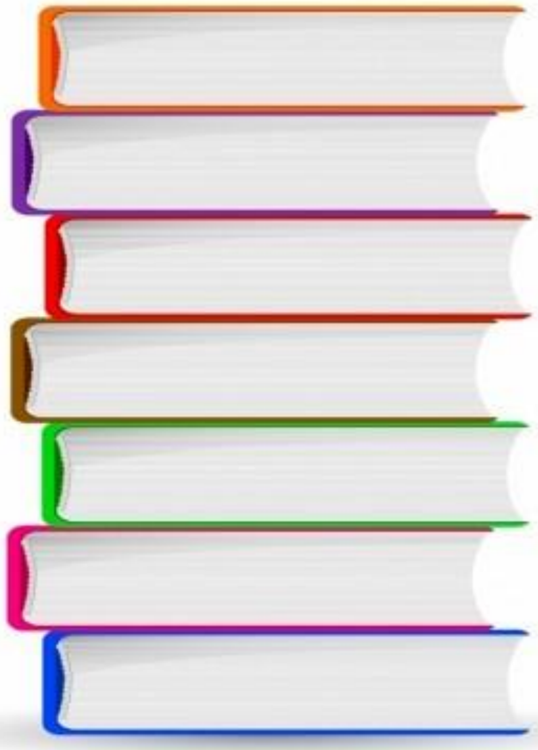
The parameter must be a valid storyboard identifier

Individual Exercise

Add a UINavigationController to a storyboard

Summary

1. Create a Navigation Controller programmatically
2. Utilize the designer to create a Navigation Controller
3. Customize the Navigation Controller



Show different views of related data with tab navigation

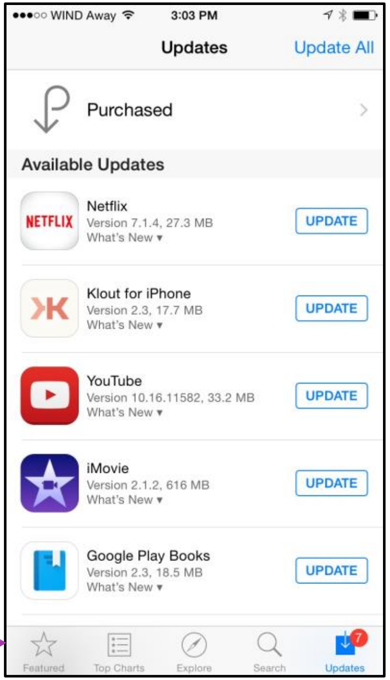
Tasks

1. Create a Tab Bar Controller
2. Populate a Tab Bar Controller
3. Customize the Tab Bar Controller

Tab Navigation

- ❖ Tab navigation allows users to switch between view controllers by selecting tabs displayed at the bottom of the screen
  - § Ideal for 3-5* screens of equal importance
  - § Allows for quick switching without interference

The active page's tab is highlighted using a highlight color


●●●○○ WIND Away 3:03 PM

Updates | Update All

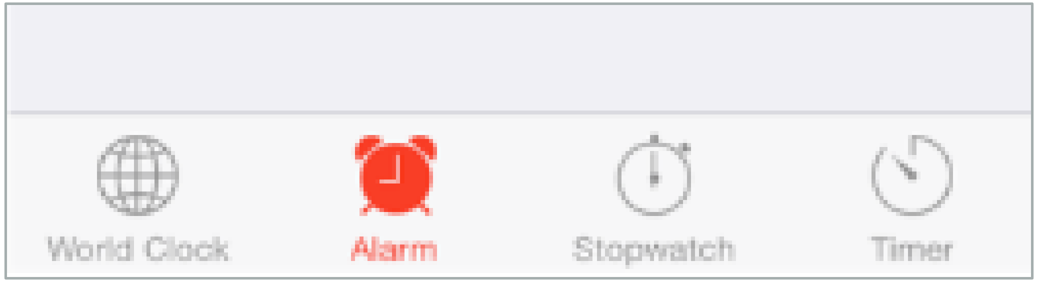
Purchased

Available Updates

Netflix	Version 7.1.4, 27.3 MB / What's New ▾	UPDATE
Klout for iPhone	Version 2.3, 17.7 MB / What's New ▾	UPDATE
YouTube	Version 10.16.11582, 33.2 MB / What's New ▾	UPDATE
iMovie	Version 2.1.2, 616 MB / What's New ▾	UPDATE
Google Play Books	Version 2.3, 18.5 MB / What's New ▾	UPDATE

What is a UITabBarController?

❖ iOS implements tab navigation with the `UITabBarController`



Displays tabs at the bottom of the screen which can show an icon and a label

A `UITabBarController` can hold a `UINavigationController` within a tab

UITabBarItem

- A **`UITabBarItem`** is an object used to describe the appearance of a single tab within a Tab Bar Controller

```
var tbi = new UITabBarItem("Clock", UIImage.FromBundle("clock.png"), 0);
```

`UITabBarItem` constructor takes a title, an image, and an integer "tag" which can be used to identify the item later

Using the UITabBarItem

- Every View Controller has a **`TabBarItem`** property which can be set to an instance of a **`UITabBarItem`**

```
var tabViewController = new ClockViewController();
var tbi = new UITabBarItem("Clock", UIImage.FromBundle("clock.png"), 0);
tabViewController.TabBarItem = tbi;
```

