iOS App Development

Juha-Matti Liukkonen

Mar 6, 2012

Contents

- Introduction to the iOS Platform
- iOS Concepts for SW Designers
- App Elements

- Creating and Deploying an App, step-by-step
- Advanced programming topics
- Distributing your App

Extensive documentation available at:

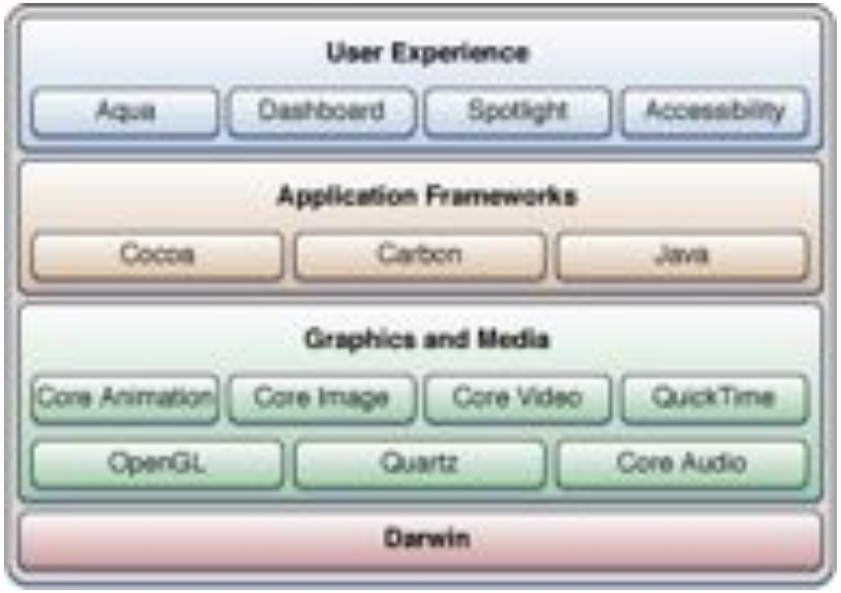
http://developer.apple.com

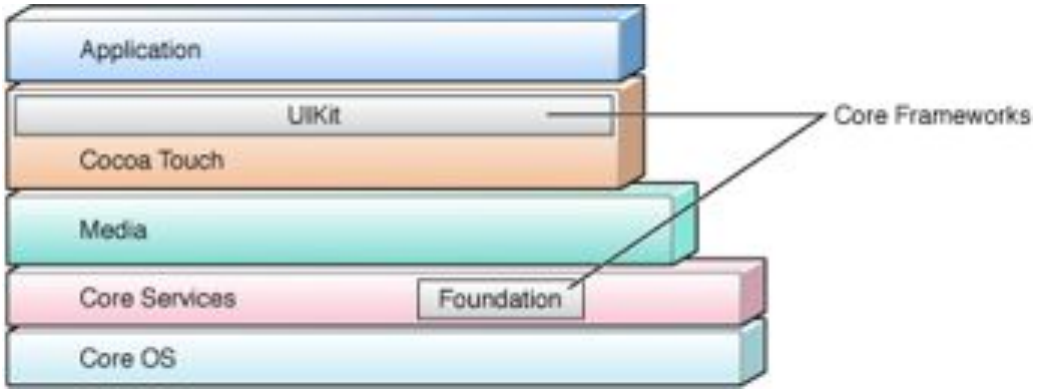
Introduction to iOS

iOS Basics

- iOS is a low-footprint adaptation of Apple's Mac OS X
  - Mach microkernel
  - BSD unix derived libraries and userland
  - Objective C runtime
  - Used in iPod Touch, iPhone, iPad, Apple TV
- Touch based user interface
  - Revolutionized the industry in 2007
  - Mac's AppKit replaced with touch-centric UIKit layer
  - Unix layers below the proprietary UIKit are mostly open source
- Software developed using Objective C/C++
  - Initially Apple attempted only Web apps, but that didn't work…
  - Code runs native, effective use of hardware resources

iOS Architecture





The full Mac OS X has a wide array of features, as can be expected from a desktop OS. There are three different app frameworks, for apps based on three different technologies.

The Core OS is shared with the full Mac OS. Core Services is a subset of full Mac OS. Apps are written to use Cocoa Touch (UIKit) and Media services layers.

iOS APIs and compatibility

- Apple has a strict 2-major-versions policy
  - Older software versions are deprecated quickly
  - Software upgrades are free and easy to do -> most users run the latest version
  - Currently iOS 4 and 5 supported
  - Only the oldest devices (iPhone, iPod Touch 1st gen) stuck with iOS 3
- iOS 2 introduced the App Store (iOS 1 was Web apps only)
- iOS 3 is single-tasking, but introduced lots of new (even basic) features
- iOS 4 brought multi-tasking, FaceTime, iBooks, iAd, UI tweaks
- iOS 5 = current version
  - Reworked notification system, Twitter integration
  - iCloud, “PC-free” management over WiFi

iOS variance

<symbio>

- Android suffers from device variance… iOS is not immune, either

	iPod Touch	iPhone 4	iPhone 4S	iPad	iPad 2
CPU	800 MHz armv7	1 GHz armv7	2x1 GHz armv7	1 GHz armv7	2x1 GHz armv7
RAM	256 MB	512 MB	512 MB	256 MB	512 MB
Screen	960x640	960x640	960x640	1024x768	1024x768
Cameras	0.7 MP + VGA front	5 MP + VGA front	8 MP + VGA front	-	0.7 MP + VGA front
Sensors	Gyro	Compass, Accel, Gyro	Compass, Accel, Gyro	Compass, Accel	Compass, Accel, Gyro
GPS	-	Yes	Yes + GLONASS	In 3G models	In 3G models

Note: only models currently in production.

iOS UI basics

The App-centric user interface is iconic. No widgets or distractions, just a quick launch shortcut bar at the bottom, and one hardware button to press for home.

User interface elements are large and easy to use with a finger.

Text input is performed with an on-screen virtual keyboard.

<symbio>

iOS Concepts
for SW Designers

App model for mobile world

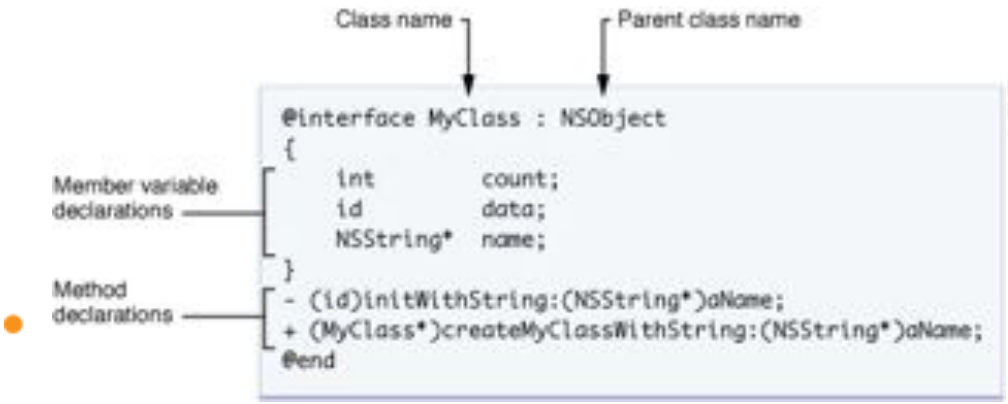
<symbio>

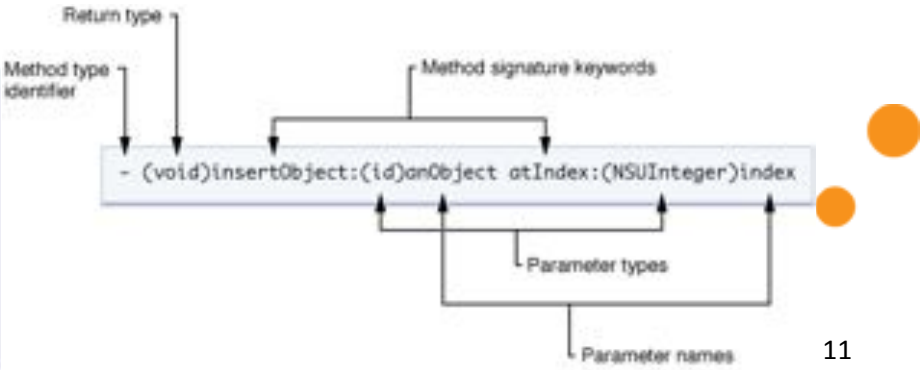
- Application model inherited from Mac OS X
  - Traditional C *main()* to instantiate UI event loop
  - Strict Model-View-Controller hierarchy
  - Data Model objects manage data content
  - View Controller objects perform view setup and most reactive actions
  - View objects manage on-screen objects
- "Quick launch, short use"
  - UI and the event loop are set up as first things (generated code)
  - Usually View Controller unserializes views from a .nib file (generated XML)
- All code is native
  - Objective C/C++, with active reference counting for memory management

Key for mobile app: save power.
Do work only when you have to.

Objective C/C++

- A cross between Smalltalk and C/C++
  - Some say, “object oriented C done right”
  - Syntactically quite different from C++, conceptually not so much
  - Significant use of pre-processor directives
  - Allows for run-time binding of objects
- Apple’s Foundation and toolkit libraries rely heavily on Objective C features
  - Original design from NextStep circa 1982
  - Evolved into current Mac OS X circa 1998
  - Proven, flexible and very performant architecture





Objective C/C++ sample


```
@implementation MyClass

- (id)initWithString:(NSString *)aName
{
    self = [super init];
    if (self) {
        name = [aName copy];
    }
    return self;
}

+ (MyClass *)createMyClassWithString: (NSString *)aName
{
    return [[[self alloc] initWithString:aName] autorelease];
}
@end
```

Objective-C implements reference counting and garbage collection via the *autoreleasepool*.

