Stanford CS193p

Developing Applications for iOS
Spring 2012

Today

- `UIImagePickerController`
  Demo
- Core Motion
  How is this device moving in space?
- Settings
  Global preferences set in the Settings application
- Localization
  Preparing your application to ship in other locales

Demo

- Kitchen Sink

  Dropping images into our sink.

  `UIImagePickerController`

Core Motion

- API to access motion sensing hardware on your device
- Primary inputs: Accelerometer, Gyro, Magnetometer
  Not all devices have all inputs (e.g. only iPhone4 and newest iPod Touch and iPad 2 have a gyro).
- Primary class used to get input is `CMMotionManager`
  Create with `alloc/init`, but use only one instance per application (else performance hit).
  It is a "global resource," so getting one via an application delegate method or class method is okay.
- Usage
  1. Check to see what hardware is available.
  2. Start the sampling going and poll the motion manager for the latest sample it has.

  ... or ...

  1. Check to see what hardware is available.
  2. Set the rate at which you want data to be reported from the hardware,
  3. Register a block (and a dispatch queue to run it on) each time a sample is taken.

Core Motion

- Checking availability of hardware sensors

  `@property (readonly) BOOL {accelerometer,gyro,magnetometer,deviceMotion}Available;`

  The "device motion" is a combination of all available (accelerometer, magnetometer, gyro).

  We'll talk more about that in a couple of slides.

- Starting the hardware sensors collecting data

  You only need to do this if you are going to poll for data.

  `- (void)start{Accelerometer,Gyro,Magnetometer,DeviceMotion}Updates;`

- Is the hardware currently collecting data?

  `@property (readonly) BOOL {accelerometer,gyro,magnetometer,deviceMotion}Active;`

- Stop the hardware collecting data

  It is a performance hit to be collecting data, so stop during times you don't need the data.

  `- (void)stop{Accelerometer,Gyro,Magnetometer,DeviceMotion}Updates;`

Core Motion

- Checking the data (polling not recommended, more later)

```
@property (readonly) CMAccelerometerData *accelerometerData;
```
CMAccelerometerData object provides `@property (readonly) CMAcceleration acceleration;`
`typedef struct { double x; double y; double z; } CMAcceleration;` // x, y, z in "g"
This raw data includes acceleration due to gravity.

```
@property (readonly) CMGyroData *gyroData;
```
CMGyroData object has one property `@property (readonly) CMRotationRate rotationRate;`
`typedef struct { double x; double y; double z; } CMRotationRate;` // x, y, z in radians/second
Sign of rotation rate follows right hand rule. This raw data will be biased.

```
@property (readonly) CMMagnetometerData *magnetometerData;
```
CMMagnetometerData object has one property `@property (readonly) CMMagneticField magneticField;`
`typedef struct { double x; double y; double z; } CMMagneticField;` // x, y, z in microteslas
This raw data will be biased.

```
@property (readonly) CMDeviceMotion *deviceMotion;
```
CMDeviceMotion is an intelligent combination of gyro and acceleration.
If you have multiple detection hardware, you can report better information about each.

CMDeviceMotion

- Acceleration Data in CMDeviceMotion

```
@property (readonly) CMAcceleration gravity;
@property (readonly) CMAcceleration userAcceleration;  // gravity factored out using gyro
typedef struct { double x; double y; double z; } CMAcceleration; // x, y, z in "g"
```

- Rotation Data in CMDeviceMotion

```
@property CMRotationRate rotationRate;  // bias removed from raw data using accelerometer
typedef struct { double x; double y; double z; } CMRotationRate; // x, y, z in radians/second

@property CMAttitude *attitude;  // device's attitude (orientation) in 3D space

@interface CMAttitude : NSObject     // roll, pitch and yaw are in radians
@property (readonly) double roll;    // around longitudinal axis passing through top/bottom
@property (readonly) double pitch;   // around lateral axis passing through sides
@property (readonly) double yaw;     // around axis with origin at center of gravity and
                                     // perpendicular to screen directed down
   // other mathematical representations of the device's attitude also available
@end
```

CMDeviceMotion

- Magnetic Field Data in CMDeviceMotion

```
@property (readonly) CMCalibratedMagneticField magneticField;
struct {
    CMMagneticField field;
    CMMagneticFieldCalibrationAccuracy accuracy;
} CMCalibratedMagneticField;
enum {
    CMMagneticFieldCalibrationAccuracyUncalibrated,
                                      Low,
                                      Medium,
                                      High
} CMMagneticFieldCalibrationAccuracy;
```

Core Motion

- Registering a block to receive Accelerometer data

```
- (void)startAccelerometerUpdatesToQueue:(NSOperationQueue *)queue
                             withHandler:(CMAccelerometerHandler)handler;
typedef void (^CMAccelerationHandler)(CMAccelerometerData *data, NSError *error);
```

We haven't talked about `NSOperationQueue`, but think of it as an OO `dispatch_queue_t`.
Use `[[NSOperationQueue alloc] init]` or `[NSOperation mainQueue` (or `currentQueue`)].