Setting the **`TabBarItem`** property will determine the title and image shown on a tab for this view controller when it's added to a tab bar controller

UITabBarController programmatically

❖ Add children to tab controller using the **`ViewControllers`** property

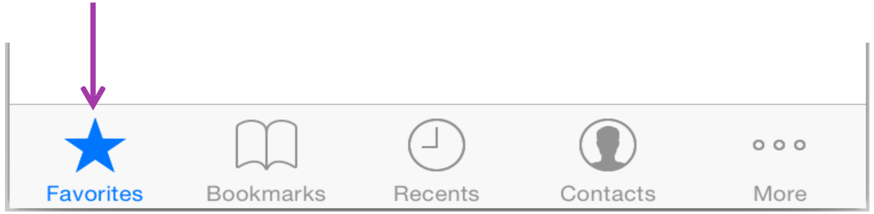
```
public class MyTabBarController : UITabBarController
{
   public override void ViewDidLoad()
   {
      base.ViewDidLoad();

      var vc1 = new ClockViewController {
         TabBarItem = new UITabBarItem("Clock",
                            UIImage.FromBundle("clock"), 0);
      };
      ...
      this.ViewControllers = new UIViewController[] { vc1, vc2, vc3
      };
   }
}
```

The **`ViewControllers`** property is set to an array of **`UIViewControllers`**

TabBar Images

- Images can be set on the tabs; resource images or system image can be displayed

Image base size should be at least 25x25 (32x32 is ideal)



Prefer monochromatic, transparent images for template (stencil) filtering

Always include **@2x** (64x64) and **@3x** (96x96) images for high-resolution devices

UITabBarSystemItem

❖ `UITabBarSystemItem` provides a small selection of built-in images that can be used to decorate the tabs

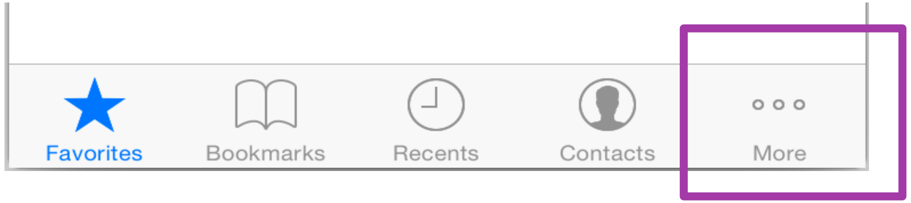
Each item has a set title that cannot be changed

```
var tab1 = new FavoritesViewController ();

 tab1.TabBarItem = new UITabBarItem (UITabBarSystemItem.Favorites, 0);
```

`UITabBarSystemItem`s can also be set from the property panel in the designer

Overflowing the UITabBarController

- The **`UITabBarController`** can show up to 5 tabs on the iPhone and 8 tabs on the iPad or iPhone 6+; if more tabs are added then the system creates a "more" tab and displays the remainder in a system-provided Table View



Overflow (more) tab

Detecting Tab Selection

- ❖ To respond to selection events on the **`UITabBarController`**, subscribe to the **`ViewControllerSelected`** event handler

```
public class ClockTabBarController : UITabBarController
{
    public override void ViewDidLoad()
    {
        ...
        this.ViewControllerSelected += TabSelected;
    }
    ...
}
```

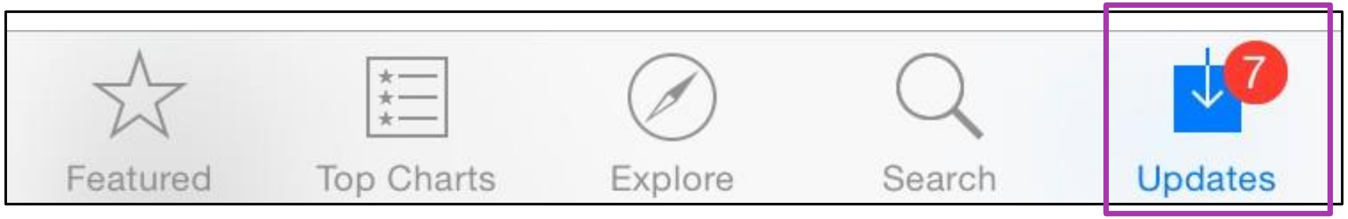
TabBar Selected Item

- To determine which tab is selected, use `TabBar.SelectedItem` property

```
public class ClockTabBarController : UITabBarController
{
    ...
    void TabSelected (object sender, UITabBarSelectionEventArgs e)
    {
        var alert = new UIAlertView("Tab tapped",
            this.TabBar.SelectedItem.Title, null, "OK", null);
        alert.Show();
    }
    ...
}
```

Tab Badges

- A *Badge* can be added to a tab to display a small amount of text by setting the `BadgeValue` property on a `UITabBarItem`