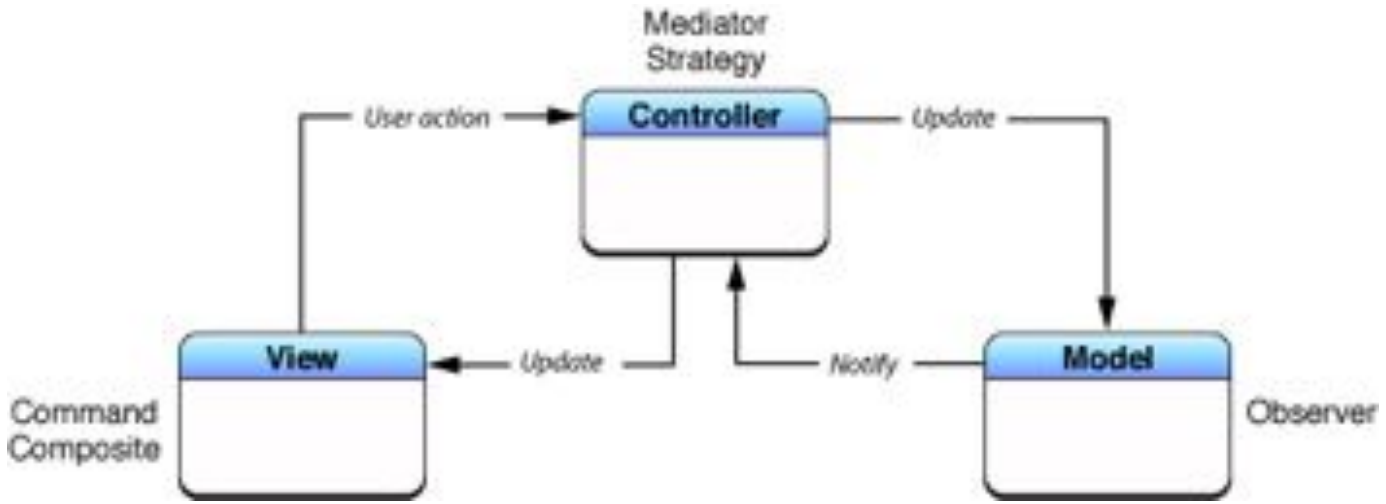
```
id MyClass::initWithString(NSString *aName)
{
    this = ::init();
    if (this) {
        name = aName->copy();
    }
    return this;
}

static MyClass* MyClass::createMyClassWithString(NSString *aName)
{
    return (new MyClass())->initWithString(aName);
}
```

Notice how the code translates *syntactically* – but semantically the result is not the same.

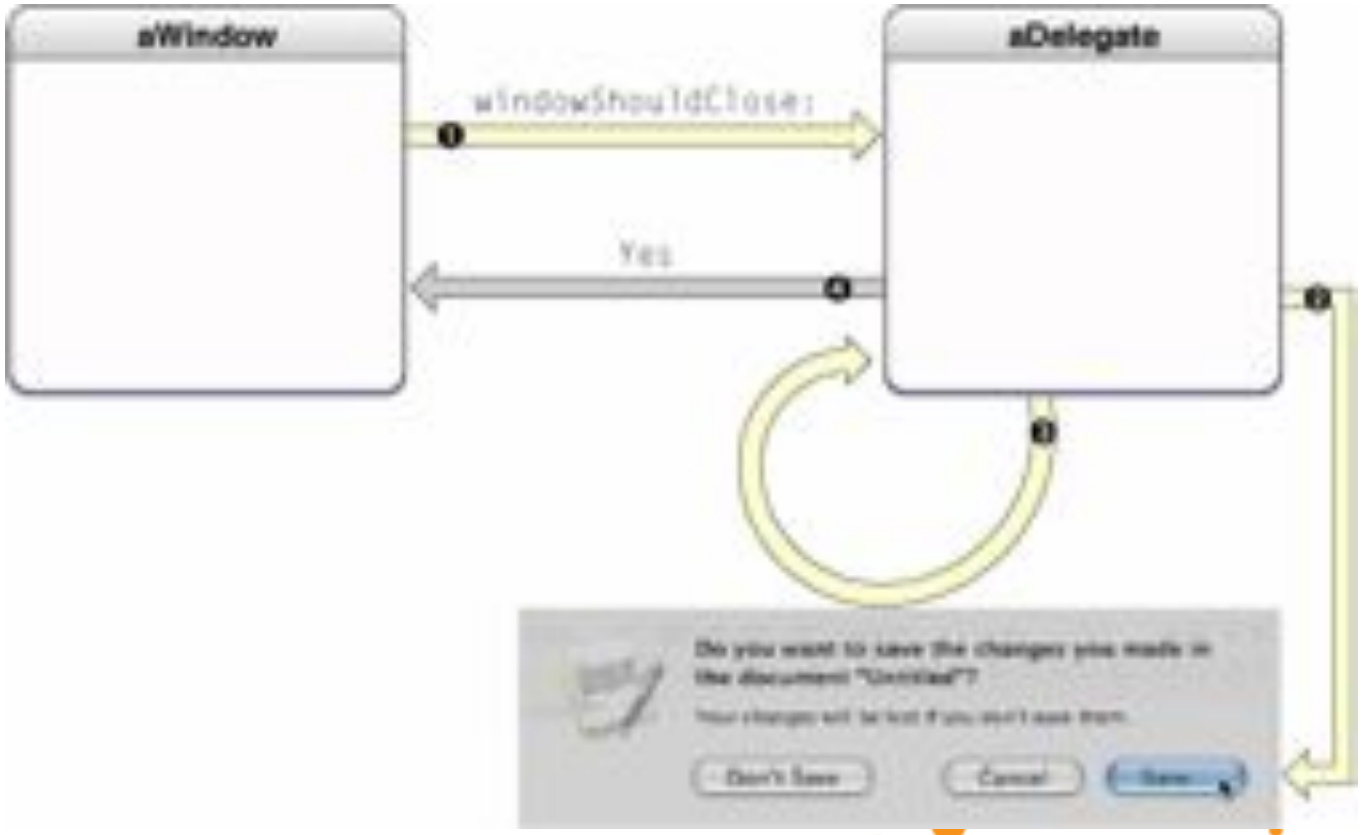
The Model-View-Controller pattern

- Strict separation of concerns
  - Model – data
  - View – display
  - Controller – mediates between the two, understanding application state

