Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Coming Up

- Wednesday
  - Alternate Final Presentation.
  - If you are using Alternate Presentation time, submit your Keynote by noon tomorrow (Tuesday).
  - Submit the slides using the normal submit script (submit again with code by Sunday).
  - We will have a "live demo testing" opportunity on Wednesday as well, so bring your demo device.
- Friday
  - No Section.
- Sunday
  - Final Project Due (by midnight).
  - Don't forget to submit your Keynote slides along with!
- Final
  - A week from Thursday at 12:15pm to 3:15pm in this room.
  - Presentation is required.
  - Presentation time limit is 2.5 minutes (150 seconds) and must be 1280x720 aspect ratio.
  - Presentation order is random (no exceptions).

Today

- Localization
  Internationalization really.
- Settings
  Adding UI to the Settings application.
- Demo
  Internationalizing Photomania.
  Adding a Bouncer setting.

Internationalization

- Two steps to making international versions of your application
  - Internationalization (i18n)
  - Localization (l10n)
- Internationalization
  - This is a process of making strings externally editable (from storyboard or code).
  - It also involves using certain "formatting" classes for things like dates, numbers, etc.
  - You (the developer) get to do this work.
- Localization
  - A process of editing those externalized strings (and then QA'ing the result) for a given language.
  - You usually hire a localization company to do this work.

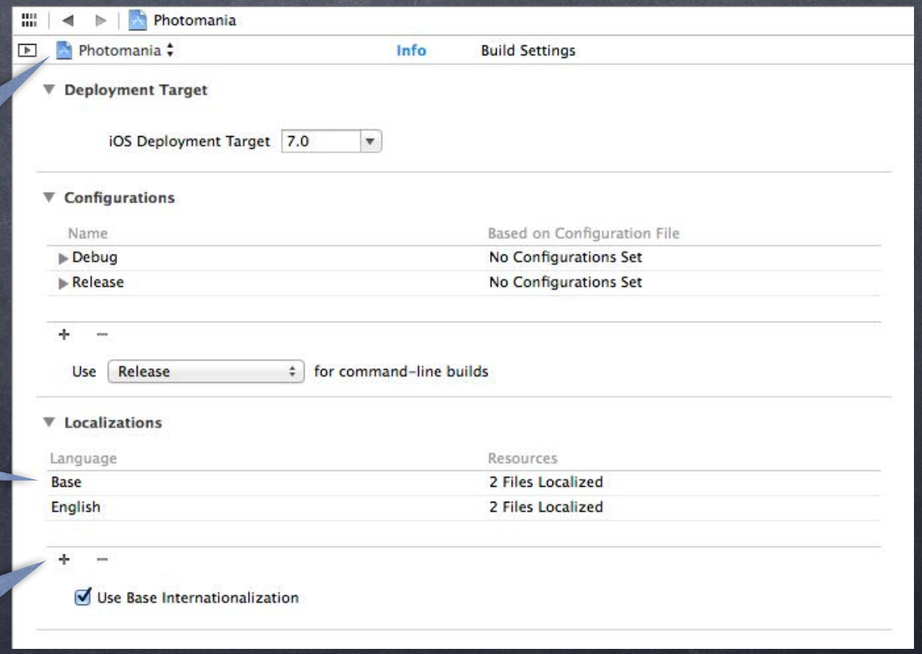
Internationalization

- Storyboards are localized by changing its strings only
  And we rely on Autolayout to make it all look nice.
- First step though: Registering Localizable Languages
  Go to the Project pane in Xcode (top in Navigator), then Info tab to add Localizations.
  If you click "Use Base Internationalization" the strings in your storyboards will be extracted into editable `.strings` files (one for each language).

You must inspect the project itself here, not the Target you build.

"Base" is the "localization" where storyboards live that are localizable using only `.strings` files (hopefully this is all storyboards).

Click this + to add more languages that you intend to support.