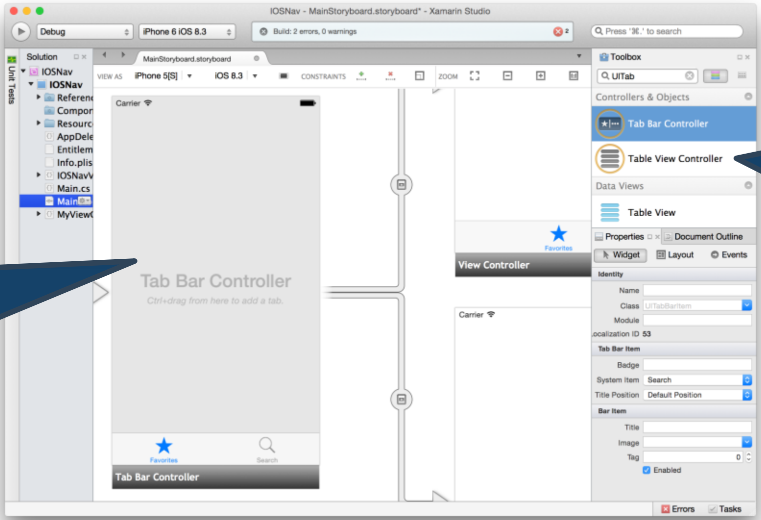


```
updatesVC.TabBarItem.BadgeValue = "7";
```

```
updatesVC.TabBarItem.BadgeValue = null;
```

Add a Tab Bar Controller to a Storyboard

- The Xamarin iOS designer can be used to add a **`UITabBarController`** to a Storyboard

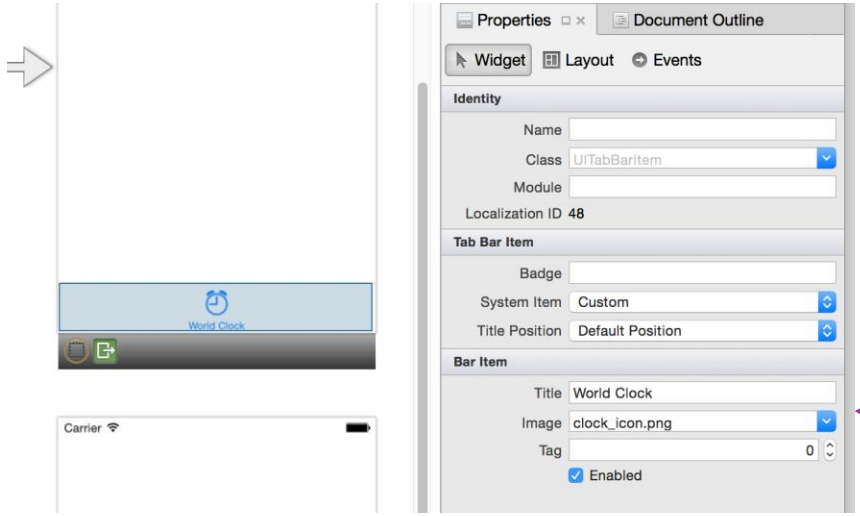


Provides a `UITabBarController` and two child `UIViewControllers`

Use the Toolbox and search for TabBar Controller

Customizing the Tabs from the Designer

- The designer will show additional UI and properties when a child view controller is connected to a `UITabBarController` via a Segue