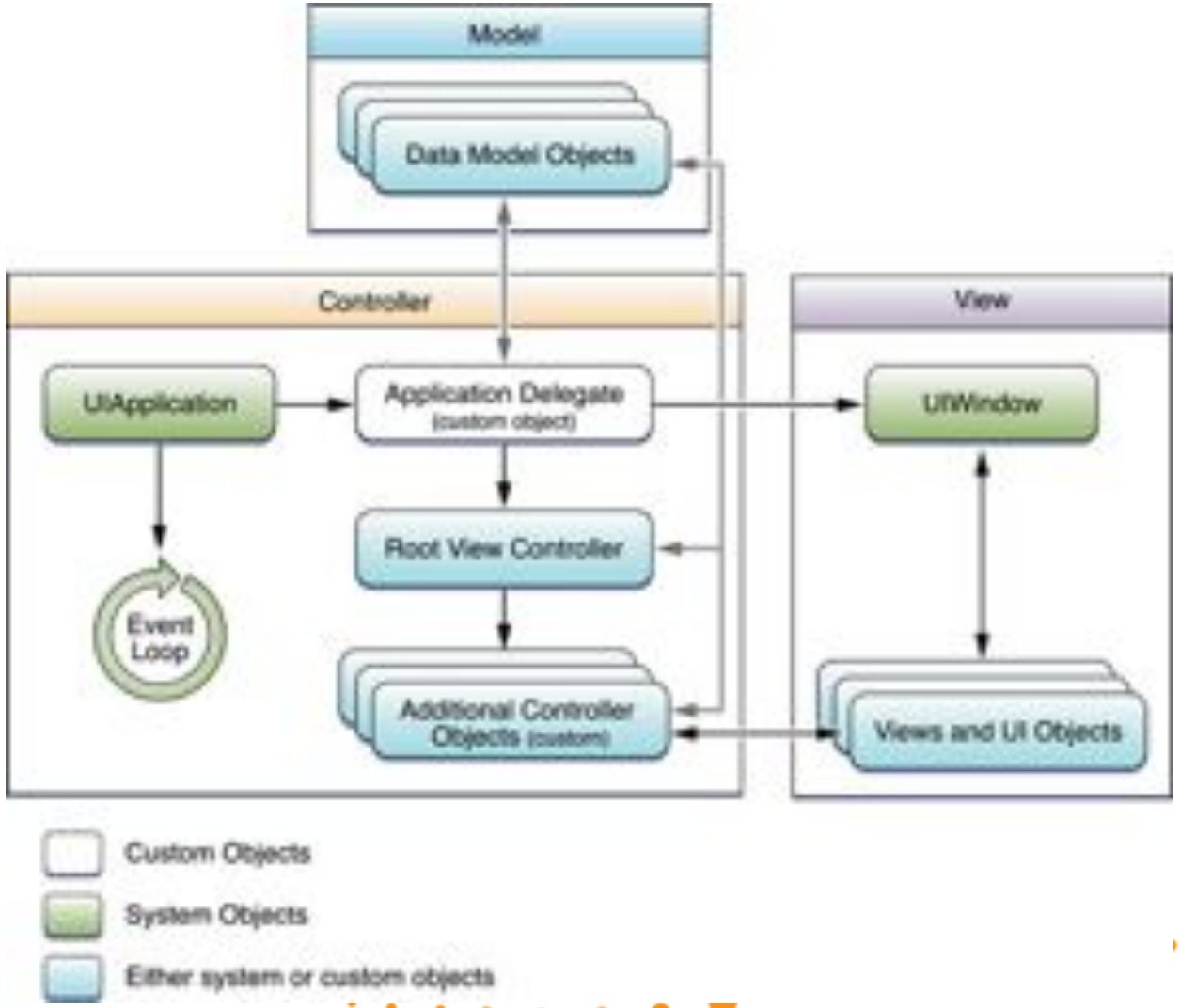


The Delegate pattern

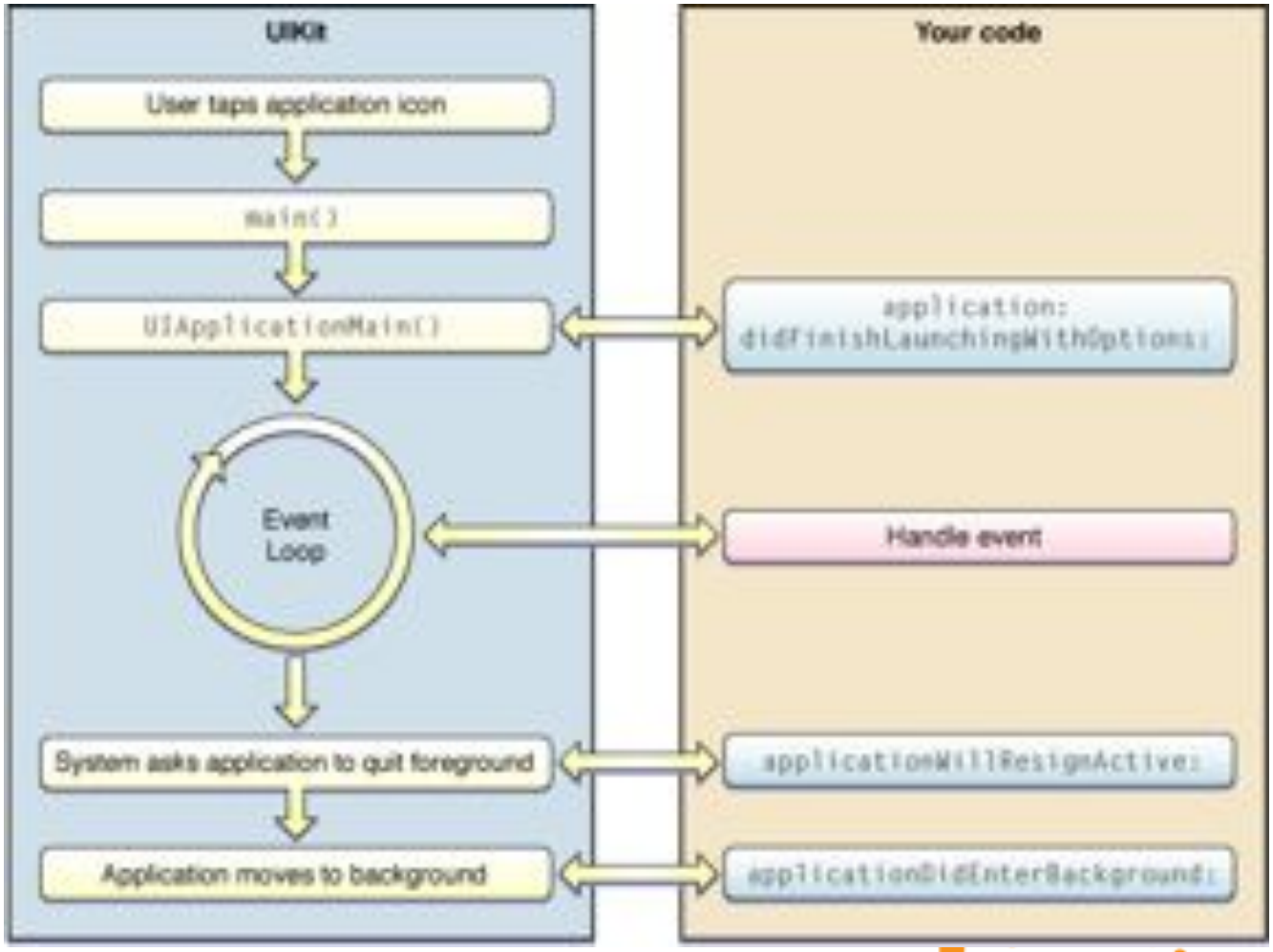
- Delegate objects handle app-specific logic
  - In C++ or Java, you create specialized subclasses – in Objective C, you delegate



iOS app logical structure



iOS app life cycle



XCode and Interface Builder

<symbio>

- IDE = Integrated Development Environment
  - XCode is Apple's IDE for iOS and Mac OS development
  - Code completion, online help, debugger, deployment to target device, …
  - Interface Builder for building the UI
  - Instruments for profiling and analyzing
  - iPhone/iPad emulator for testing
- Also used to manage developer identities
  - Must have a developer identity to deploy apps to real devices
  - Developer identity obtained from Apple Store, 67 € / year
- XCode is free for everyone
  - Xcode 4 required for iOS 5 development – free download from App Store
  - Xcode 4 requires OS X 10.7 (Lion)

Getting ready to develop

<symbio>

- Follow instructions in http://developer.apple.com/
  - Download XCode 3 or 4, your preference (XCode 4 needed for iOS 5 features!)
  - Run the installer
  - Optionally: run XCode and enter your developer information
    - Required to deploy apps to real devices
    - Real devices are recognized when they are plugged in
- The download is 4 GB (Xcode 3) or 1,5 GB (XCode 4 from App Store)
  - Installation takes another 30..45 minutes but is generally painless

Step 1: Get started
Step 2: …
Step 3: Victory!

App Elements

The Info.plist file

<symbio>

- The Info.plist defines your application
  - Display name of your application
  - Executable name of your application
  - Device environment that your application may require (iPhone/iPad)
  - Main .nib file name to load your UI from (created by Interface Builder)
  - UIRequiredDeviceCapabilities to require GPS, Camera, etc.
    - Cannot request all differentiating factors, some need to be checked in code
- Important, but much simpler than an Android Manifest
  - Rarely need to touch the XCode generated file

Resources

- Stored in subdirectory *Resources* in your application source tree
  - Drawables: icons, bitmaps
  - Layouts and menus: XML view definitions (the .nib and .xib files)
  - Strings: a .strings plain text file mapping logical names to values
- Localization
  - Get Info on a resource file in XCode, click Make Localizable
  - Generates a copy you can localize, including UI layouts and graphics
    - Localization should be done only after your UI layouts are final
    - Command line tool *ibtool* can be used to synchronize updates
  - Results in *Language1.lproj*, *Language2.lproj* subdirectories in your app bundle
  - System will load resources from current language *.lproj* – or use the development language defined in your Info.plist if the localization is not found

Source files

- The .h files for declarations, .m files for code
  - Forward declarations using @class *name*
  - Interface declarations using @interface *name* … @end
  - Class member accessor declarations using @property (…) *name*
  - Headers included using traditional #include "*filename*"
  - Implementation using @implementation *name* … @end
  - Class member accessor implementations using @synthesize *name*
- AppDelegate implements life cycle methods
  - Including instantiating your initial view
- ViewController binds your views' UI elements to your data and logic
  - Implements the behavior of your app