- Registering a block to receive Gyro data

```
- (void)startGyroUpdatesToQueue:(NSOperationQueue *)queue
                    withHandler:(CMGyroHandler)handler;
typedef void (^CMGyroHandler)(CMGyroData *data, NSError *error)
```

- Registering a block to receive Magnetometer data

```
- (void)startMagnetometerUpdatesToQueue:(NSOperationQueue *)queue
                             withHandler:(CMMagnetometerHandler)handler;
typedef void (^CMMagnetometerHandler)(CMMagnetometerData *data, NSError *error)
```

Core Motion

- Registering a block to receive (intelligently) combined data

```
- (void)startDeviceMotionUpdatesToQueue:(NSOperationQueue *)queue
                            withHandler:(CMDeviceMotionHandler)handler;
typedef void (^CMDeviceMotionHandler)(CMDeviceMotion *motion, NSError *error);
Interesting NSError types: CMErrorDeviceRequiresMovement/CMErrorTrueNorthNotAvailable

- (void)startDeviceMotionUpdatesUsingReferenceFrame:(CMAttitudeReferenceFrame)frame
                                            toQueue:(NSOperationQueue *)queue
                                        withHandler:(CMDeviceMotionHandler)handler;
enum {
    CMAttitudeReferenceFrameXArbitraryZVertical,
                            XArbitraryCorrectedZVertical, // needs magnetometer; more CPU
                            XMagneticZVertical,           // above + device movement
                            XTrueNorthZVertical           // requires GPS + magnetometer
}
@property (nonatomic) BOOL showsDeviceMovementDisplay; // whether to put up UI if required
```

Core Motion

- Setting the rate at which your block gets executed

```
@property NSTimeInterval accelerometerUpdateInterval;
@property NSTimeInterval gyroUpdateInterval;
@property NSTimeInterval magnetometerUpdateInterval;
@property NSTimeInterval deviceMotionUpdateInterval;
```

- It is okay to add multiple handler blocks

  Even though you are only allowed one `CMMotionManager`.
  However, each of the blocks will receive the data at the same rate (as set above).
  (Multiple objects are allowed to poll at the same time as well, of course.)

Demo

- Kitchen Sink

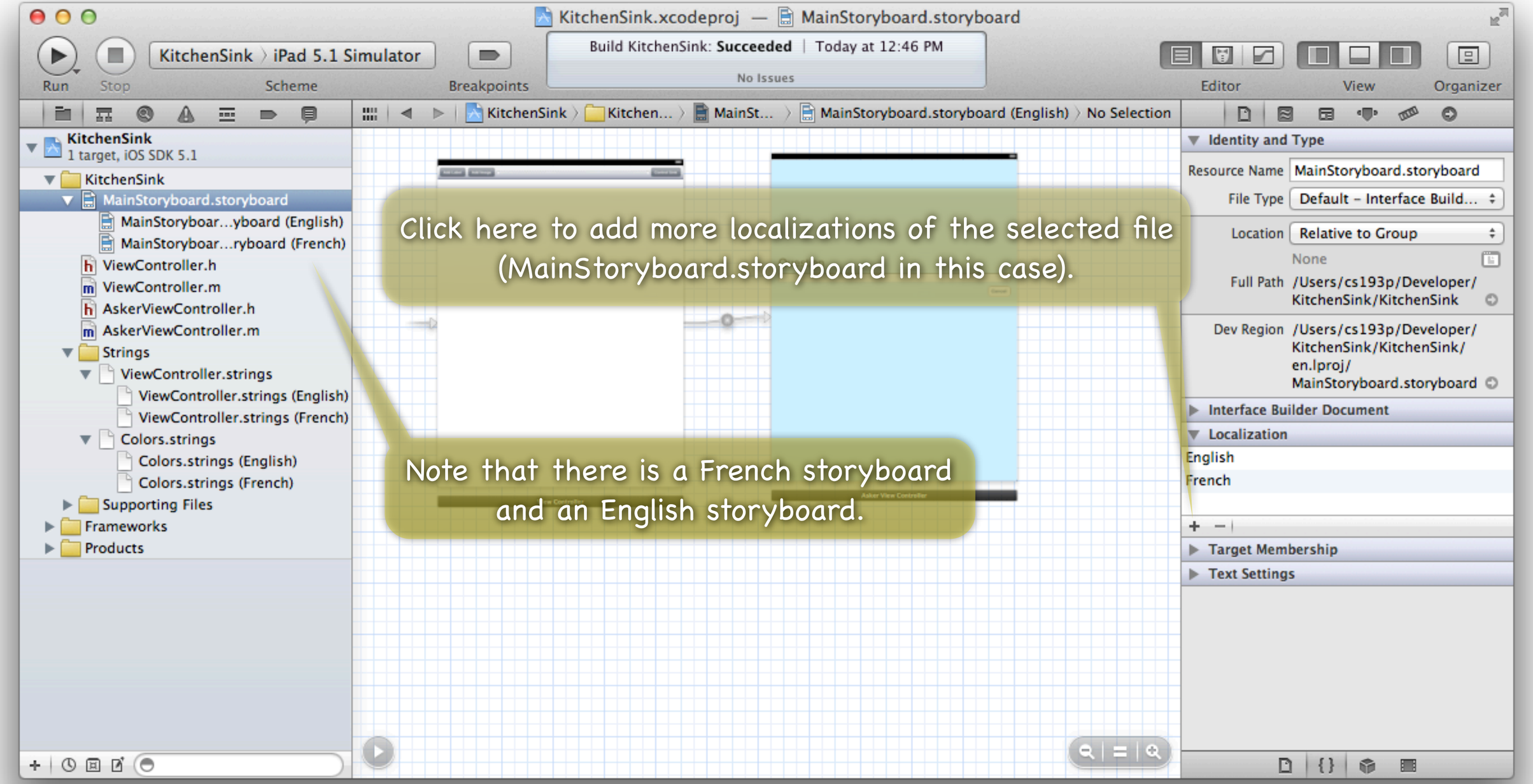
  Swishing things around in our sink by tilting the device.