Identity

Name	
Class	UITabBarItem
Module	
Localization ID	48

Tab Bar Item

Badge	
System Item	Custom
Title Position	Default Position

Bar Item

Title	World Clock
Image	clock_icon.png
Tag	0
✓ Enabled	

Set the title and tab bar image

Individual Exercise

Add a UITabBarController to a Storyboard

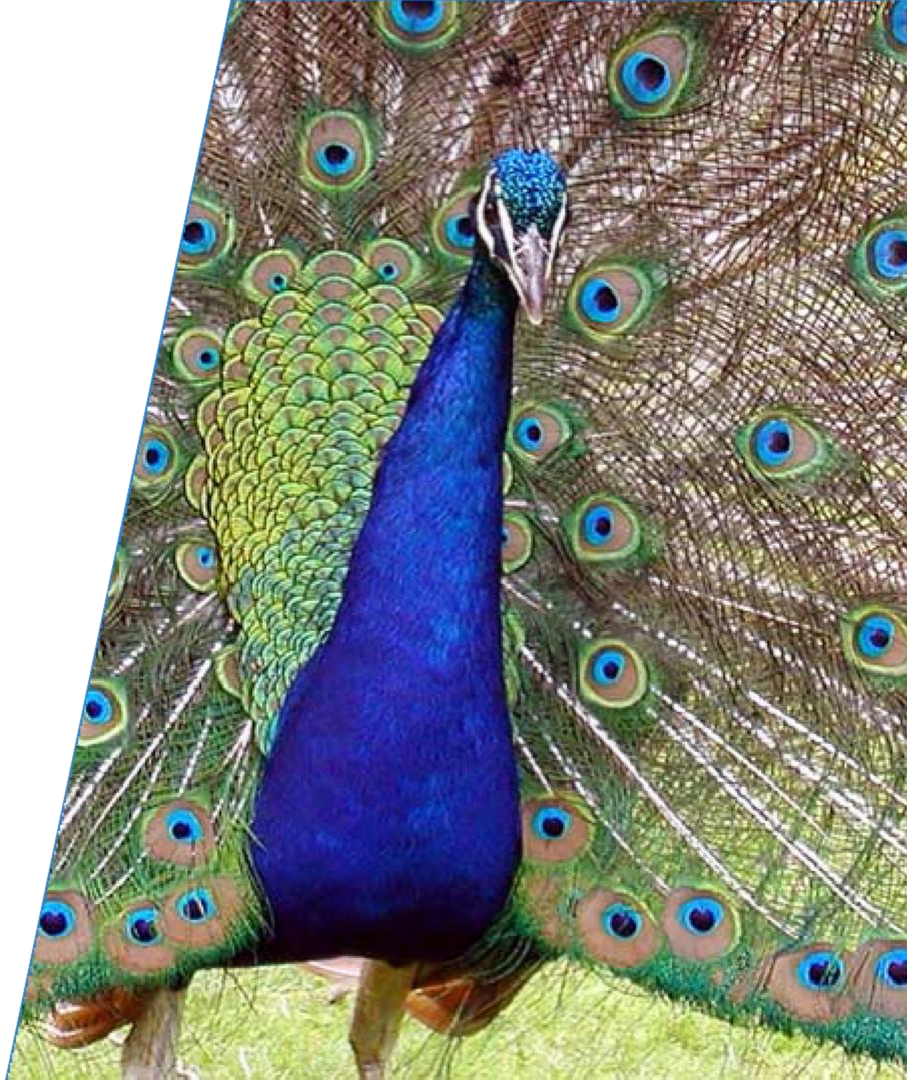
Summary

1. Create a Tab Bar Controller
2. Populate a Tab Bar Controller
3. Customize the Tab Bar Controller

Display hierarchical relationships with master/detail navigation

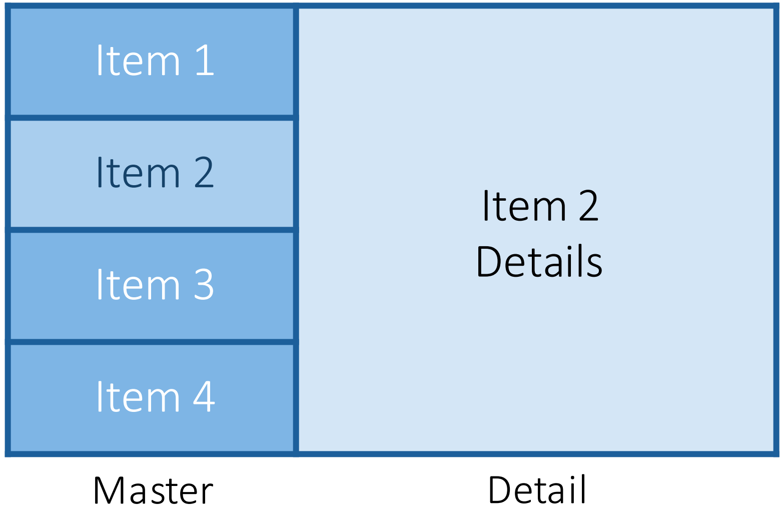
Tasks

1. Working with the Split View Controller
2. Using a Split View Controller in code
3. Using the iOS Designer to define a Split View Controller



Master/Detail Navigation

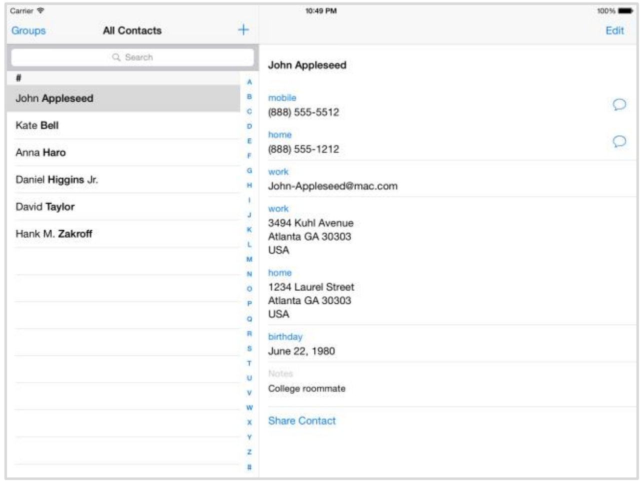
- A Master/Detail navigation pattern displays a “Master” list used for primary navigation along side a second visual area displaying the “Details” for the currently selected item



What is the UISplitViewController?

❖ The **`UISplitViewController`** class manages the display of two side-by-side view controllers

Left displays a list of items for navigation (Master)



John Appleseed
Kate Bell
Anna Haro
Daniel Higgins Jr.
David Taylor
Hank M. Zakroff