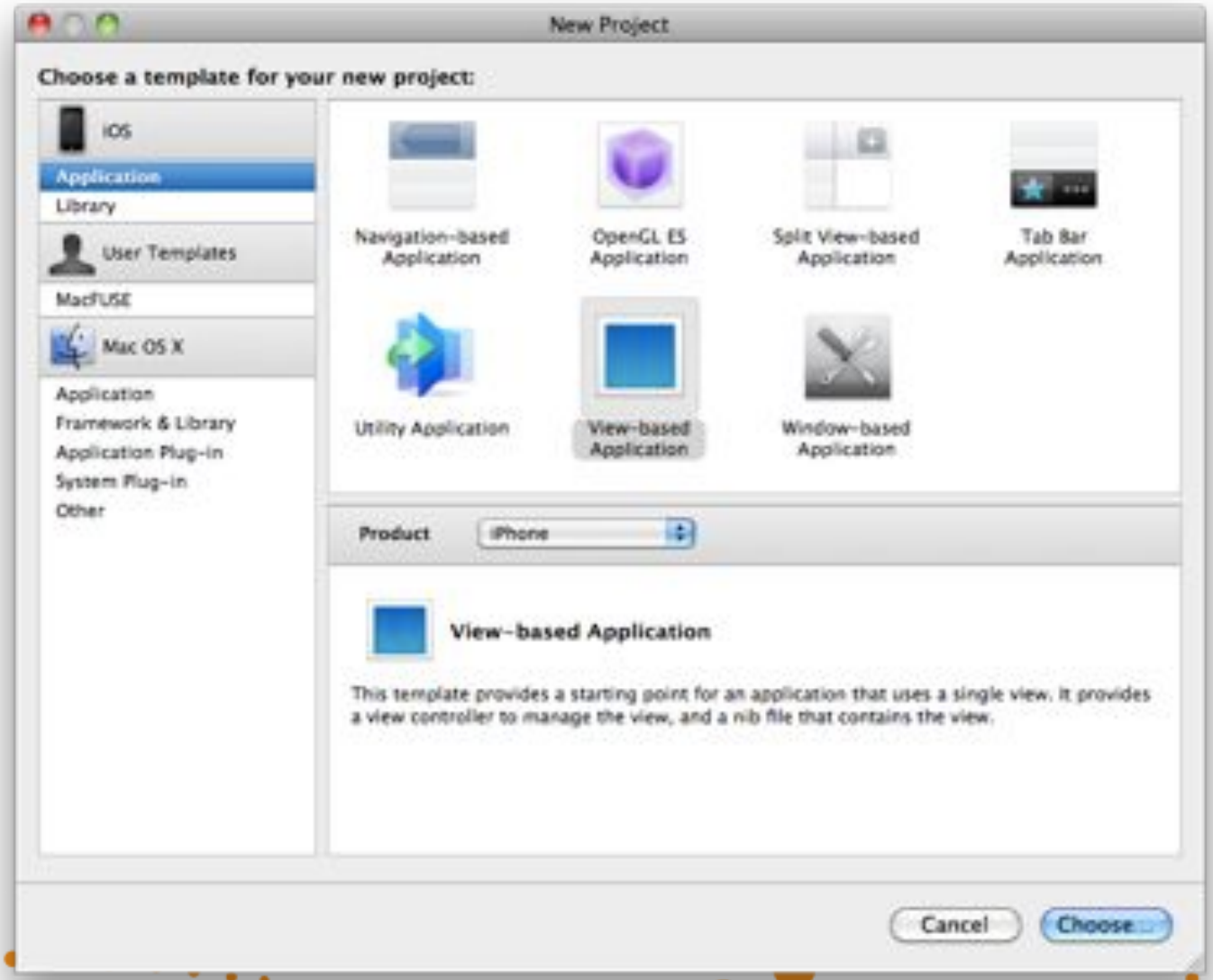
Creating and Deploying an App

Basic steps

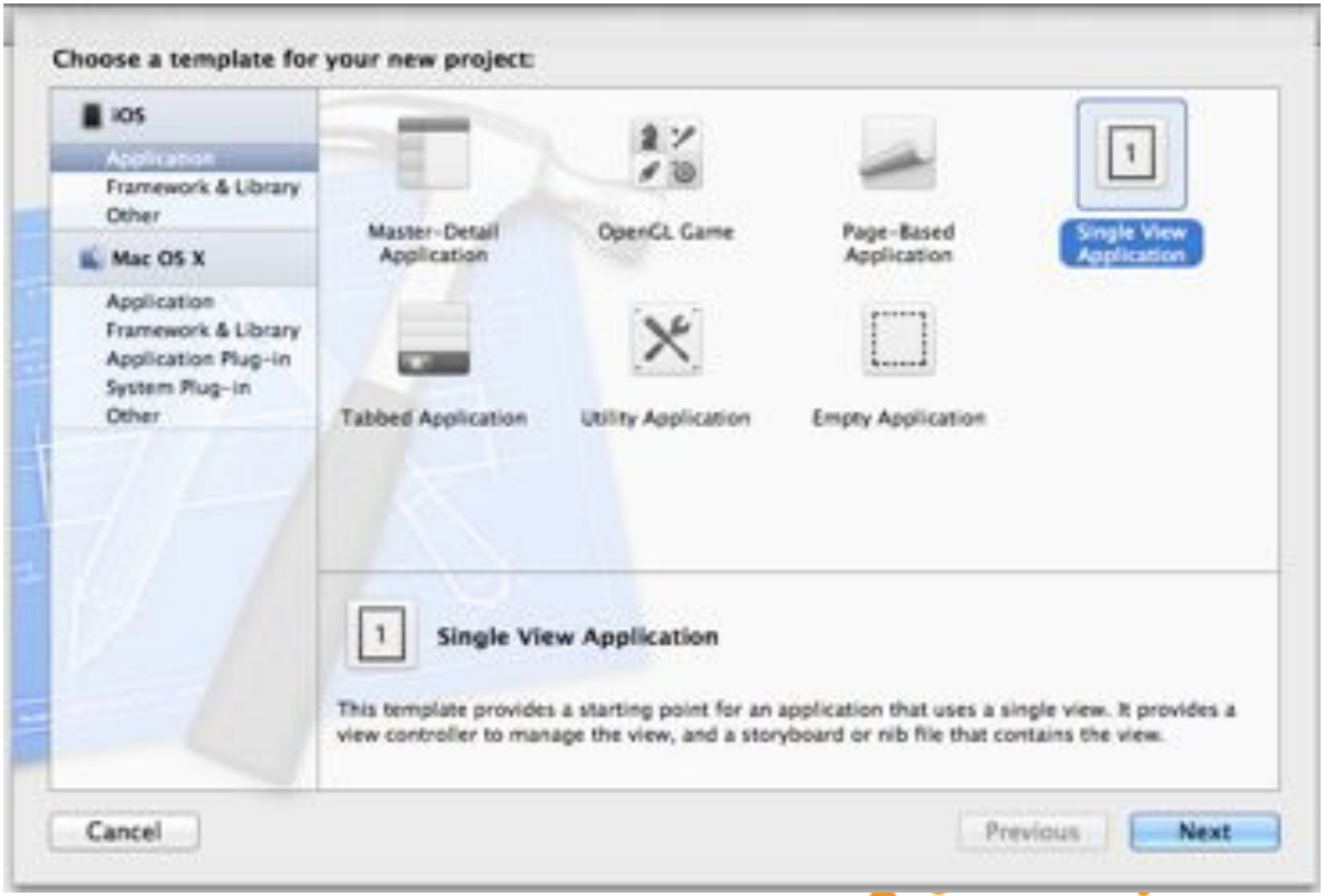
<symbio>

- In XCode, File -> Create New Project, iOS Application
  - Select the View-based Application to get a View and a Controller
  - XCode generates the directory structure and skeleton files for you
- Add functionality to your ObjC code
  - For example, a button click handler
- Add resources
  - For example, a picture (drag-and-drop to Resources folder)
- Open the *.xib* file and add some user interface elements
  - E.g. a button; bind your code objects to UI objects with drag-and-drop
- Run your app!
  - Either in emulator, or over USB on your iOS device (if you have paid!)