Localization

- Two steps to making international versions of your application
  Internationalization (i18n)
  Localization (l10n)
- Resources are drawn from a "bundle" using the user's locale
  Inside a bundle, there will be "`.lproj`" directories (e.g. `en.lproj`, `fr.lproj`, etc.)
  Inside these `.lproj` directories, there will be `.storyboard` files, `.strings` files, images, sounds, etc.
  When you get a path to a file from a bundle, it tries top-level first, then searches `.lprojs`
  (depending on the language the user has chosen for his system in Settings app)
- Bundles can be associated with a framework or an application
- Using `NSBundle` API to get a resource (e.g. an image or sound)

```
NSBundle *bundle = [NSBundle bundleForClass:[self class]];
NSString *path = [bundle pathForResource:@"speedlimit" ofType:@"jpg"];
```

`bundleForClass:` knows whether that class came from a framework or just with the application


KitchenSink.xcodeproj — MainStoryboard.storyboard
Run
Stop
KitchenSink 〉 iPad 5.1 Simulator
Scheme
Breakpoints
Build KitchenSink: Succeeded | Today at 12:46 PM
No Issues
Editor
View
Organizer
KitchenSink
1 target, iOS SDK 5.1
KitchenSink
MainStoryboard.storyboard
MainStoryboar...yboard (English)
MainStoryboar...ryboard (French)
ViewController.h
ViewController.m
AskerViewController.h
AskerViewController.m
Strings
ViewController.strings
ViewController.strings (English)
ViewController.strings (French)
Colors.strings
Colors.strings (English)
Colors.strings (French)
Supporting Files
Frameworks
Products
KitchenSink 〉 Kitchen... 〉 MainSt... 〉 MainStoryboard.storyboard (English) 〉 No Selection
Click here to add more localizations of the selected file (MainStoryboard.storyboard in this case).
Note that there is a French storyboard and an English storyboard.
Identity and Type
Resource Name MainStoryboard.storyboard
File Type Default – Interface Build...
Location Relative to Group
None
Full Path /Users/cs193p/Developer/KitchenSink/KitchenSink
Dev Region /Users/cs193p/Developer/KitchenSink/KitchenSink/en.lproj/MainStoryboard.storyboard
Interface Builder Document
Localization
English
French
Target Membership
Text Settings

Localization

- Localizing literal @"strings" in your application

  Special NSBundle method for that (usually sent to [NSBundle mainBundle]) …

```
- (NSString *)localizedStringForKey:(NSString *)key
                              value:(NSString *)defaultValue // if nil, will be key
                              table:(NSString *)tableName;   // if nil: Localizable.strings
```