John Appleseed

mobile
(888) 555-5512

home
(888) 555-1212

work
John-Appleseed@mac.com

work
3494 Kuhl Avenue
Atlanta GA 30303
USA

home
1234 Laurel Street
Atlanta GA 30303
USA

birthday
June 22, 1980

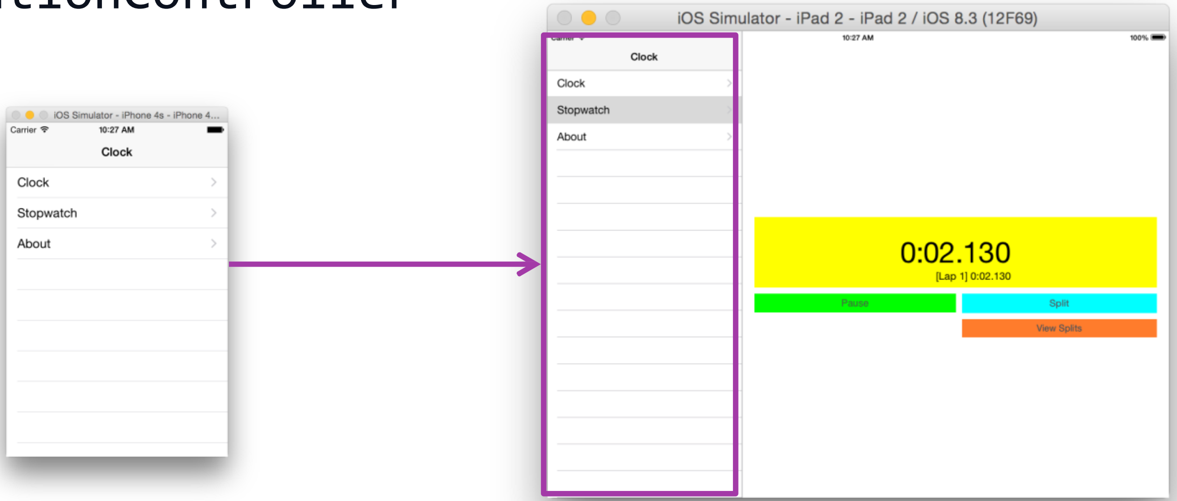
Notes
College roommate

Share Contact

Right displays details about the selected item (Detail)

Master/Detail Responsive design

- ❖ When the Split View Controller is used on smaller displays (iPhone), it "collapses" so only one View Controller is shown, effectively mimicking a `UINavigationController`



Navigation on tablets is limited to two levels. Best practice is to limit to two levels for consistent behavior on phone and tablet.

Creating a SplitView Controller

- When creating a **UISplitViewController** programmatically, the master/detail views are assigned to the **ViewControllers** property

```
public class EventSplitViewController : UISplitViewController
{
    MasterViewController masterView;
    DetailViewController detailView;

    public EventSplitViewController() : base()
    {
        masterView = new MasterViewController();
        detailView = new DetailViewController();

        ViewControllers = new UIViewController[] { masterView, detailView
        };
    }
}
```

Navigating programmatically in iOS8

- In iOS8, there is a new adaptive method **`ShowViewController`**, defined on the **`UIViewController`** which is used for forward navigation

```
public void ShowViewController(
    UIViewController controller, NSObject sender)
```

Sets the master view, or current navigation view, or modal view

Navigating programmatically in iOS8

- The `ShowDetailViewController` method is used to update the detail view in a Split View Controller

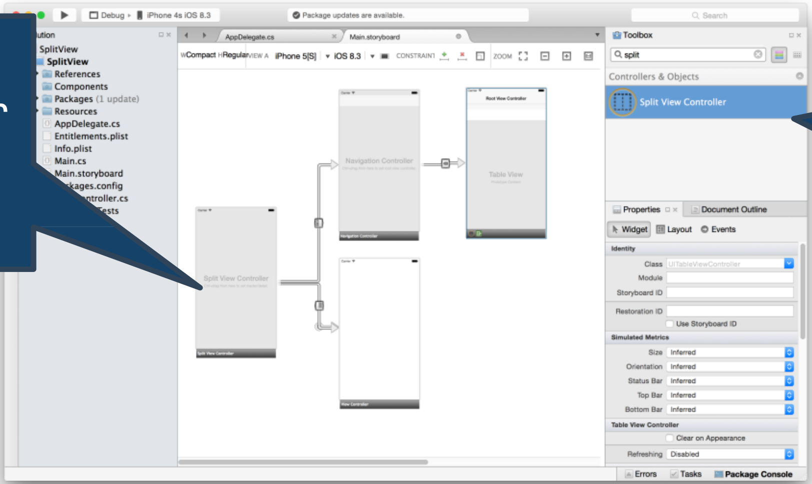
```
public void ShowDetailViewController(
     UIViewController controller, NSObject sender)
```

Replaces the detail view (right side of a split view)

Add a Split View Controller (designer)

- The Xamarin iOS designer can be used to add a `UISplitViewController` to a Storyboard

Provides a `UISplitViewController` and three child view controllers

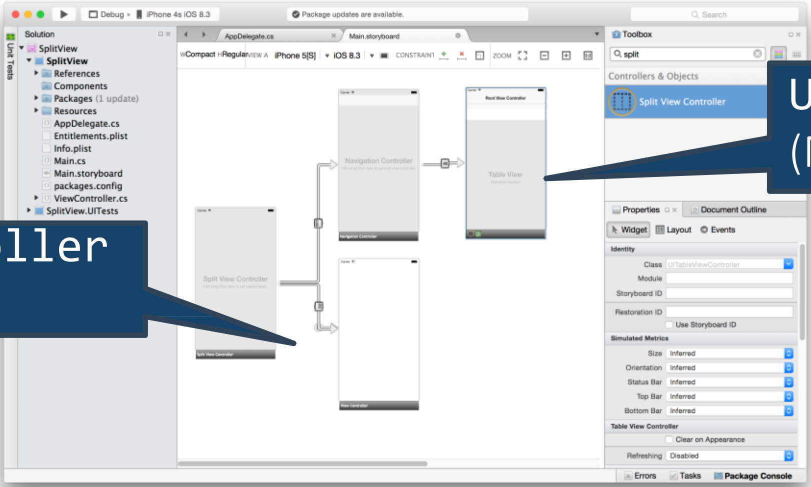


Identity
Class UITableViewController
Module
Storyboard ID
Restoration ID
Use Storyboard ID
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
Table View Controller
Clear on Appearance
Refreshing Disabled

Use the Toolbox and search for Split View Controller

Master/Detail View Controllers

- The Xamarin iOS designer provides a **`UITableViewController`** within a **`UINavigationController`** for the Master UI, and a simple **`UIViewController`** for the details UI