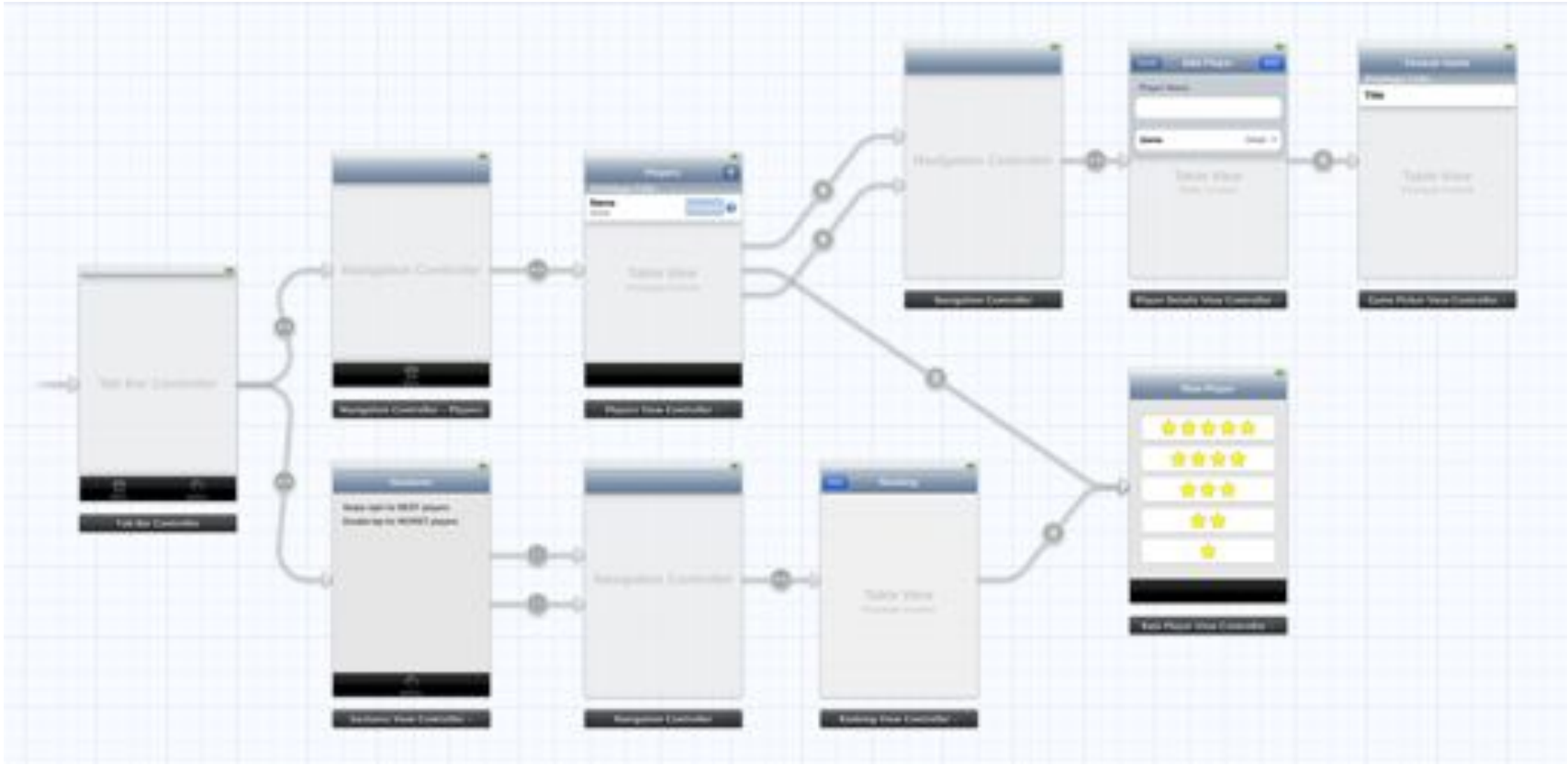
XCode 3: create iOS project



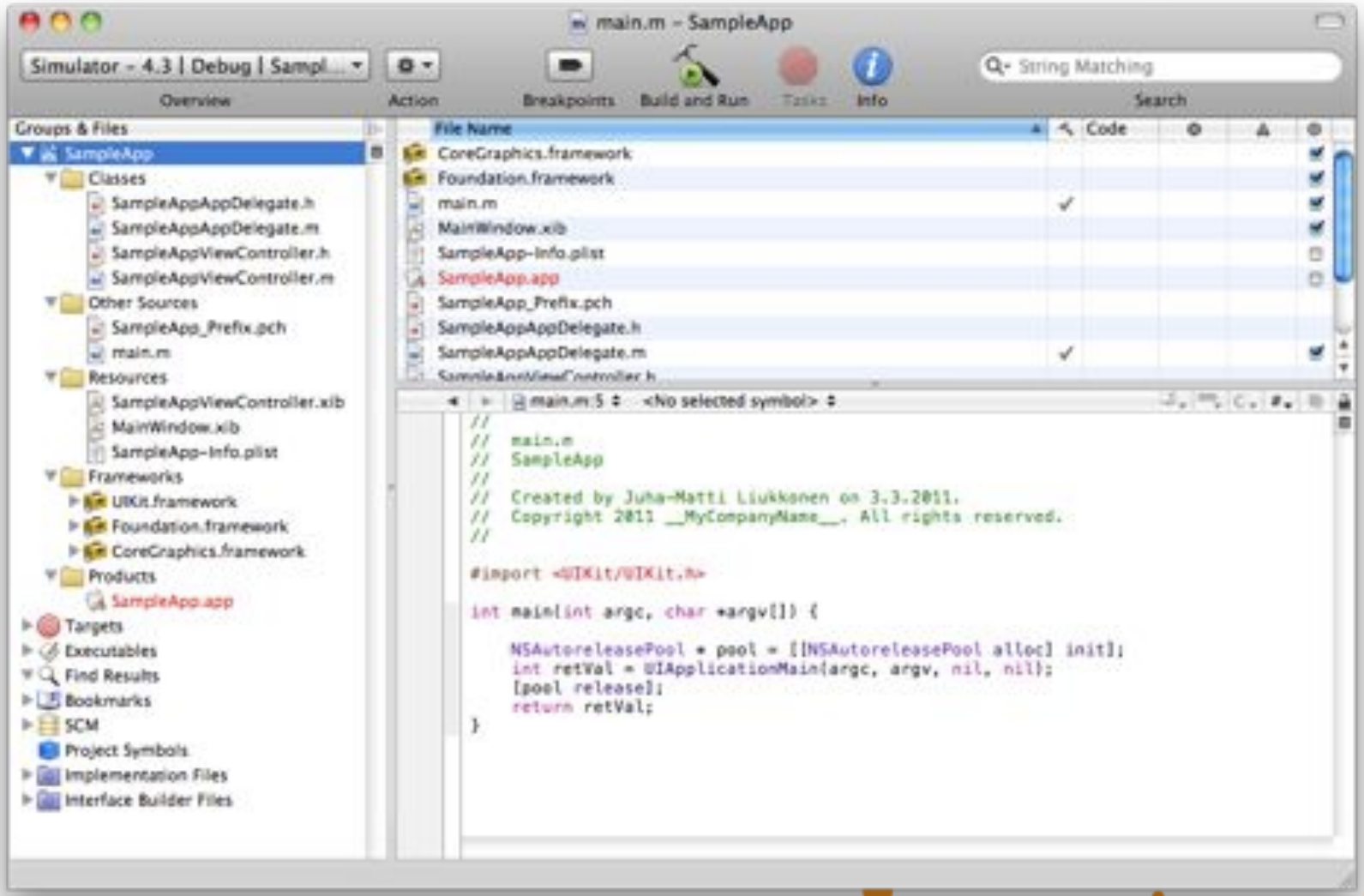
XCode 4: create iOS project



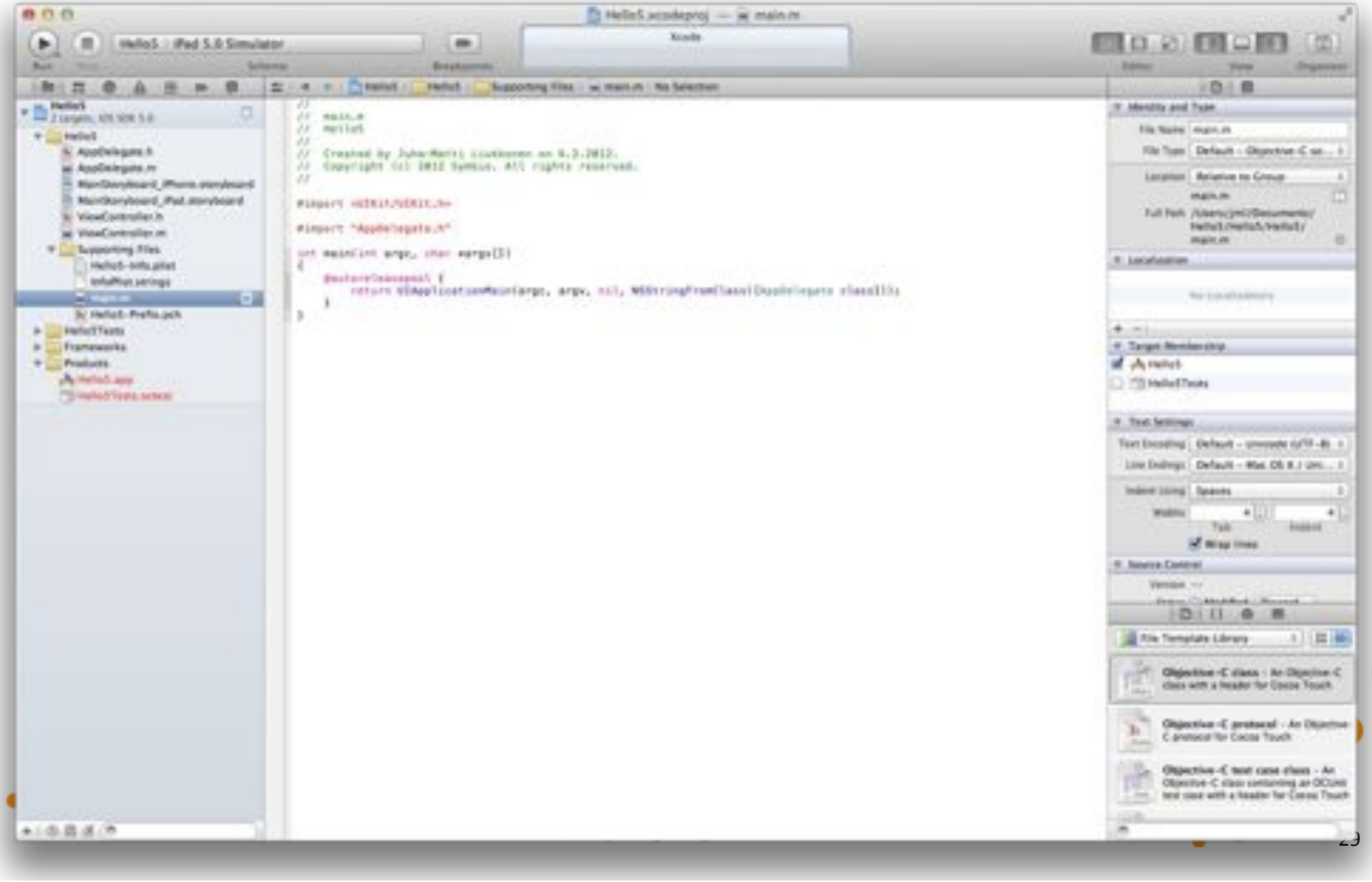
New in iOS 5: Storyboards



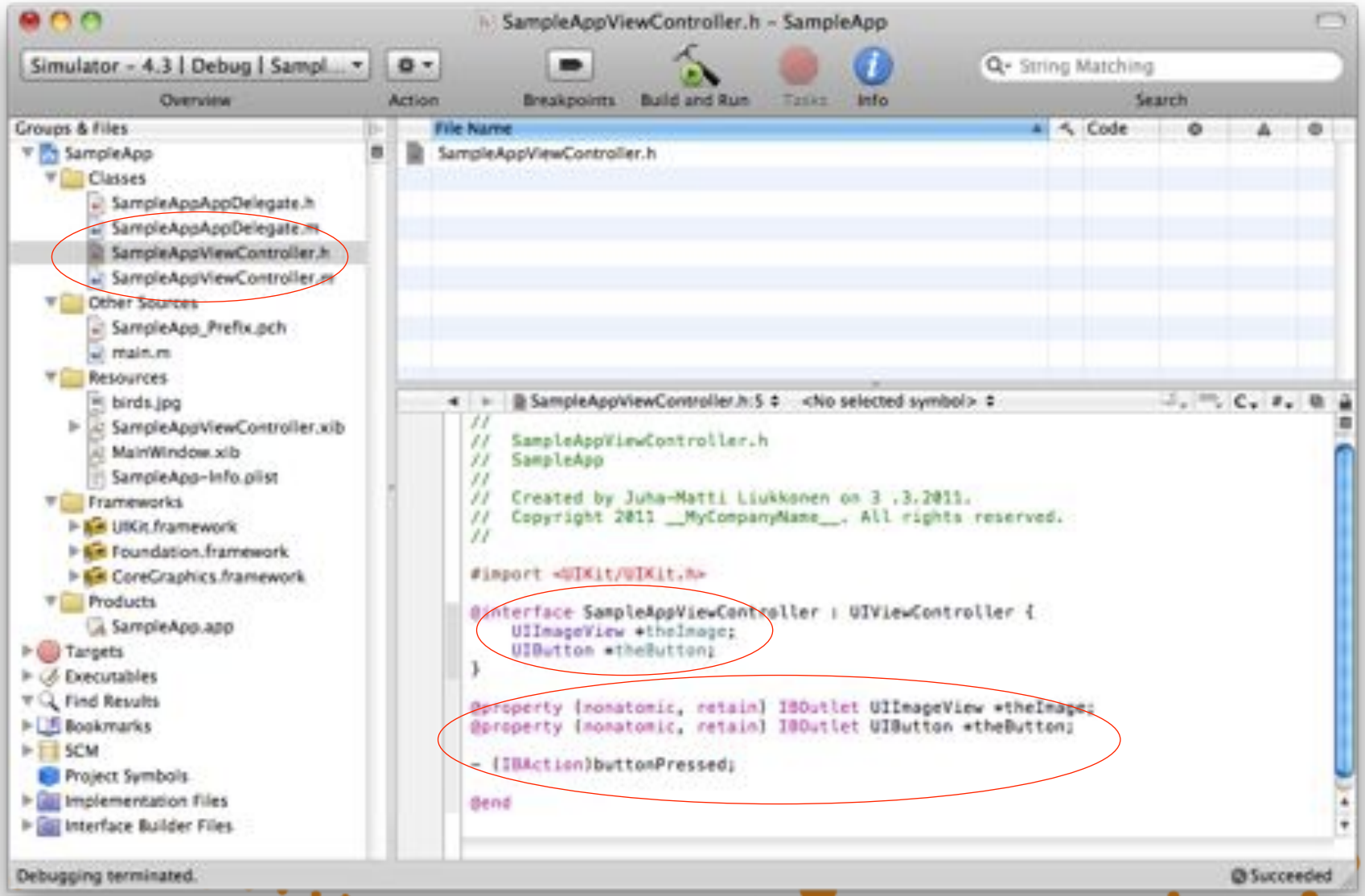
XCode 3: project structure



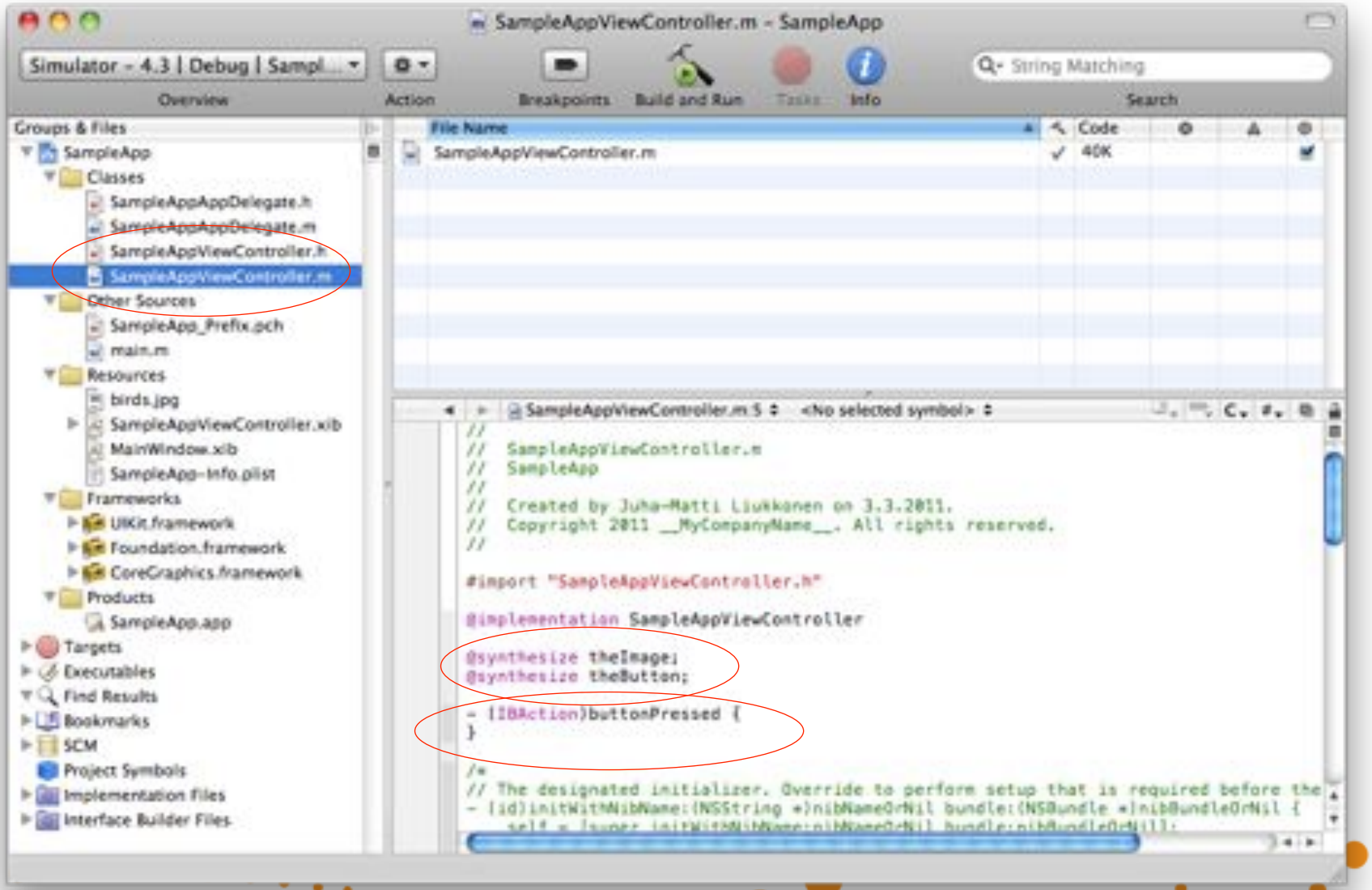
XCode 4: project structure



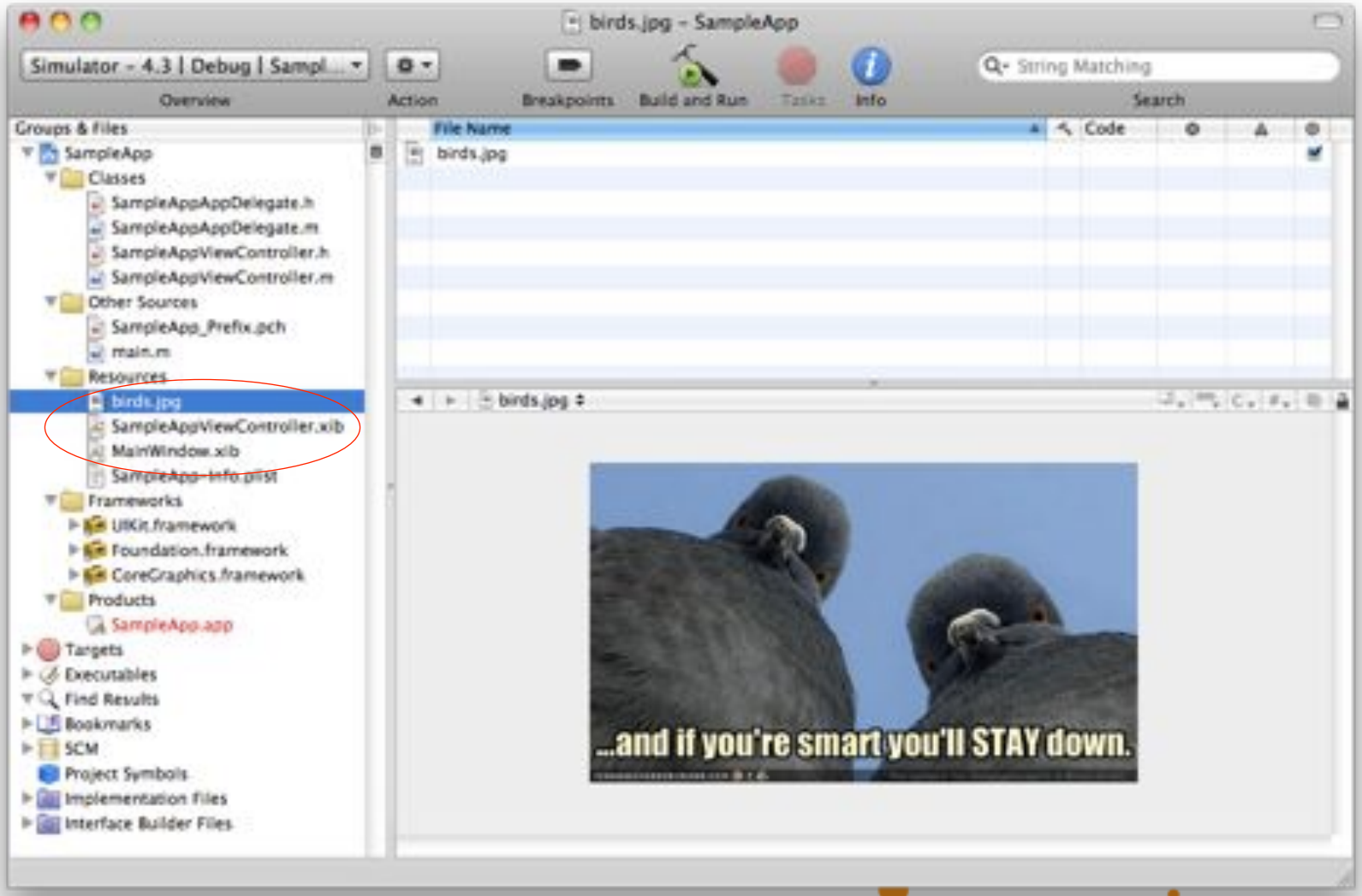
XCode: add UI elements



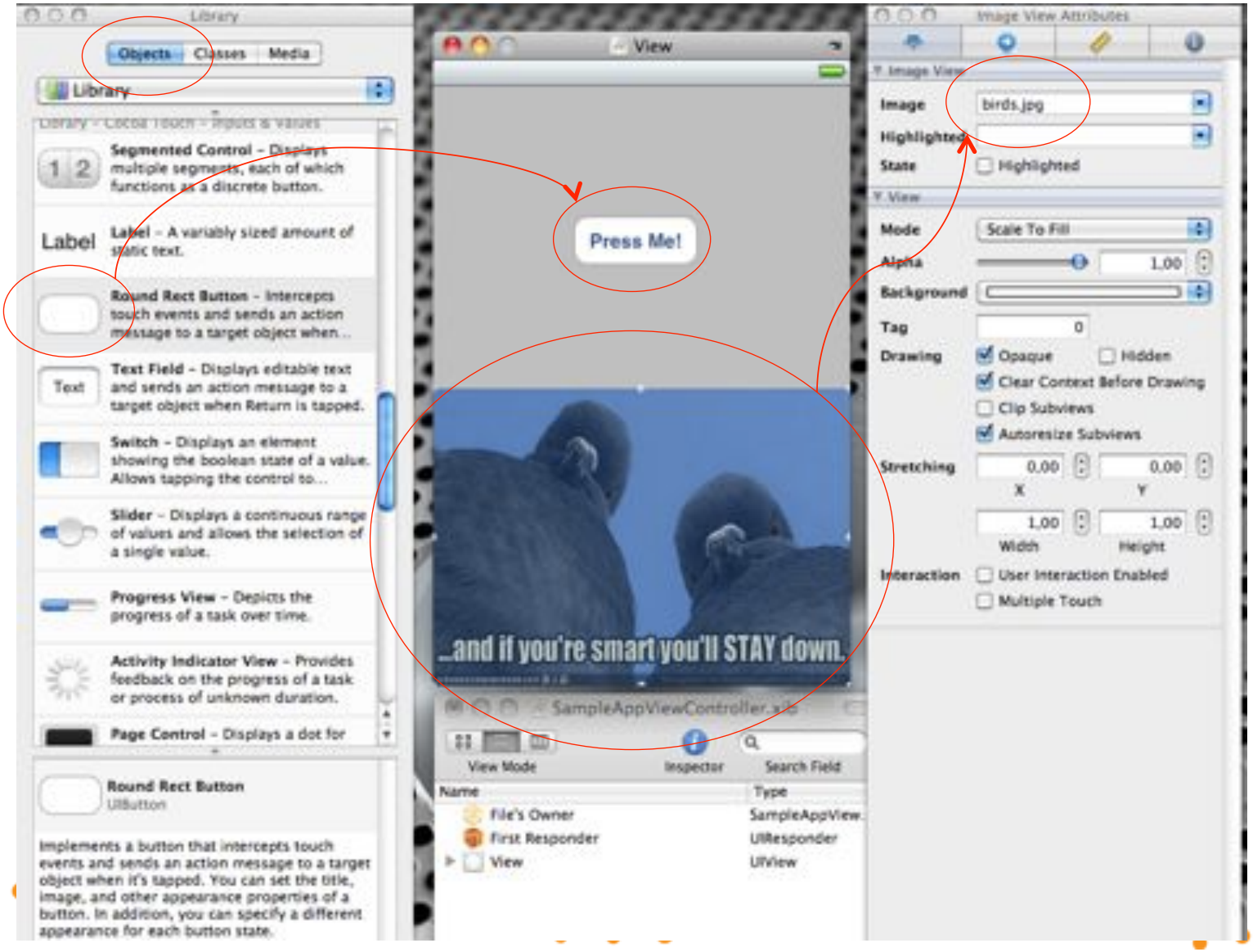
XCode: add implementation



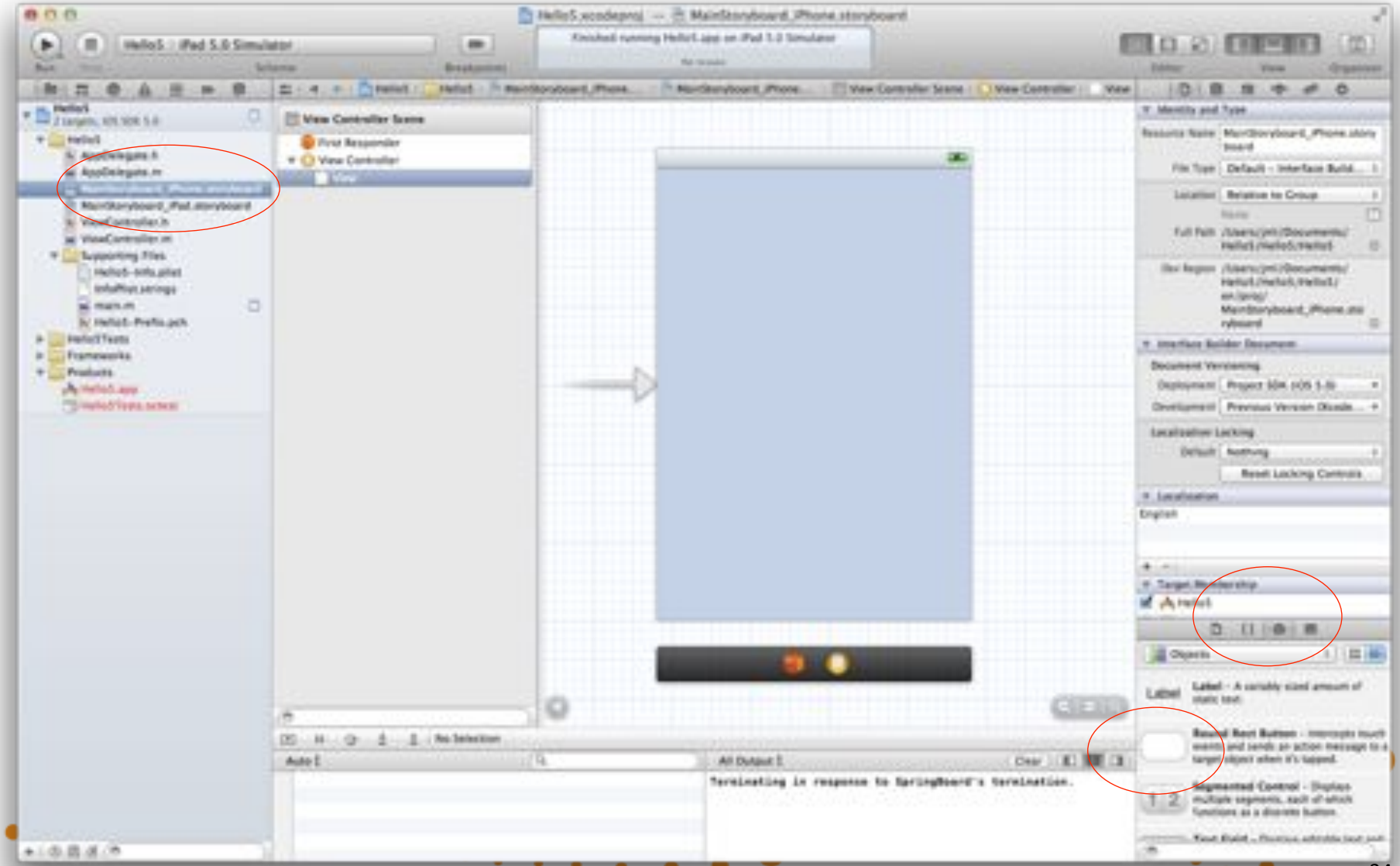
XCode: add resources



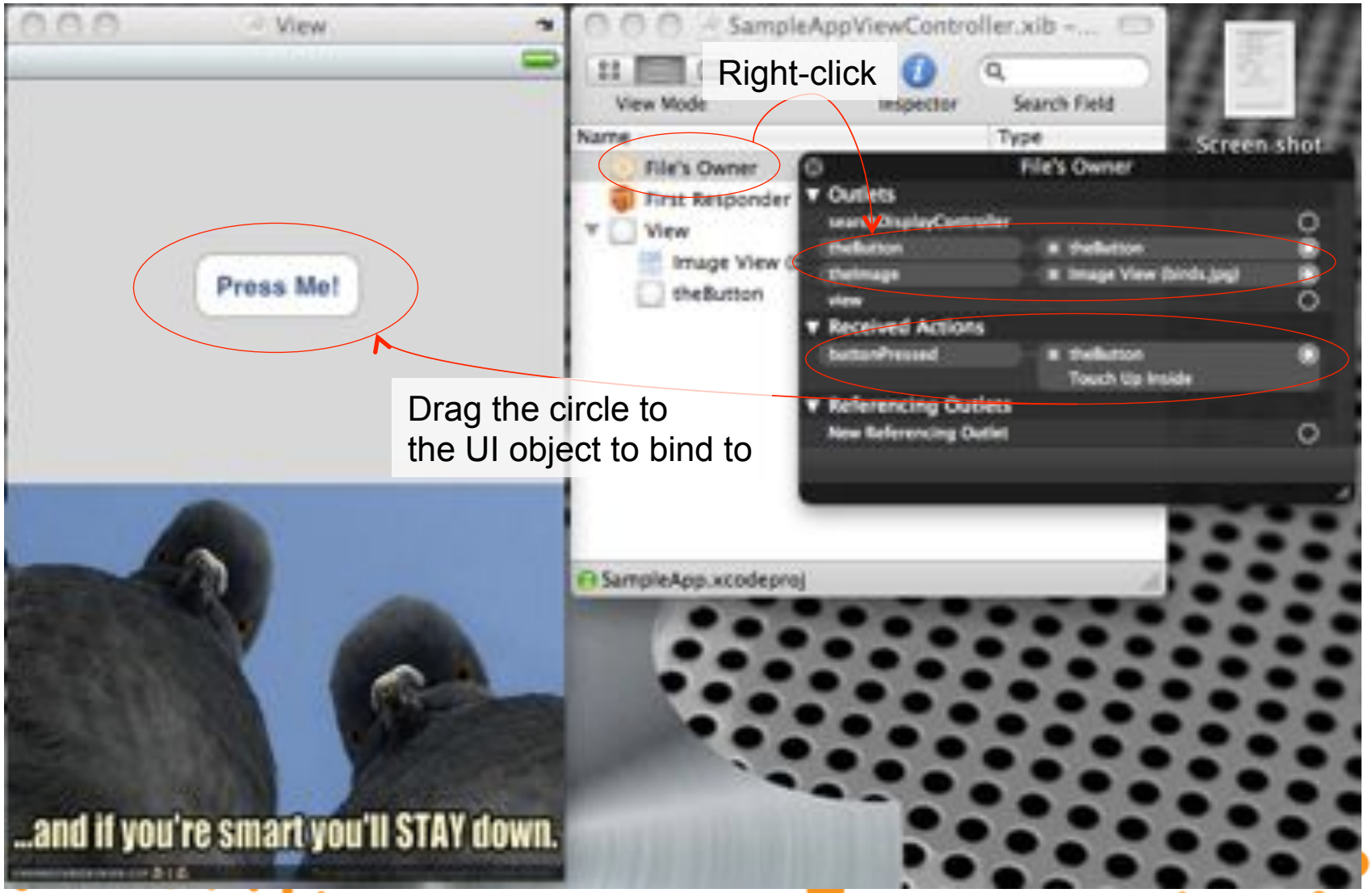
IB: Add UI elements



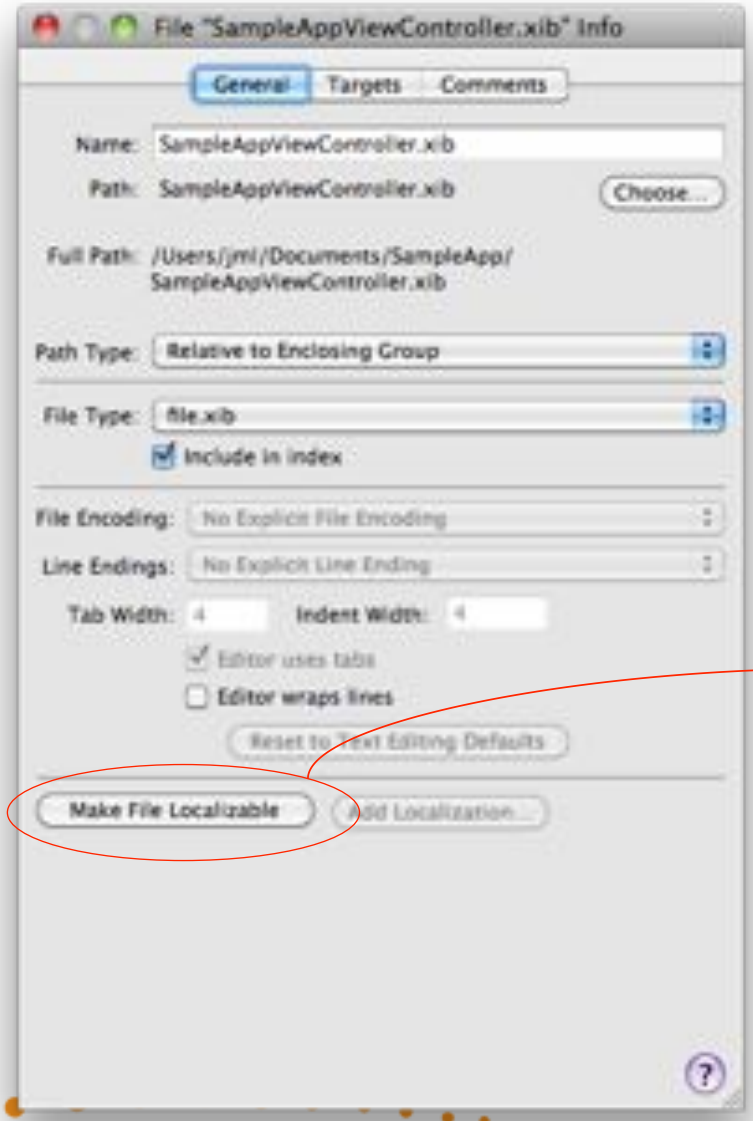
XCode 4: Add UI elements

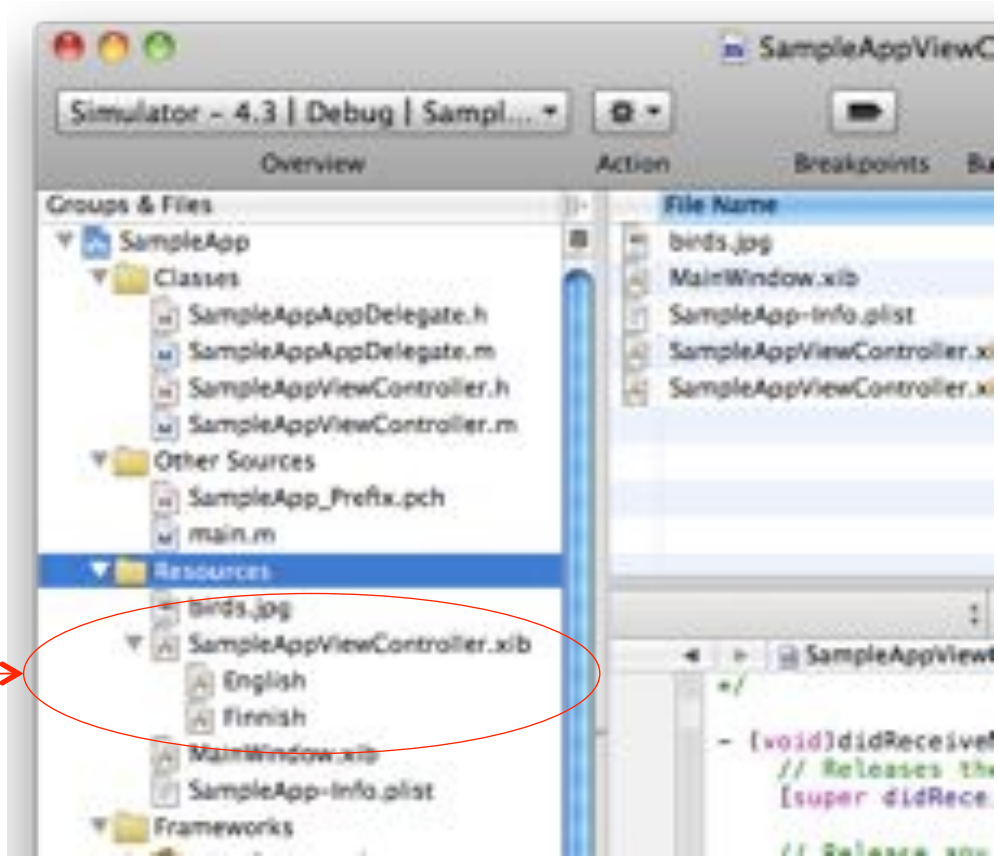


IB: Bind code to UI



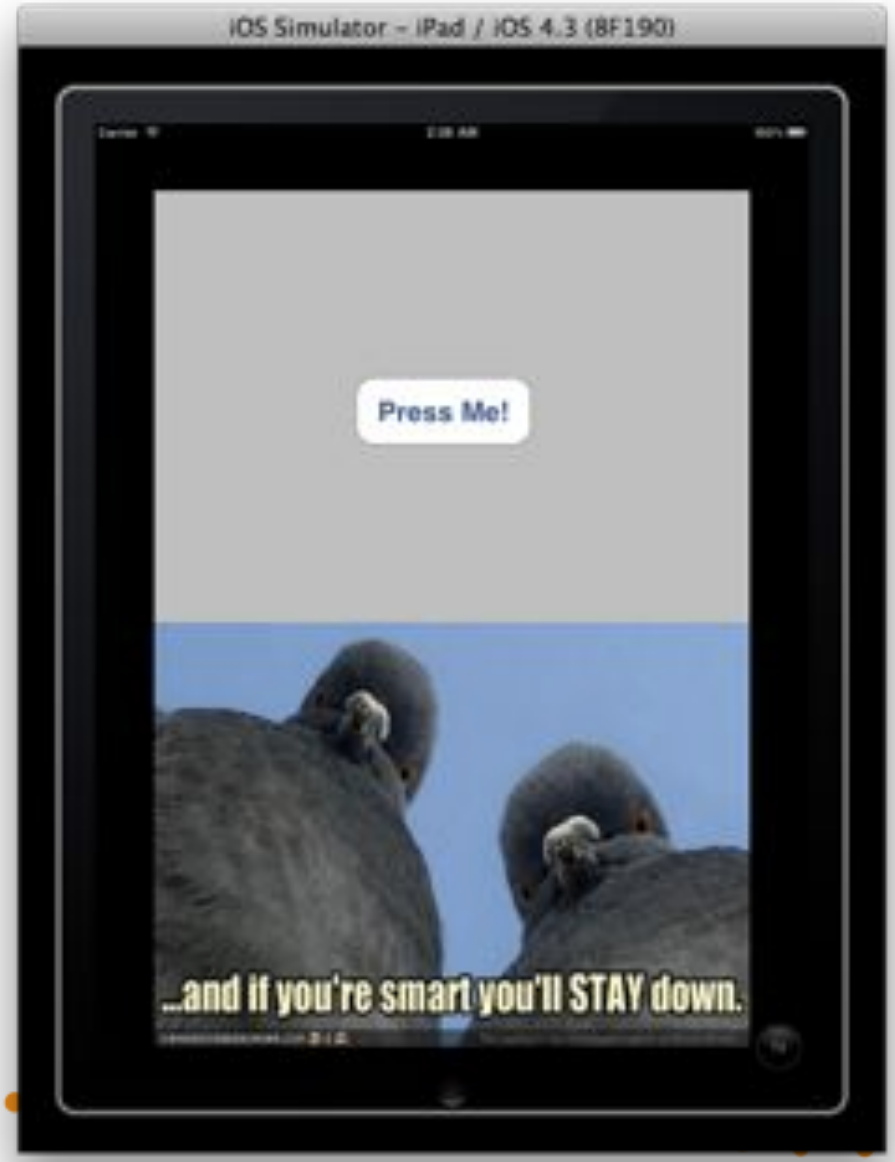
XCode/IB: localization





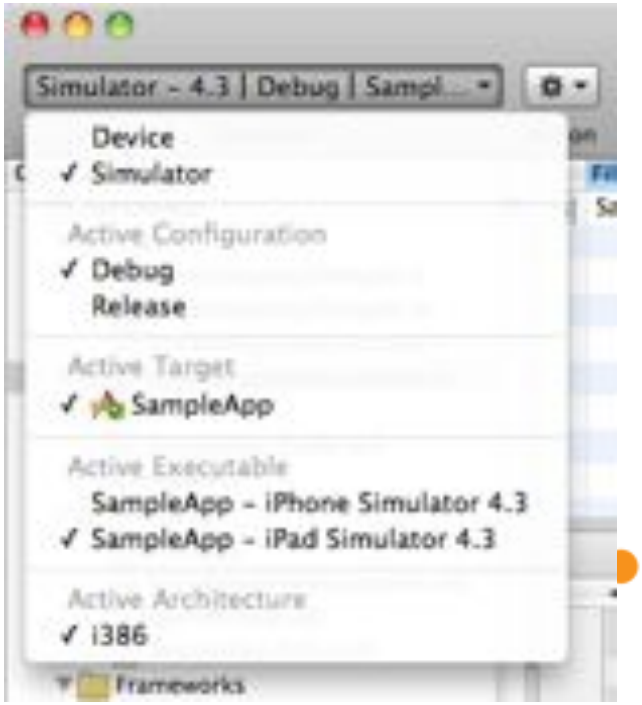
The last step is to edit all .xib file variants in Interface Builder for proper localization.

XCode: run on Target!



Click the “Run” button in the XCode toolbar.

If you have an iOS device, your app can run directly in your device. Choose your target in XCode.