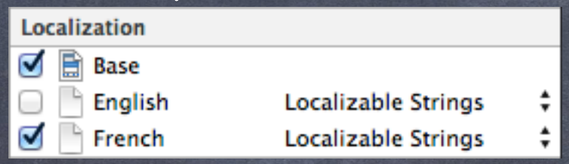


Localizing Storyboards

- Storyboards in Navigator will now have localizations
  - Send the `.strings` files out to localizers to translate the strings.
  - Localizers appreciate a demo of your application in your Base language.
  - Or at least send them the storyboards so they can get context.

Navigator

File Inspector



Internationalization

- What about strings not in storyboards?

  i.e., literal strings @"string"

  Replace them with a variant of NSLocalizedString …

```
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)

  and NSLocalizedString() (defaultValue is key; mainBundle; table Localizable.strings)

  Example: Change @"hello" to NSLocalizedString(@"hello", @"Greeting at start of application.")

- What these macros do …

  They send this method to [NSBundle mainBundle] (or the specified bundle if macro takes one) …

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

Localization

- Generating `.strings` files with `genstrings`

  Once you have used `NSLocalizedString` and its variants to eliminate literal strings …

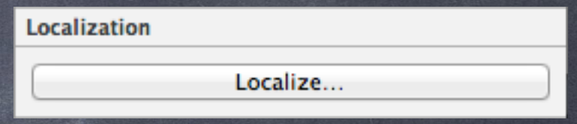
  You can use the command line utility `genstrings` to generate `.strings` files from `.m` files.

  ```
  > cd <directory where all your .m files are>
  > genstrings *.m
  ```

  Example: `NSLocalizedString(@"hello", @"Greeting at start of application.")`

  … would generate an entry in `Localizable.strings` which looks like this …

  ```
  /* Greeting at start of application. */
  "hello" = "hello";
  ```



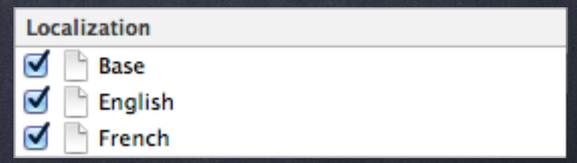
- Drag the `.strings` into Xcode and then inspect to Localize

  Hit the button "Localize" in the File Inspector on the strings file or storyboard.

  You can then pick languages for which there is a localization set up for your application.

  (As per the first slide on this topic.)

  E.g., French localizers would change entry to `"hello" = "bonjour"`.



Bundles

- 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 `.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.

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.

- Build Clean

  If changes you make to .strings files don't seem to be appearing when you run … try Build Clean. Usually this is not necessary, but it's something to try if things get out of sync.

Locales

- Formats

  Dates and numbers are written in different formats in different locales.

- Locale

  Locale is different from language.

  The `NSLocale` class encapsulates the locale the user has chosen in Settings.

  It knows all about date and number formats (independent of the language that is currently set).

```
+ (NSLocale *)currentLocale;
+ (NSLocale *)autoupdatingCurrentLocale; // watch NSCurrentLocaleDidChangeNotification
```

  Usually you don't need to access this directly because you'll use a formatter which is looking at it.

NSNumberFormatter

- Lots going on here. Check out the documentation.
  - But we'll look at two simple cases …
- Displaying numbers
  - Shouldn't really use `[NSString stringWithFormat:@"%g"]` for user-visible floats.
  - Instead use this NSNumberFormatter class method …

```
+ (NSString *)localizedStringFromNumber:(NSNumber *)number
                            numberStyle:(NSNumberFormatterStyle)style
```

  - Example styles: NSNumberFormatter`DecimalStyle` or `CurrencyStyle` or even `SpellOutStyle`
- Parsing numbers
  - Don't use `intValue` to parse a number typed in by the user, use …