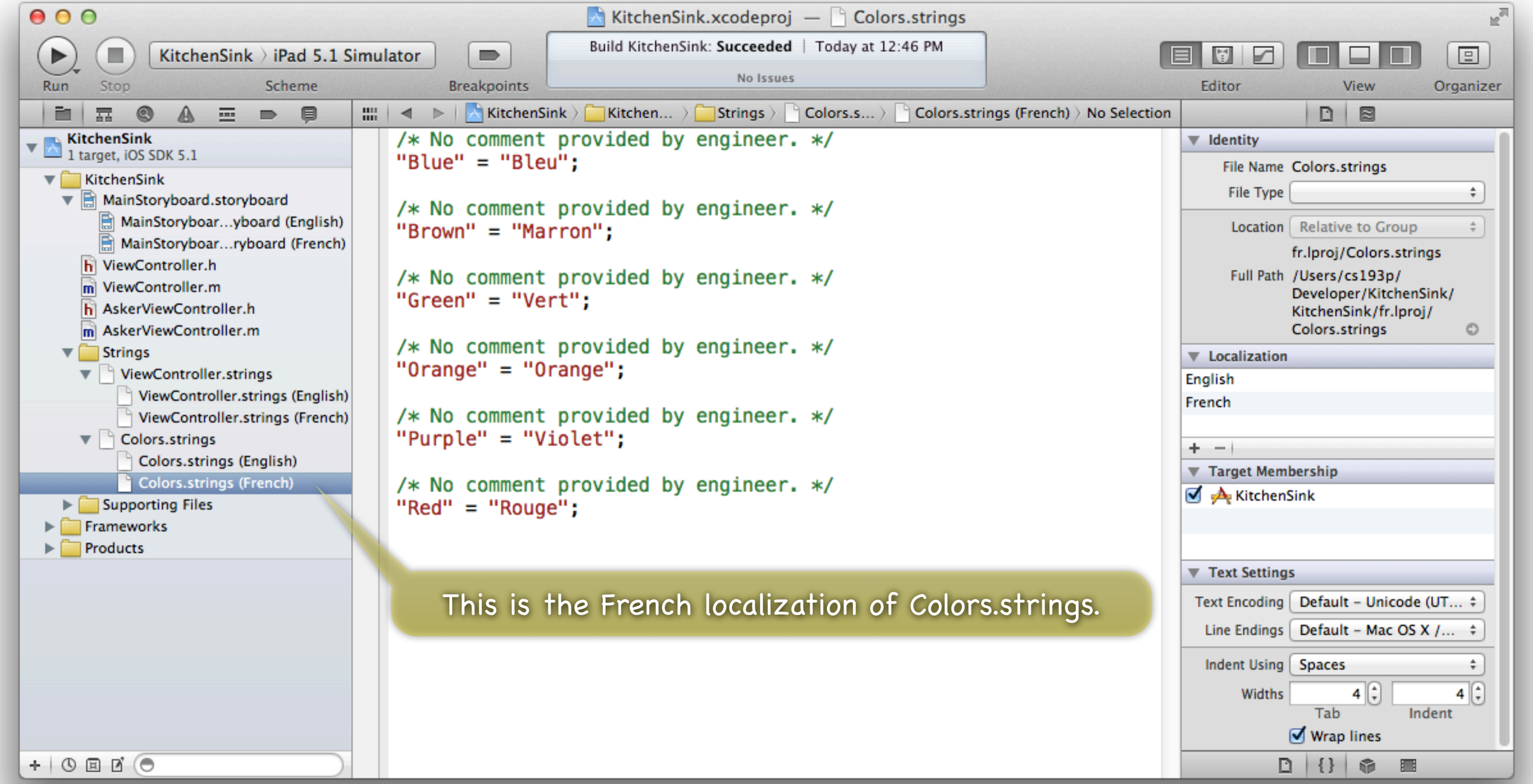
- But there's a better way …

```
NSString *NSLocalizedStringWithDefaultValue(NSString *key, NSString *table,
                                            NSString *bundle, NSString *defaultValue,
                                            NSString *comment); // comment is for localizers
```

  Also NSLocalizedStringFromTableInBundle() (defaultValue is the key)
  and NSLocalizedStringFromTable() (defaultValue is the key and uses mainBundle).

  Now use the command line utility genstrings to generate template .strings files from .m files!

```
cd <directory where all your .m files are>
genstrings -o en.lproj *.m
```



Stanford CS193p
Spring 2012

Localization

- Debugging

  Set the `NSUserDefault` `NSShowNonLocalizedStrings` to `YES` and a message will be logged to the console whenever these `NSLocalizedString` methods cannot find a string.

- Warning

  When you change a file (e.g. `ViewController.strings`) to be localized, you should probably Build Clean (sometimes the old, non-localized version will get installed at the top level and then your localized versions will not get used).