Advanced programming topics

State saving

- When user presses the Home button,
  - In iOS 3, your app will be killed – iOS invokes your delegate's *applicationWillTerminate*
  - In iOS 4 and 5, your app will remain in memory – iOS invokes your delegate's *applicationDidEnterBackground*
  - Even in iOS 4 and 5, if resources become scarce, your app will be killed
  - Best to save state in a dictionary (name, value pairs) whenever either state change method is invoked by iOS
- When your app starts up
  - In your delegate's *application:didFinishLaunchingWithOptions* method, read and restore your state data from the parameter dictionary (if not *nil*)
  - Data stored as (an XML format) *plist* in your app configuration area

Game programming

<symbio>

- iOS apps run native code
  - Maximum performance
  - All hardware resources in use
- Direct access to OpenGL ES
  - Complex 3D user interfaces possible
- Need to honor the app life cycle notifications
  - Immediately stop animations, sounds, etc when told to go to background
  - Device must react to incoming calls, SMS, etc immediately

Distributing your App

The App Store

Change from 2010:
- App count +100%
- Developers +50%
- Ratio of free apps +50%
- Average price – no change
- Less very expensive apps
- Low end price range +50%

- The definitive on-line app store
  - 583000 app titles (Android: 400000)
  - 142000 unique developers (Android: 240000)
  - 45% free apps, 55% paid apps (Android: 70% free, 30% paid)
  - Average paid app price in 2011: $2,06
  - 50% of apps between $0,99 .. $4,99, only 1% at $9,99 or more
  - 30% of revenue goes to Apple, 70% to developer (out of which you pay taxes)
- The 67 € / year fee buys you fairly good QA
  - Easy to submit apps to the Store
  - Apple engineers verify that your app works, keeps quality high
- Apple has a strict acceptance policy
  - Any content which *might* get Apple into liability court is rejected / removed

In Conclusion

- Apple's iOS sets the bar for others
  - Harded to get into than Android – Objective-C language and fairly complicated design patterns required