```
NSNumberFormatter *formatter = [[NSNumberFormatter alloc] init];
[formatter setNumberStyle:NSNumberFormatterDecimalStyle];
NSNumber *parsedNumber = [formatter numberFromString:userInputtedString];
```

  - Note that this will return `nil` if a number of the proper format is not found.
  - That can be valuable to differentiate from the user entering "zero" for example.

NSDateFormatter

- Dates are rather complicated to display properly

  If you are presenting dates to the user, familiarize yourself with these concepts …

  Calendars. Not all locales use the Gregorian calendar that we do. `NSCalendar`.

  Date Components, e.g., what is a "month" (calendar dependent)? `NSDateComponents`.

  And if you have in mind something like MM/DD/YYYY, check out this method first …

```
+ (NSString *)dateFormatFromTemplate:(NSString *)template
                              options:(NSUInteger)options
                               locale:(NSLocale *)locale;
```

- Simple date formatting

  At least use this `NSDateFormatter` class method …

```
+ (NSString *)localizedStringFromDate:(NSDate *)date
                            dateStyle:(NSDateFormatterStyle)dateStyle
                            timeStyle:(NSDateFormatterStyle)timeStyle;
```

  Example styles: `NSDateFormatterShortStyle` or `MediumStyle` or `LongStyle` or `FullStyle`

NSString

- Searching in strings

  Do not use plain `rangeOfString:` if you are looking around in user-inputted strings.

  Instead, use this …