Identity
Class: UITableViewController
Module:
Storyboard ID:
Restoration ID:
☐ Use Storyboard ID
Simulated Metrics
Size: Inferred
Orientation: Inferred
Status Bar: Inferred
Top Bar: Inferred
Bottom Bar: Inferred
Table View Controller
☐ Clear on Appearance
Refreshing: Disabled

`UITableViewController`
(Master)

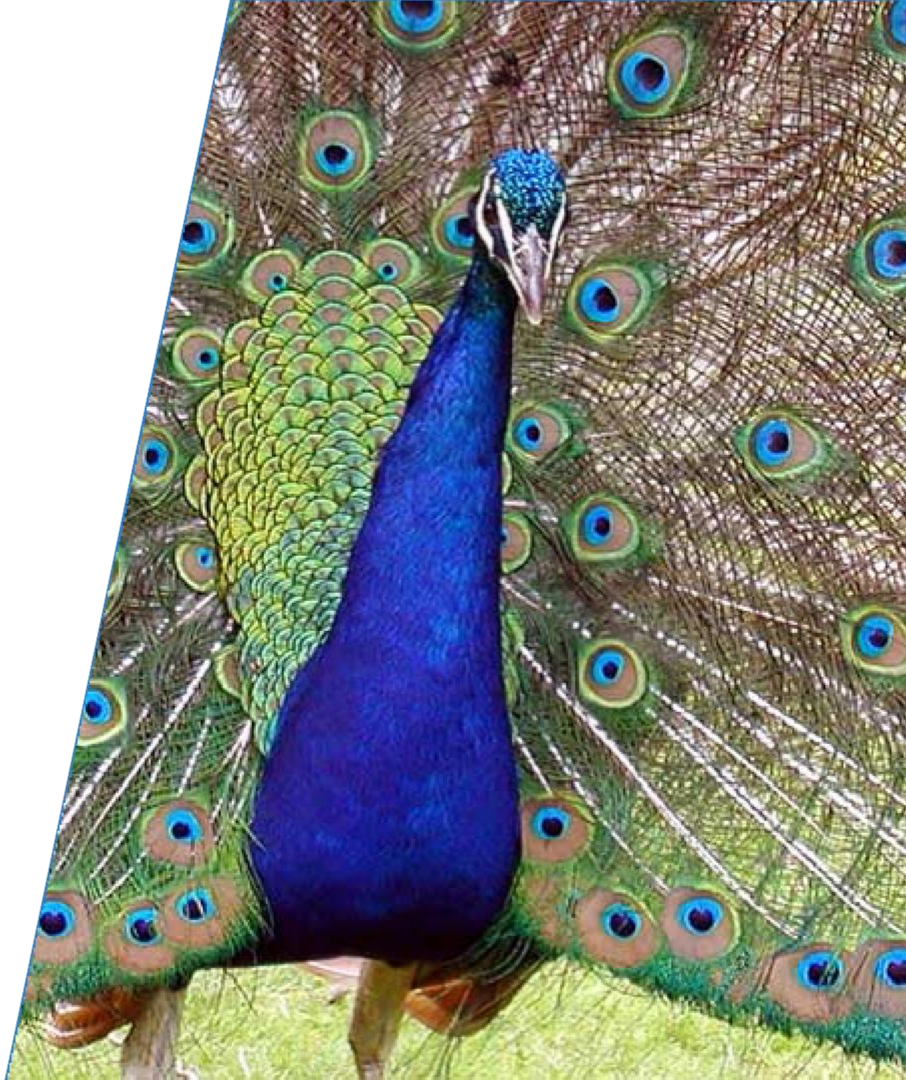
`UIViewController`
(Detail)

Individual Exercise

Add a UISplitViewController to a Storyboard

Summary

1. Working with the Split View Controller
2. Using a Split View Controller in code
3. Using the iOS Designer to define a Split View Controller



Organize pages of information with page navigation

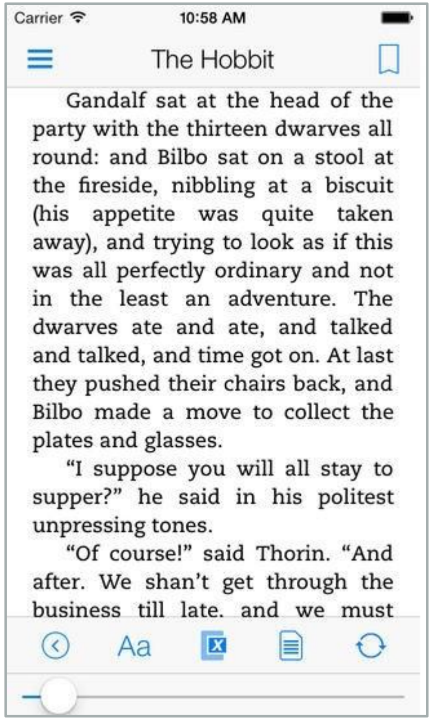
Tasks

1. Displaying pages of data
2. Populating pages
3. Using the designer
4. Page-based app template

Displaying pages of data

- iOS supports paging through related views of data

- Can be used to step the user through a sequence of data where they should consume the information in a progressive fashion



Gandalf sat at the head of the party with the thirteen dwarves all round: and Bilbo sat on a stool at the fireside, nibbling at a biscuit (his appetite was quite taken away), and trying to look as if this was all perfectly ordinary and not in the least an adventure. The dwarves ate and ate, and talked and talked, and time got on. At last they pushed their chairs back, and Bilbo made a move to collect the plates and glasses.