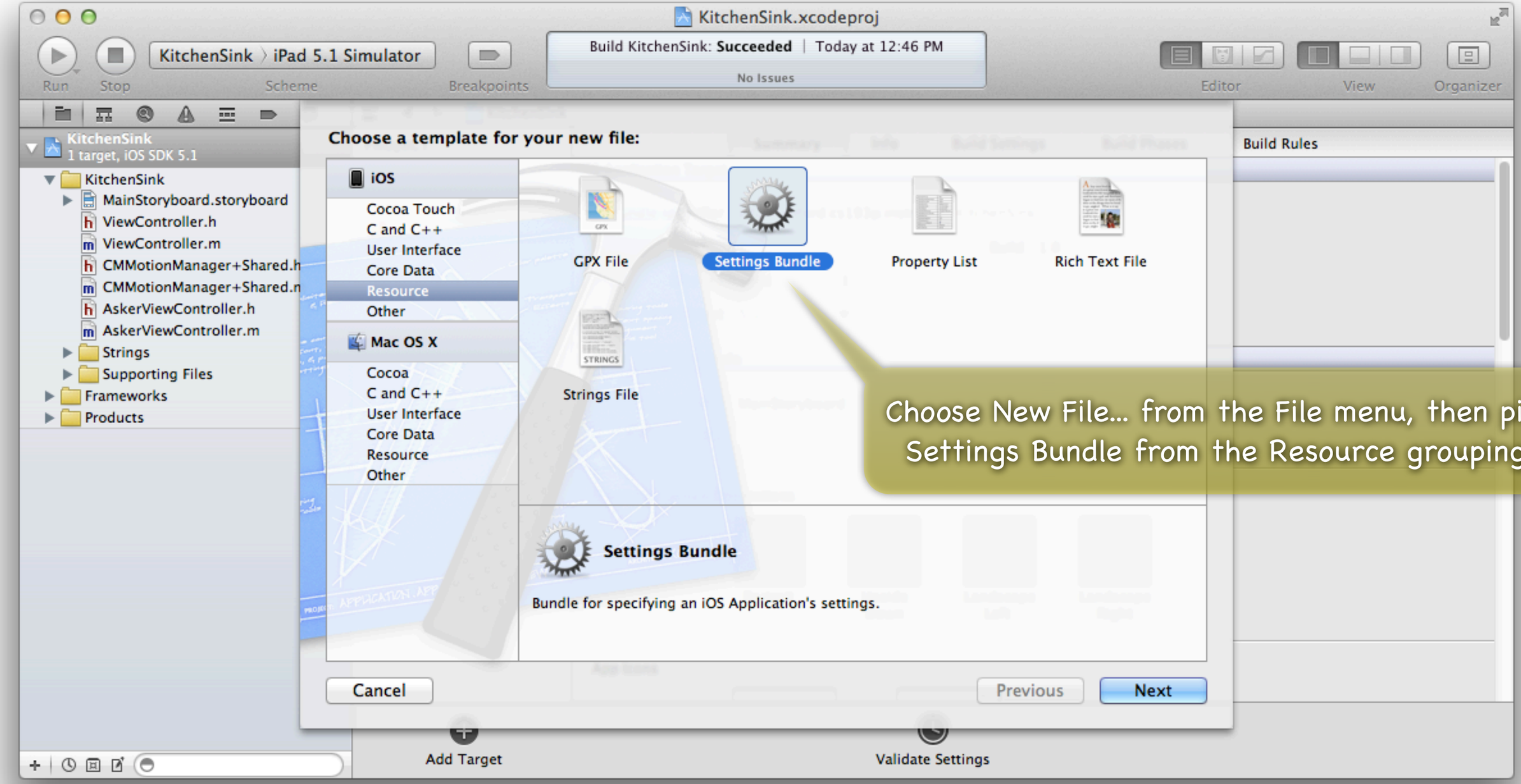
Demo

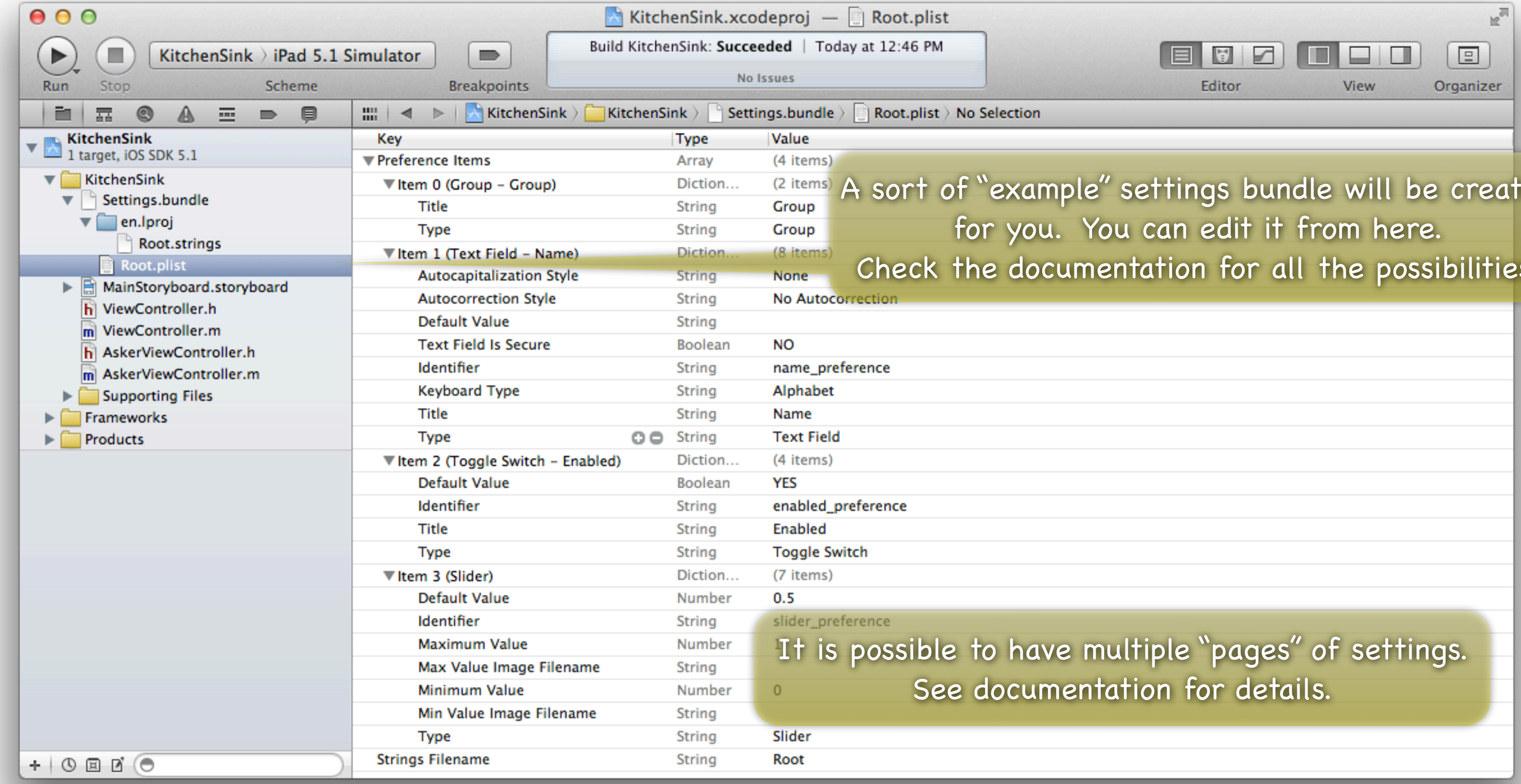
