Smaller View Controllers

Smaller View Controllers

OR

Smaller View Controllers

OR

The Parts of (Programming) Speech

English

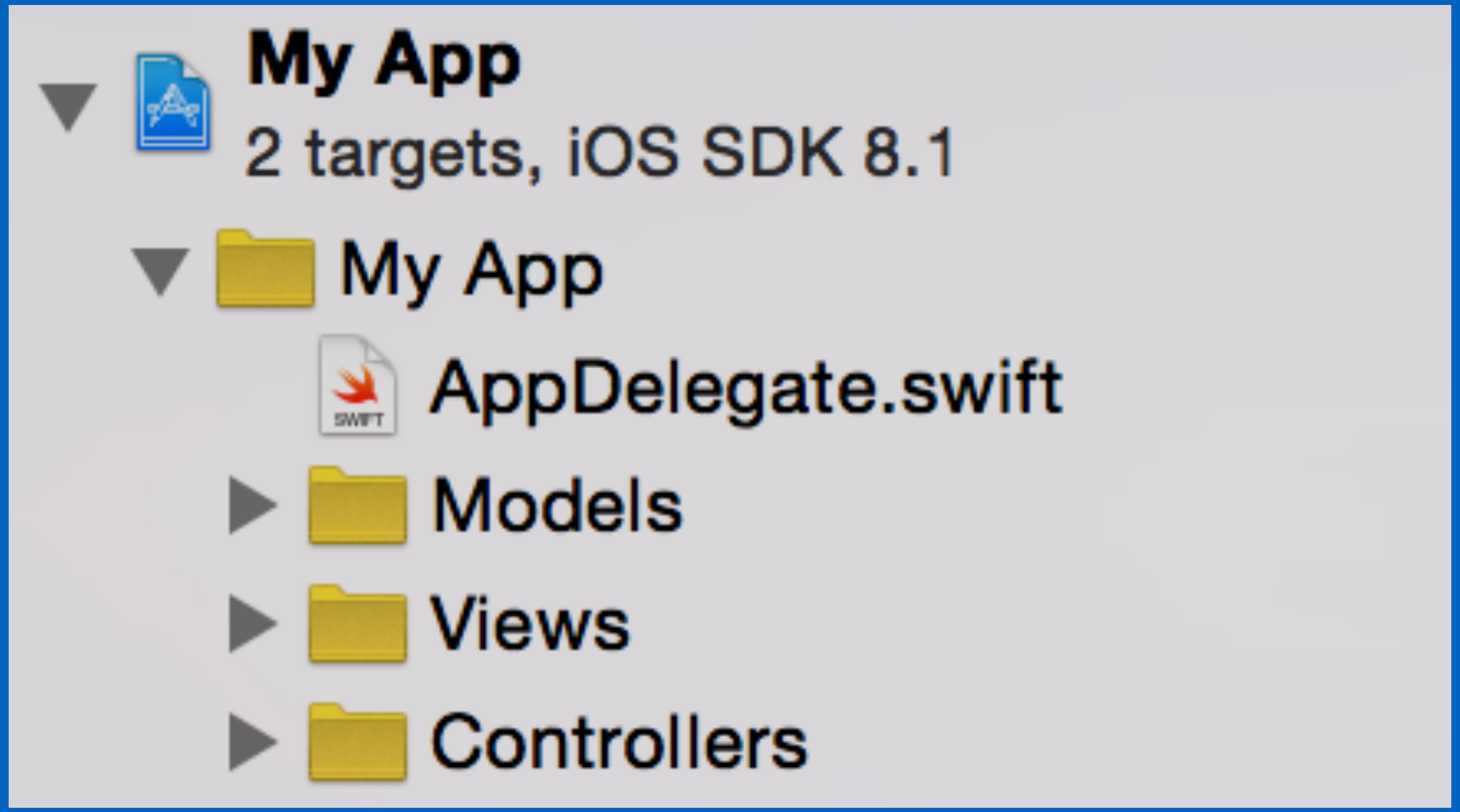
Nouns
Verbs
Adjectives
Adverbs
Pronouns
Prepositions
Interjections
Conjunctions

Typical Cocoa

Models
Views
Controllers

… Maybe some categories?