```
+ (NSRange)rangeOfString:(NSString *)useEnteredSubstring
                 options:(NSStringCompareOptions)options // e.g. case-insensitively
                   range:(NSRange)rangeToSearchIn
                  locale:(NSLocale *)locale;
```

  … especially if you are searching case-insensitively, since this concept is locale-specific.

UIImage

- The method `imageNamed:` does the right thing!
  - It searches inside the `.lproj`'s to find images.

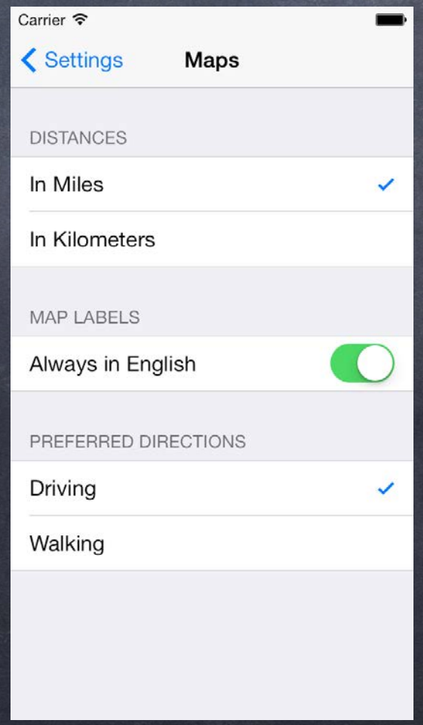