- Kitchen Sink

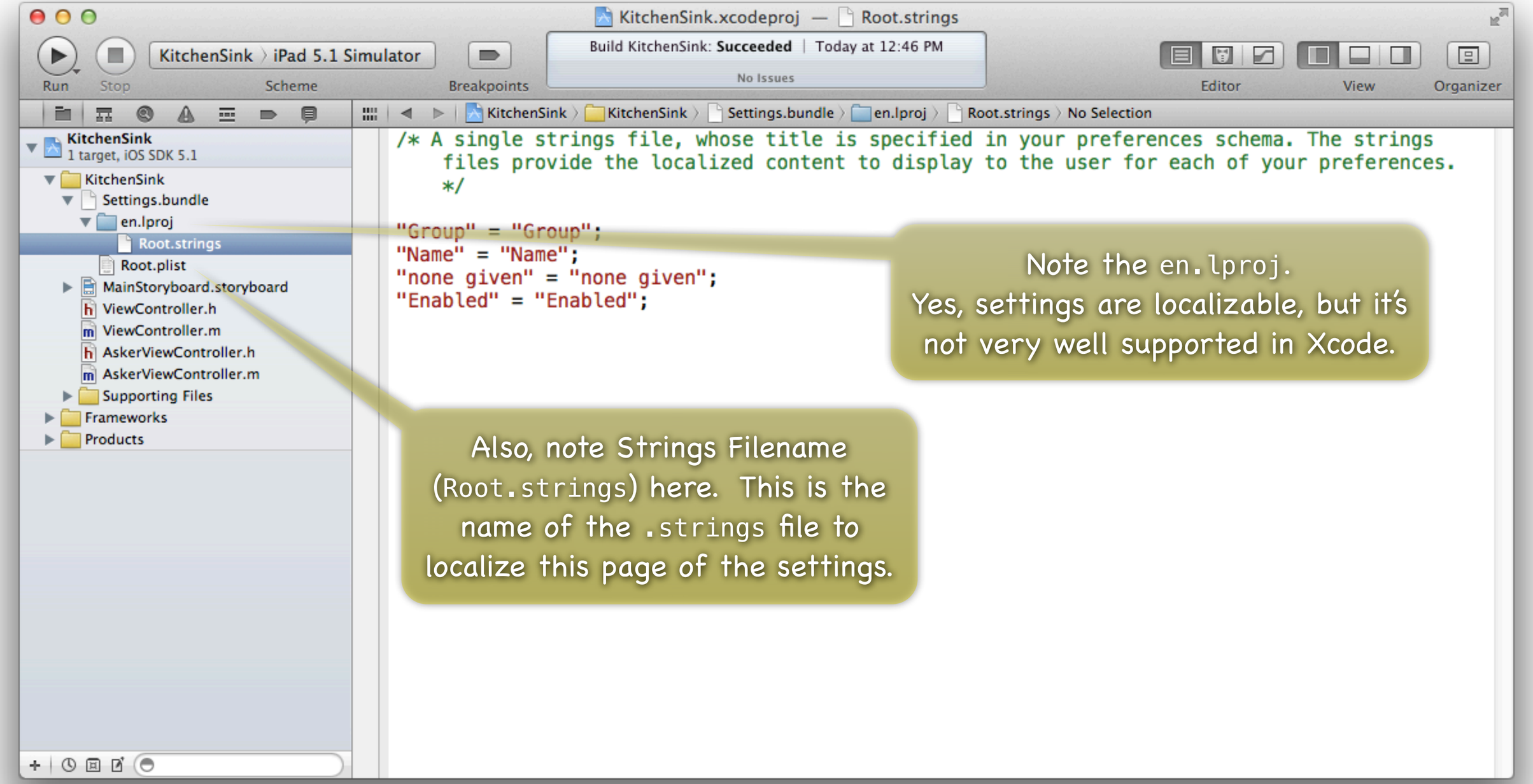
  En Français!