Model View Controller

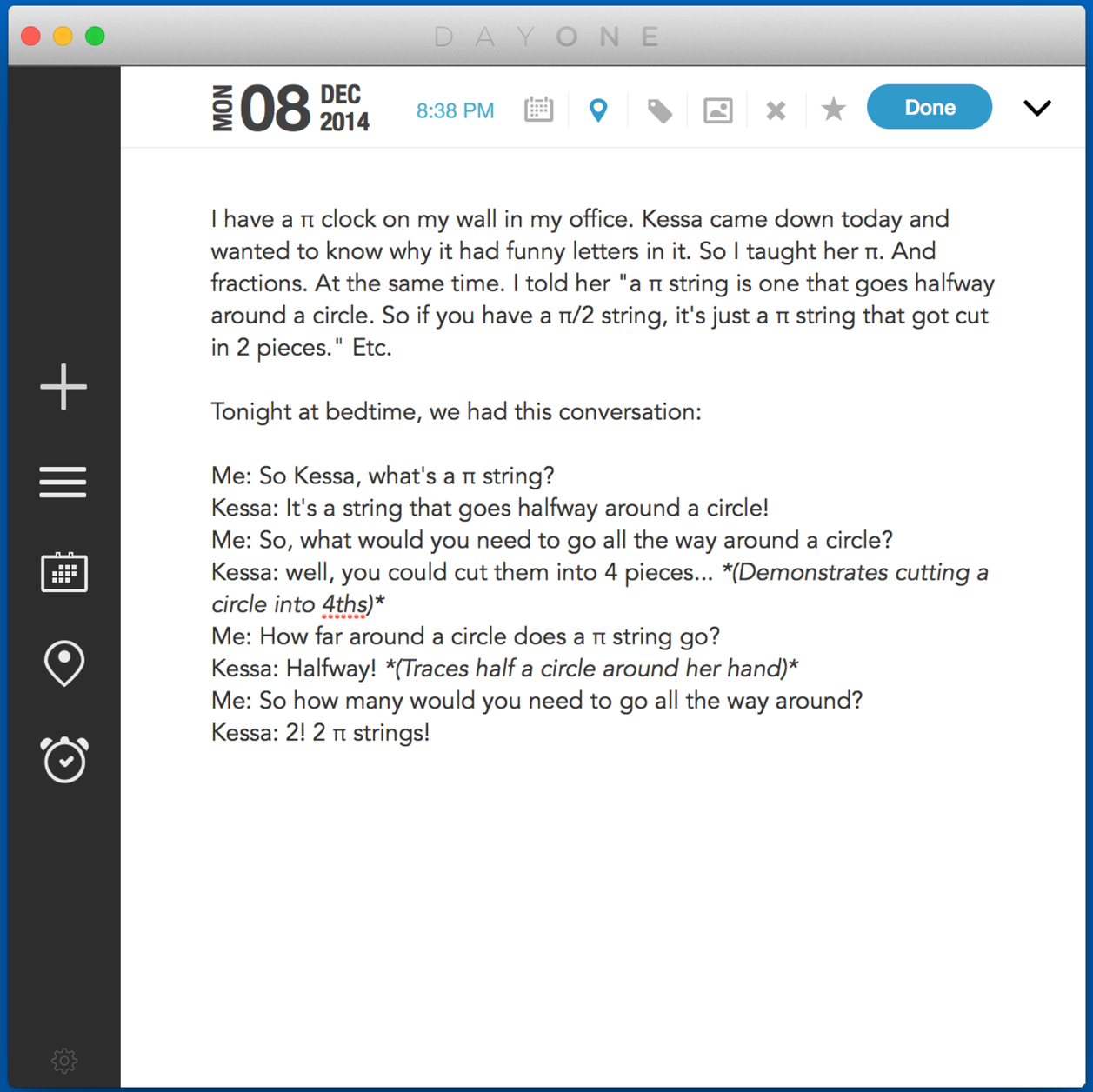


Models are easy