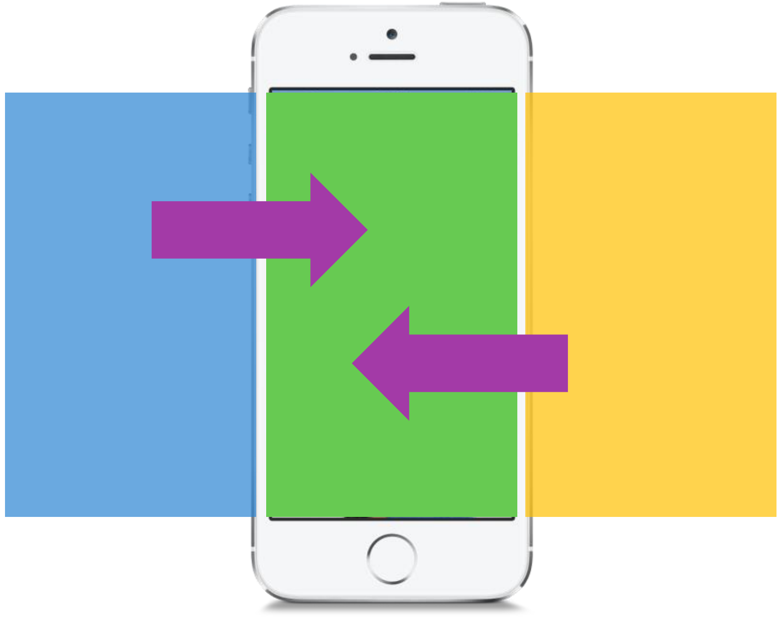
"I suppose you will all stay to supper?" he said in his politest unpressing tones.

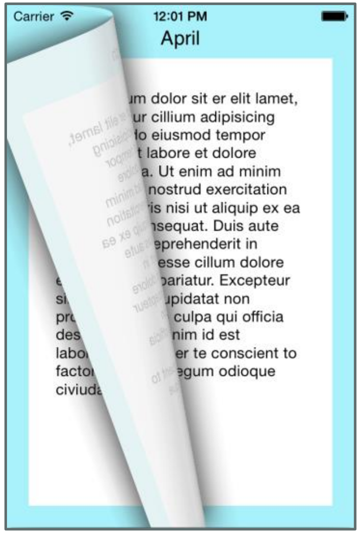
"Of course!" said Thorin. "And after. We shan't get through the business till late. and we must

Book and magazine applications use this technique to display "pages" of data

Page-based navigation

❖ The **`UIPageViewController`** displays child view controllers as "pages"





Supports various animations including a "page-turn" animation

Using UIPageViewController

❖ The **UIPageViewController** is used for page navigation and displays child view controllers to represent each page

```
var pageViewController = new UIPageViewController (
        UIPageViewControllerTransitionStyle.PageCurl,
        UIPageViewControllerNavigationOrientation.Horizontal,
        UIPageViewControllerSpineLocation.Mid);
...
AddChildViewController(pageViewController);
View.AddSubview(pageViewController.View);
```

Transition animation, scrolling direction and how pages are split is controlled by constructor arguments

Populating the Pages

❖ Pages are populated through a data source class which holds child view controllers – each child is a single page

```
UIPageViewController pageViewController;
ModelController dataSource;

public override void ViewDidLoad()
{
    ...
    pageViewController.WeakDataSource = dataSource;
    ...
}
```

The Page View Data Source

- Page controller calls methods on the data source to retrieve pages asthe user navigates through the data forward and backward

```
public class ModelController :UIPageViewControllerDataSource
{
   ...
   public override UIViewController GetNextViewController (
                 UIPageViewController pageViewController,
                 UIViewController referenceViewController ) { ... }

   public override UIViewController GetPreviousViewController (
                 UIPageViewController pageViewController,
                 UIViewController referenceViewController ) { ... }
}
```

Omitting the data source class

- Xamarin.iOS adds convenience delegate properties to support page retrieval from an arbitrary class

```
UIPageViewController pageViewController;
UIViewController[] pages { get; set; }
...

pageViewController.GetPreviousViewController = (pvc, vc) => {
    int page = Array.IndexOf(pages, vc);
    return page == 0 ? null : pages[page - 1];
}
```

Delegate signatures match the methods on the data source class

Provide the first page of data

❖ By default, the page controller does not show the first page – must supply the view controller(s) for the first view

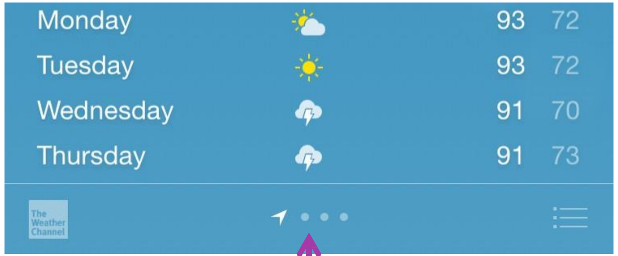
```
UIPageViewController pageViewController;
ModelController dataSource;
public override void ViewDidLoad()
{
    ...
    pageViewController.SetViewControllers(
        new [] { dataSource.Pages[0] },
        UIPageViewControllerNavigationDirection.Forward,
        false, null);
    ...
```

Parameters indicate the initial direction and whether it will animate in

If you use the `PageCurl` transition with a mid-page spine display, then you must provide two pages initially since it will always show two pages at a time