Demo

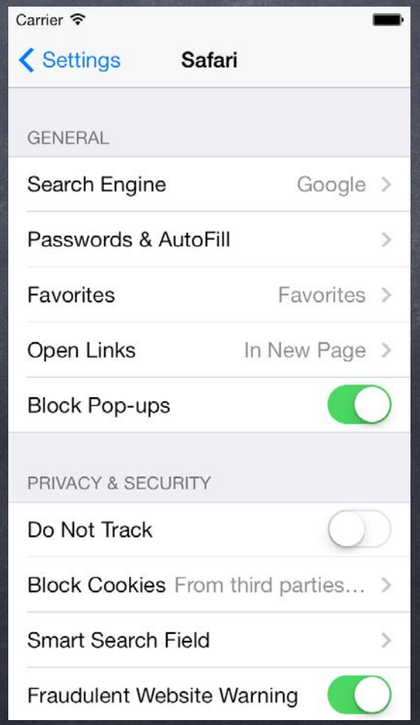
- Photomania

  Let's internationalize it.

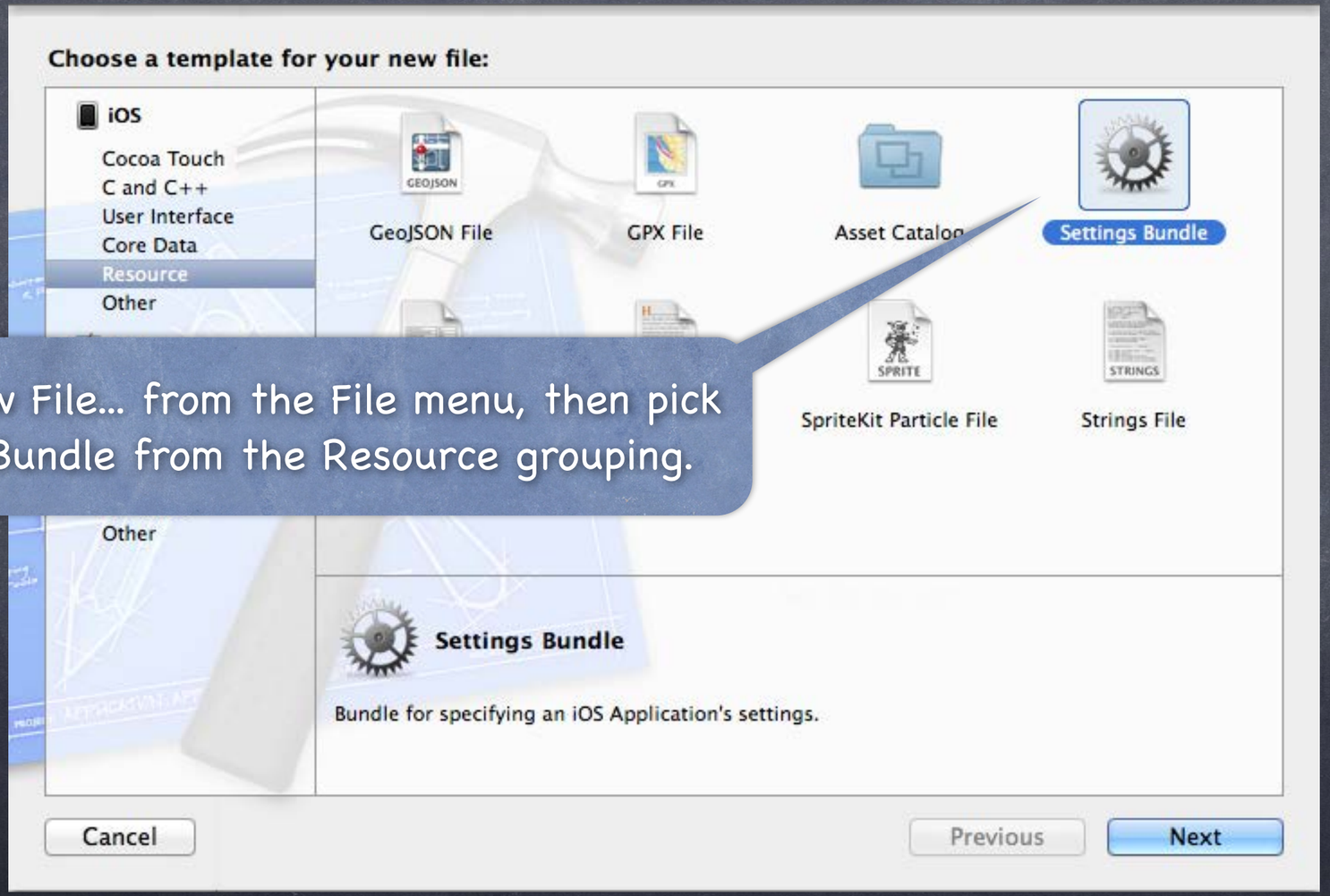
Settings

- 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 defaults in a property list file.





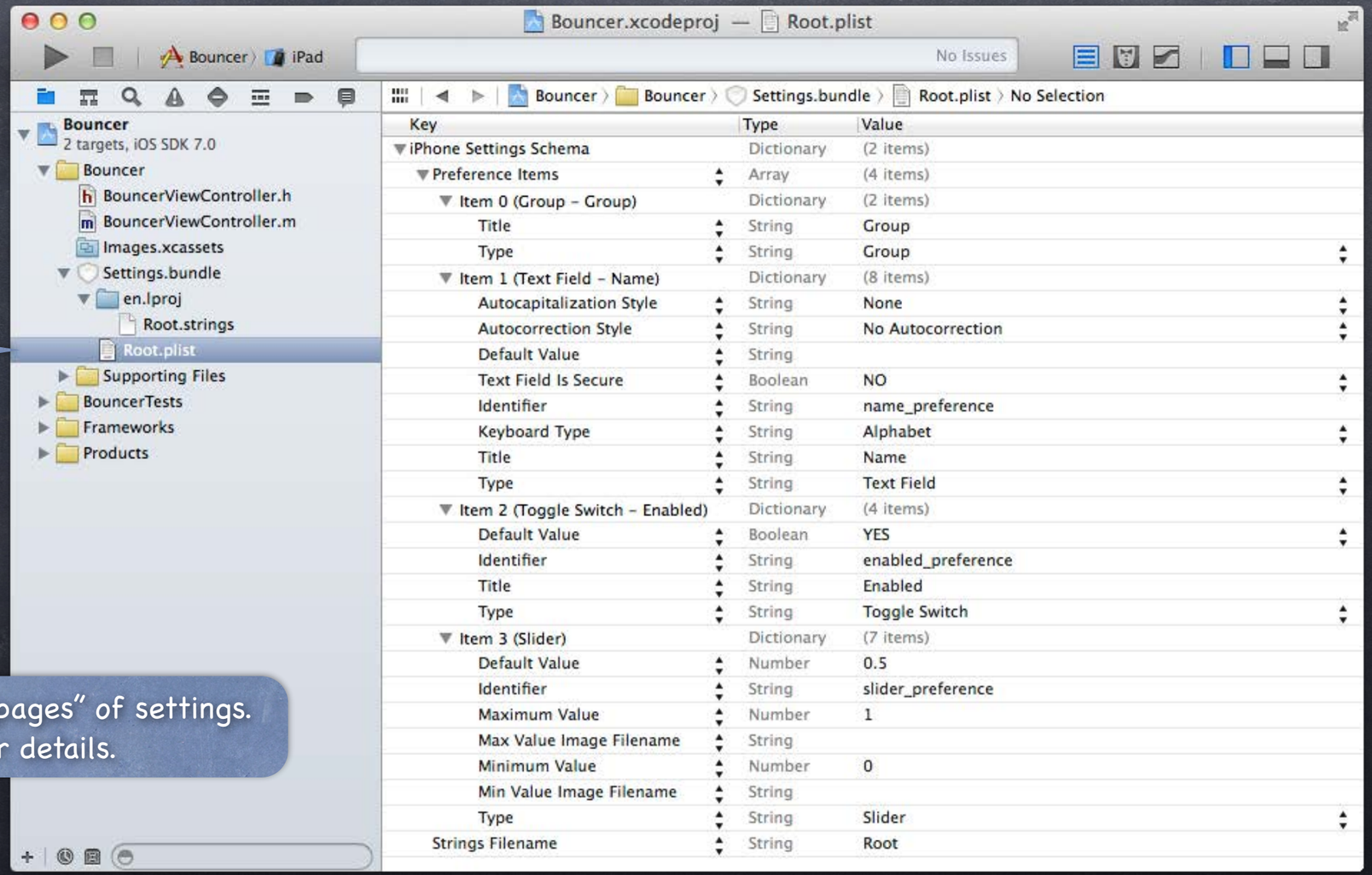
Settings



Choose New File... from the File menu, then pick Settings Bundle from the Resource grouping.

Settings

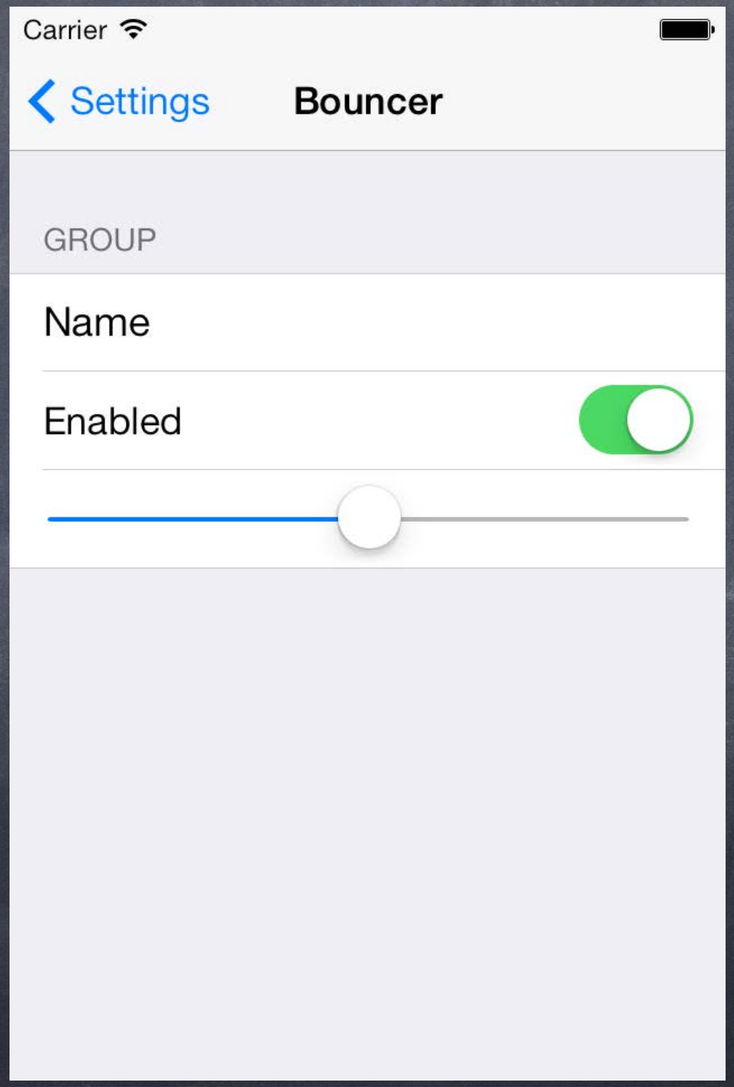