Settings

- A little bit of UI for your application in the Settings application
  - You should use this sparingly (if at all).
  - It's appropriate only for very rarely used settings or default behavior.
  - You don't want to make your users ever have to go here for normal use of your application.
  - The settings appear in your application via `NSUserDefaults`.
  - You specify the UI and the associated default in a property list file …

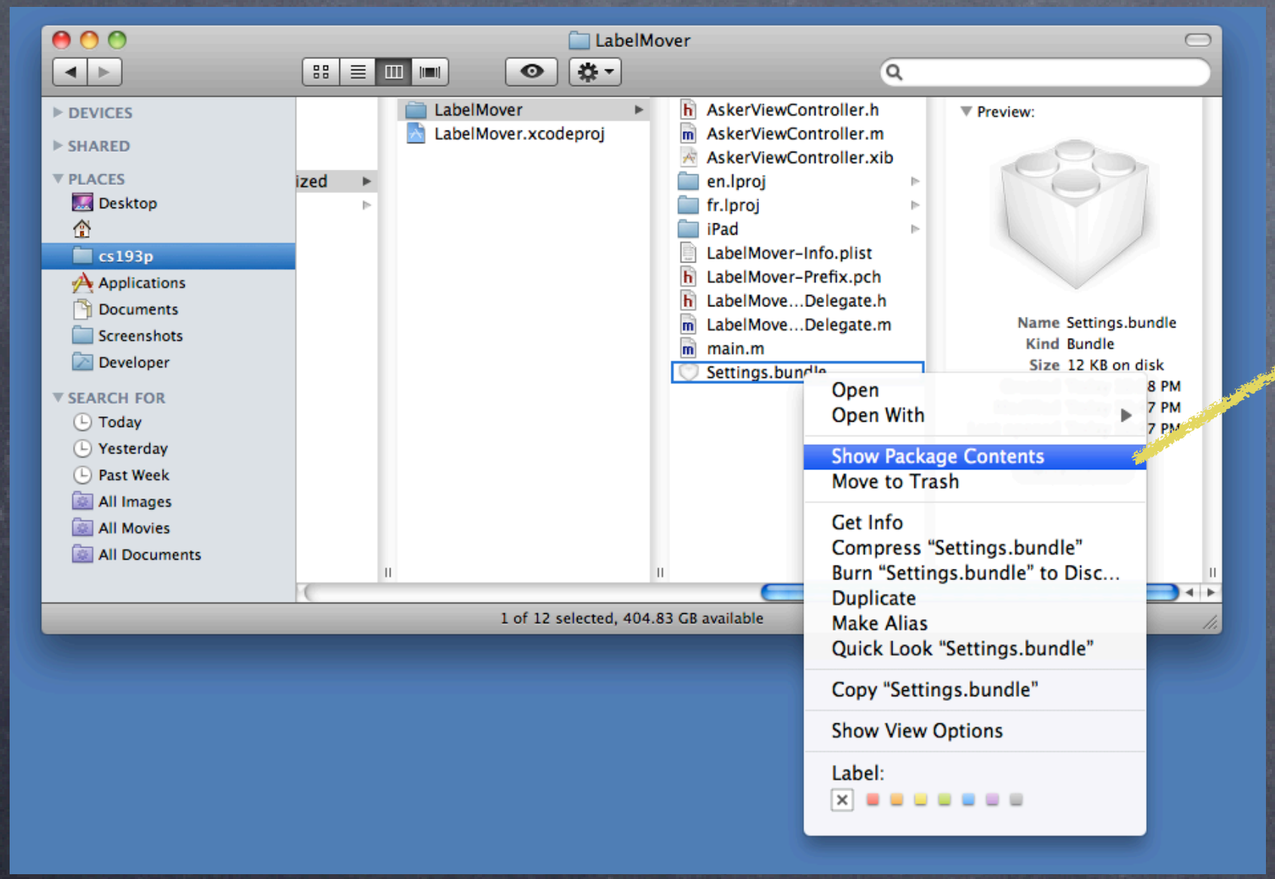

KitchenSink.xcodeproj
Run
Stop
KitchenSink 〉 iPad 5.1 Simulator
Scheme
Breakpoints
Build KitchenSink: Succeeded | Today at 12:46 PM
No Issues
Editor
View
Organizer
KitchenSink
1 target, iOS SDK 5.1
KitchenSink
MainStoryboard.storyboard
ViewController.h
ViewController.m
CMMotionManager+Shared.h
CMMotionManager+Shared.m
AskerViewController.h
AskerViewController.m
Strings
Supporting Files
Frameworks
Products
Build Rules
Choose a template for your new file:
iOS
Cocoa Touch
C and C++
User Interface
Core Data
Resource
Other
Mac OS X
Cocoa
C and C++
User Interface
Core Data
Resource
Other
GPX File
Settings Bundle
Property List
Rich Text File
Strings File
Settings Bundle
Bundle for specifying an iOS Application's settings.
Cancel
Previous
Next
Add Target
Validate Settings
Choose New File... from the File menu, then pick Settings Bundle from the Resource grouping.