Views are easy

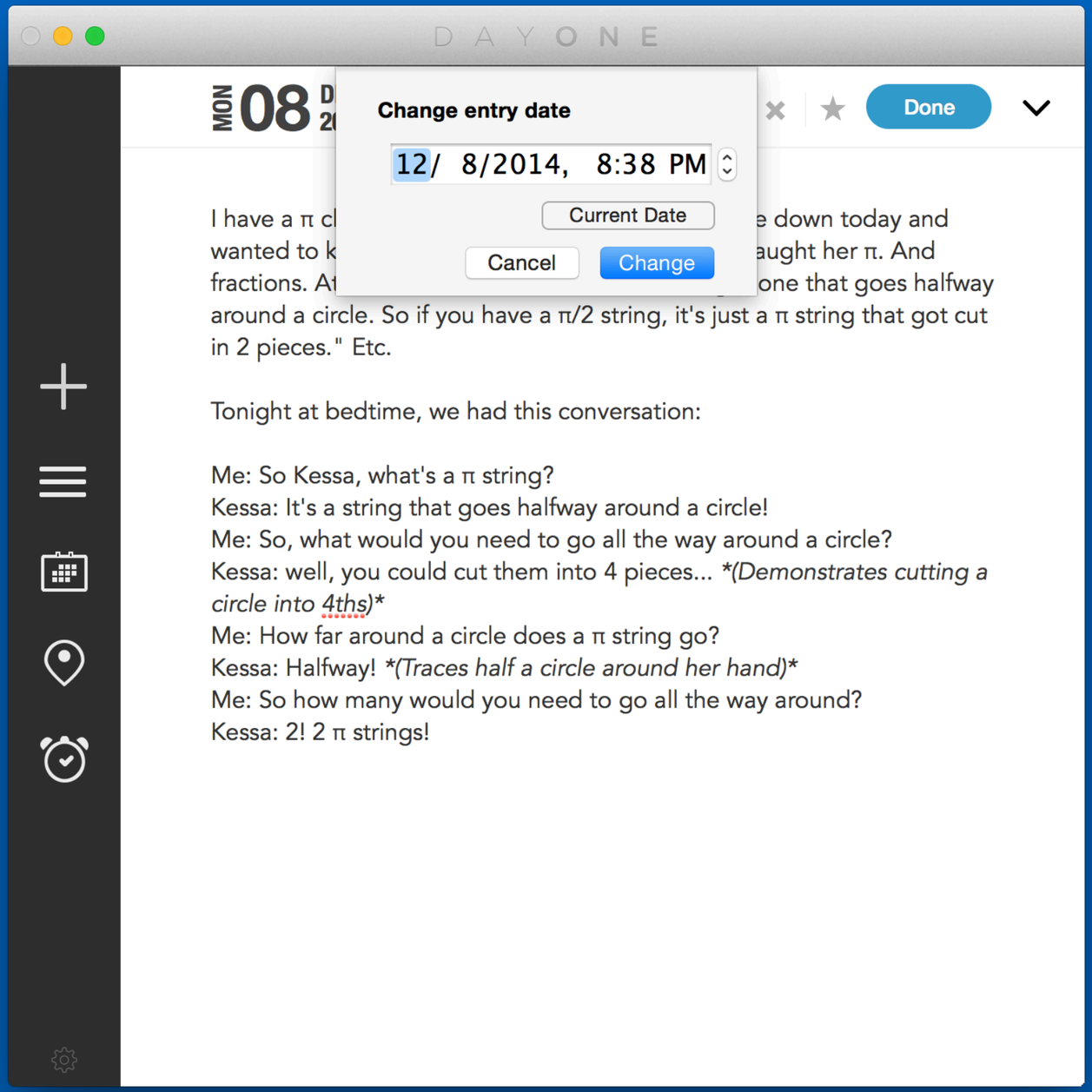
Custom views are fun!