  - Native code runs efficiently, maximum performance and capabilities
  - Has a reputation of high quality, people expect that and are willing to pay
- Developer tools extensive but fairly complicated
  - New XCode 4 streamlines development
  - Excellent and extensive documentation
  - Sample code in developer portal (requires membership = money)
- Rich and mature APIs
  - Data access, multimedia, wireless services, …
  - Reasonably easy to create simple apps, complex apps require little more

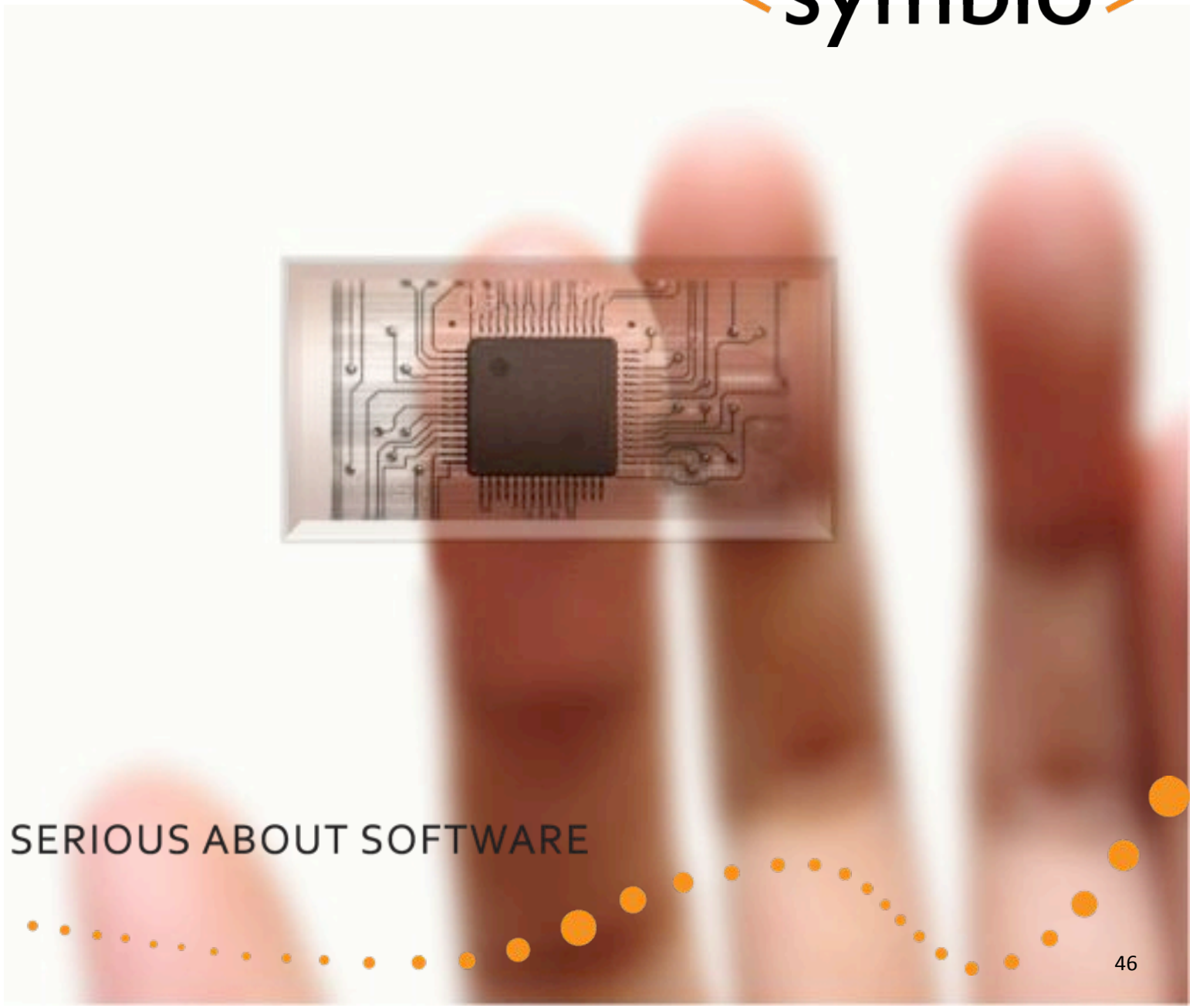
Brief Comparison

Food for thought

<symbio>

	Android	iOS	WP7	Symbian
Units sold in 2010	67 million	47 million	12 million	**111 million**
Device Variants	Many	Some	**None**	Many
Easy if you know…	Java	Objective C	C# or Silverlight	Qt and C++, or Java
Dev platform	**Linux, Mac, Windows**	Mac	Windows	**Linux, Mac, Windows**
Cost to Develop	Free	Free	Free	Free
Cost to distribute	**Free**	$99 / year	$99 / year	1 €
Competition in app space	Fierce	Fierce and controlled	**Not much**	**Not much**

Units sold data: Gartner, Feb 2011


symbio
SERIOUS ABOUT SOFTWARE