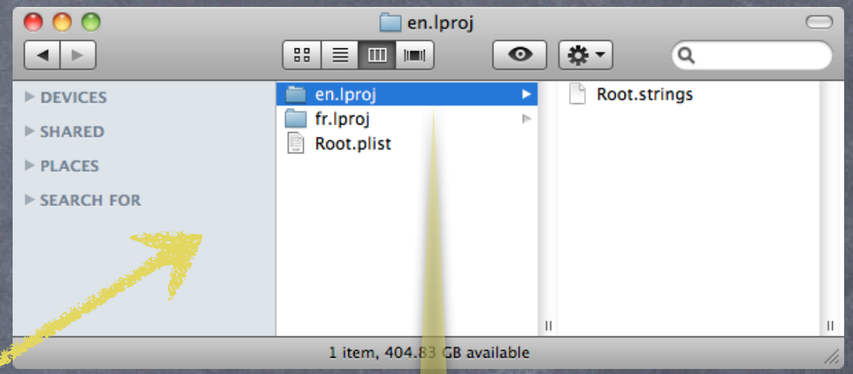

KitchenSink.xcodeproj — Root.plist
Run
Stop
KitchenSink 〉 iPad 5.1 Simulator
Scheme
Breakpoints
Build KitchenSink: Succeeded | Today at 12:46 PM
No Issues
Editor
View
Organizer
KitchenSink 〉 KitchenSink 〉 Settings.bundle 〉 Root.plist 〉 No Selection
KitchenSink
1 target, iOS SDK 5.1
KitchenSink
Settings.bundle
en.lproj
Root.strings
Root.plist
MainStoryboard.storyboard
ViewController.h
ViewController.m
AskerViewController.h
AskerViewController.m
Supporting Files
Frameworks
Products
Key
Type
Value
Preference Items
Array
(4 items)
Item 0 (Group - Group)
Diction...
(2 items)
Title
String
Group
Type
String
Group
Item 1 (Text Field - Name)
Diction...
(8 items)
Autocapitalization Style
String
None
Autocorrection Style
String
No Autocorrection
Default Value
String
Text Field Is Secure
Boolean
NO
Identifier
String
name_preference
Keyboard Type
String
Alphabet
Title
String
Name
Type
String
Text Field
Item 2 (Toggle Switch - Enabled)
Diction...
(4 items)
Default Value
Boolean
YES
Identifier
String
enabled_preference
Title
String
Enabled
Type
String
Toggle Switch
Item 3 (Slider)
Diction...
(7 items)
Default Value
Number
0.5
Identifier
String
slider_preference
Maximum Value
Number
Max Value Image Filename
String
Minimum Value
Number
0
Min Value Image Filename
String
Type
String
Slider
Strings Filename
String
Root
A sort of "example" settings bundle will be created for you. You can edit it from here. Check the documentation for all the possibilities.
It is possible to have multiple "pages" of settings. See documentation for details.


KitchenSink.xcodeproj — Root.strings
Run
Stop
KitchenSink 〉 iPad 5.1 Simulator
Scheme
Breakpoints
Build KitchenSink: Succeeded | Today at 12:46 PM
No Issues
Editor
View
Organizer
KitchenSink 〉 KitchenSink 〉 Settings.bundle 〉 en.lproj 〉 Root.strings 〉 No Selection
KitchenSink
1 target, iOS SDK 5.1
KitchenSink
Settings.bundle
en.lproj
Root.strings
Root.plist
MainStoryboard.storyboard
ViewController.h
ViewController.m
AskerViewController.h
AskerViewController.m
Supporting Files
Frameworks
Products
/* A single strings file, whose title is specified in your preferences schema. The strings files provide the localized content to display to the user for each of your preferences. */
"Group" = "Group";
"Name" = "Name";
"none given" = "none given";
"Enabled" = "Enabled";
Note the en.lproj.
Yes, settings are localizable, but it's not very well supported in Xcode.
Also, note Strings Filename (Root.strings) here. This is the name of the .strings file to localize this page of the settings.

Settings

- Unfortunately, localization of settings is a bit of a pain
  You have to find the `Settings.bundle` in your Finder and create `.lproj` directories yourself.
  Each `.lproj` directory should contain a `.strings` file for each screen in your settings.





Copy and paste `en.lproj` to other languages (like `fr.lproj`), then edit the `Root.strings` (or other `.strings` files) inside for each language.

Demo

- Kitchen Sink
  Settings

Coming Up

- Final Projects
  - One week to go!
  - Proposal time is over (you can still get feedback from us, but move ahead)
- Alternate Final Presentation
  - Next Tuesday
  - Let me know TODAY (if you haven't already) if you want to do this
  - Submit slides by Monday at the latest