A sort of "example" settings bundle will be created for you. You can edit it by clicking here. Check the documentation for all the possibilities.



It is possible to have multiple "pages" of settings. See documentation for details.

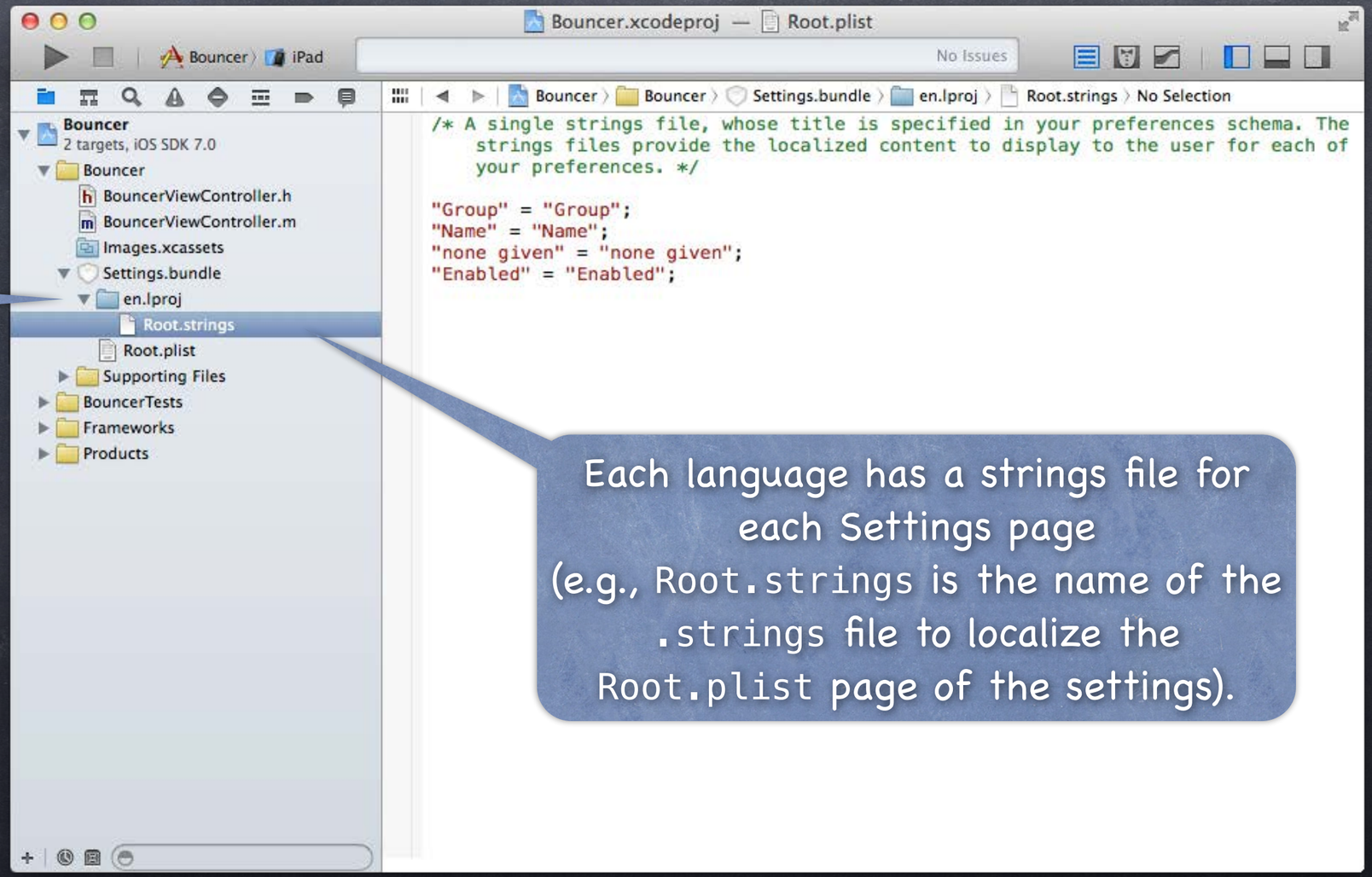
Settings

The sample from the previous slide would result in a Settings UI like this.



Settings

Note the `en.lproj`. Yes, settings are localizable, but it's not very well supported in Xcode.

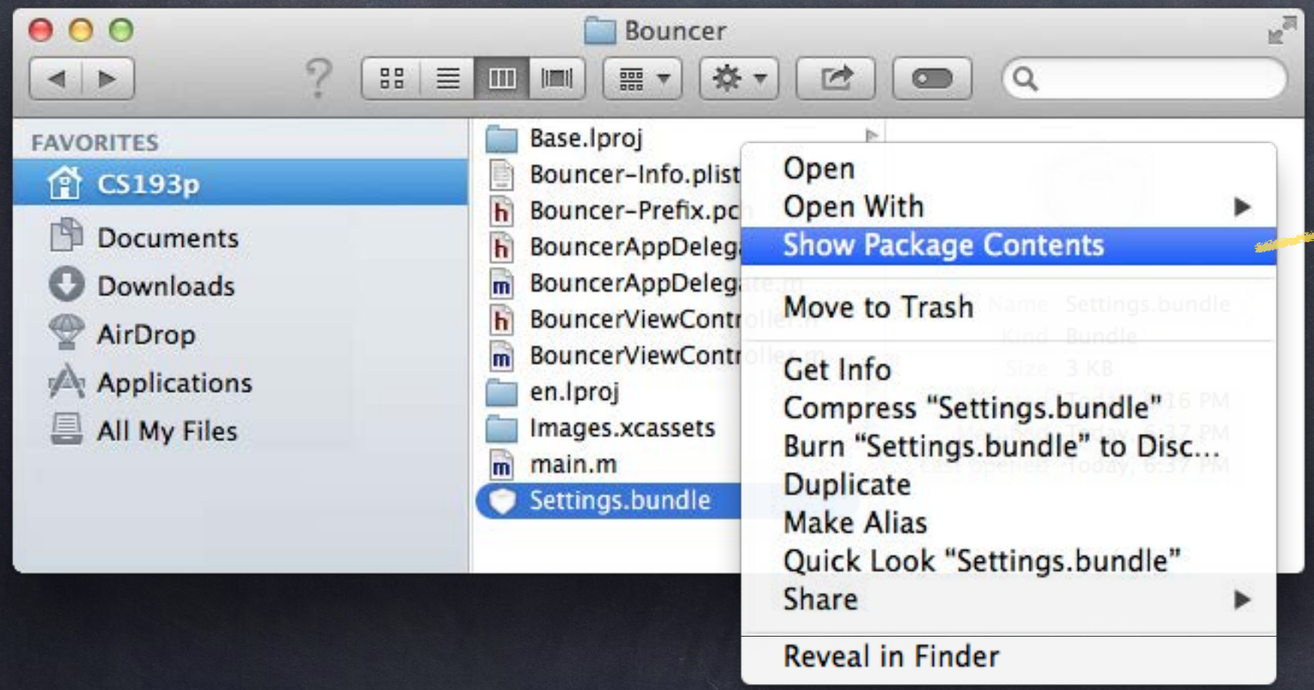


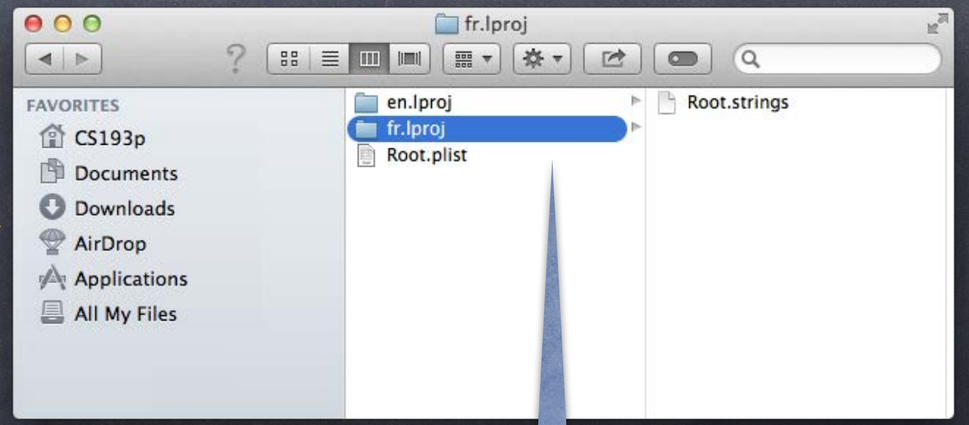
Each language has a strings file for each Settings page (e.g., `Root.strings` is the name of the `.strings` file to localize the `Root.plist` page of the settings).

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

- Bouncer

  Allow setting the Elasticity from Settings.

Coming Up

- Wednesday
  - Alternate Final Presentation.
  - If you are using Alternate Presentation time, submit your Keynote by noon tomorrow (Tuesday).
  - Submit the slides using the normal submit script (submit again with code by Sunday).
  - We will have a "live demo testing" opportunity on Wednesday as well, so bring your demo device.
- Friday
  - No Section.
- Sunday
  - Final Project Due (by midnight).
  - Don't forget to submit your Keynote slides along with!
- Final
  - A week from Thursday at 12:15pm to 3:15pm in this room.
  - Presentation is required.
  - Presentation time limit is 2.5 minutes (150 seconds) and must be 1280x720 aspect ratio.
  - Presentation order is random (no exceptions).