Supporting random access

- Page controller can supply a quick navigation UI index (`UIPageControl`) to jump between pages when it knows the number of pages available



Monday	93	72
Tuesday	93	72
Wednesday	91	70
Thursday	91	73

Index is useful if the number of pages is constrained or the information is not tightly related

Adding a page index

- Transition style must be set to scroll and the navigation orientation must be horizontal to display pager control

`UIPageControl` is added to bottom of page view

```
var pageViewController = new UIPageViewController (
       UIPageViewControllerTransitionStyle.Scroll,
       UIPageViewControllerNavigationOrientation.Horizontal,
       UIPageViewControllerSpineLocation.Min);
```

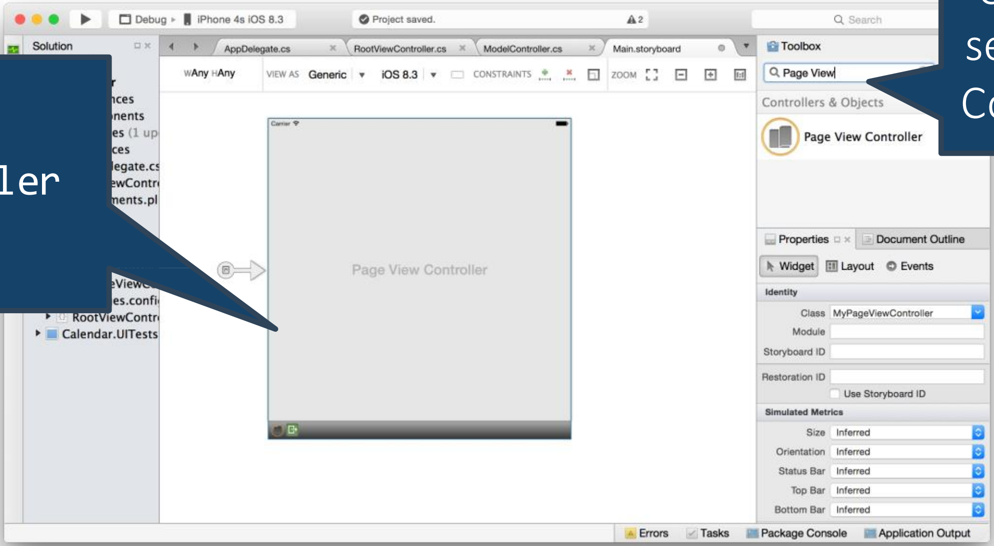
Adding a page index

- Must override two methods to provide the number of pages and starting page number to provide quick jump index

```
public class ModelController : UIPageViewControllerDataSource
{
   public UIViewController[] Pages { get; set; }
   public override nint GetPresentationCount(UIPageViewController pvc) {
      return Pages.Length;
   }

   public override nint GetPresentationIndex(UIPageViewController pvc) {
      return 0; // starting page#
   }
}
```

Add a Page View Controller (designer)

❖ The Xamarin iOS designer can be used to add a **`UIPageViewController`** to a Storyboard

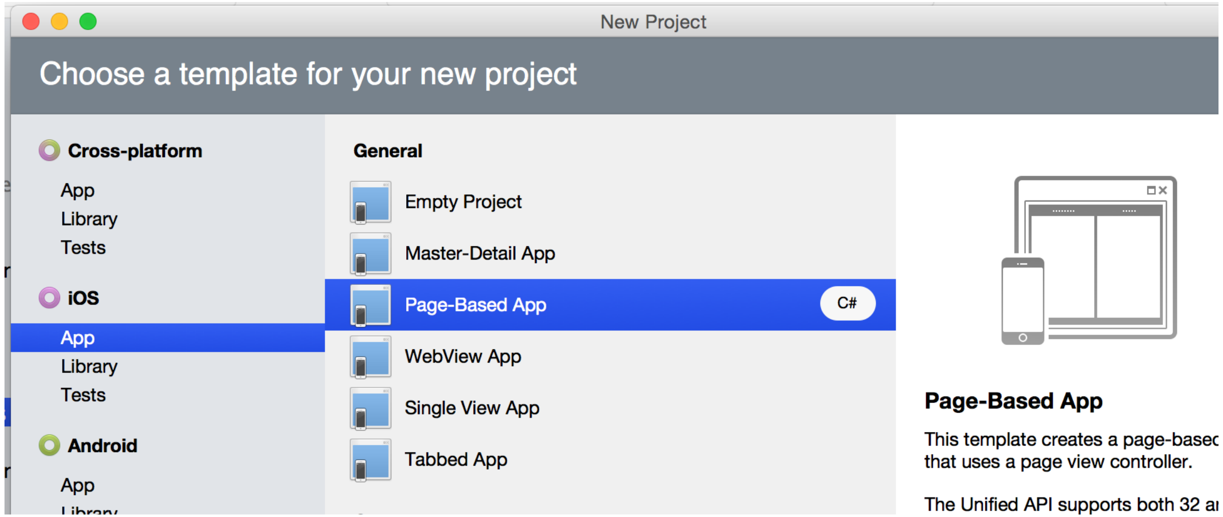


Provides a `UIPageViewController` without children

Use the Toolbox and search for Page View Controller

Page View Controller template

- There is a Page-Based App template that creates a project with a populated Page View Controller



Page-Based App

This template creates a page-basec that uses a page view controller.

The Unified API supports both 32 ar

Thank You!