Massive View Controllers



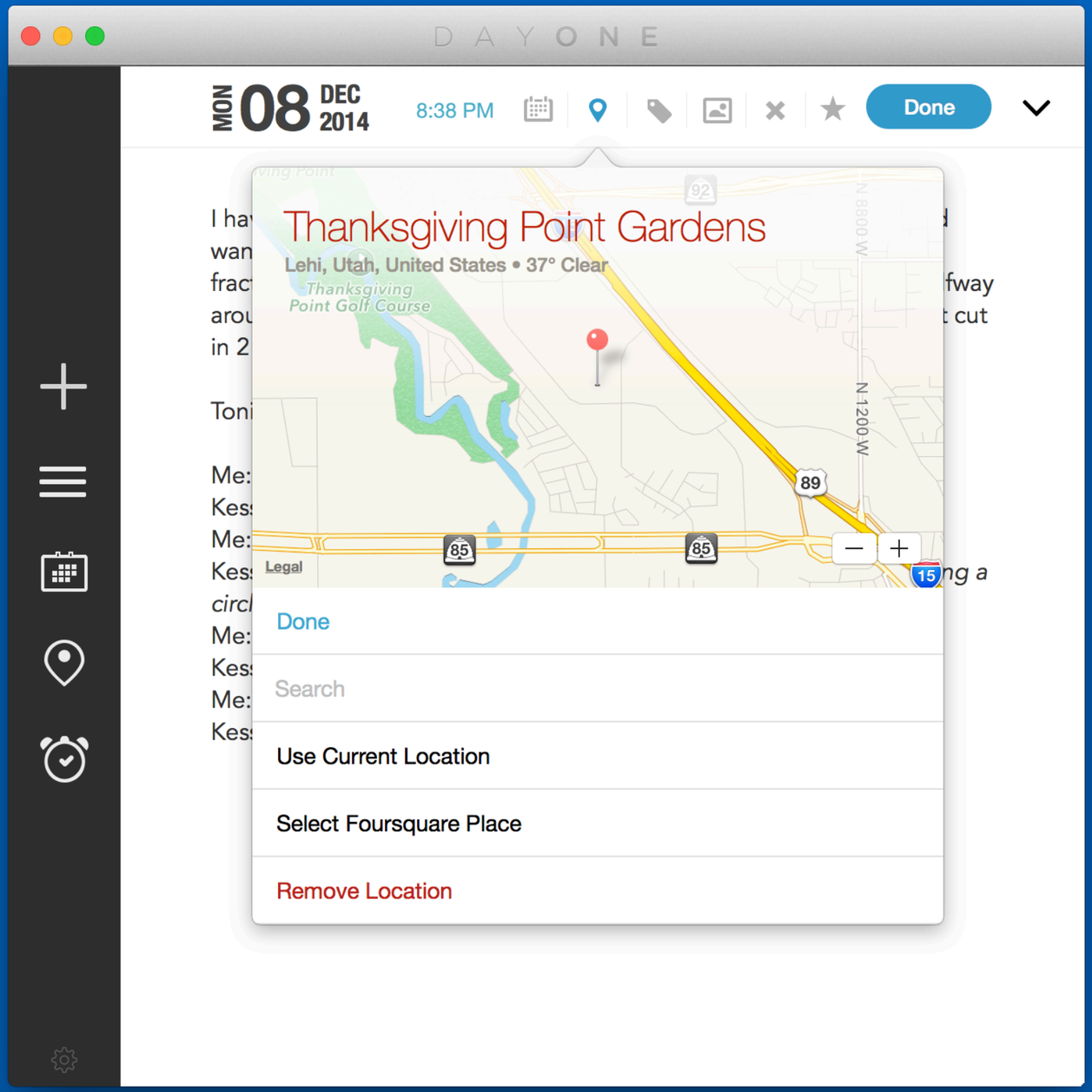
`EntryViewController`

- Editing Text



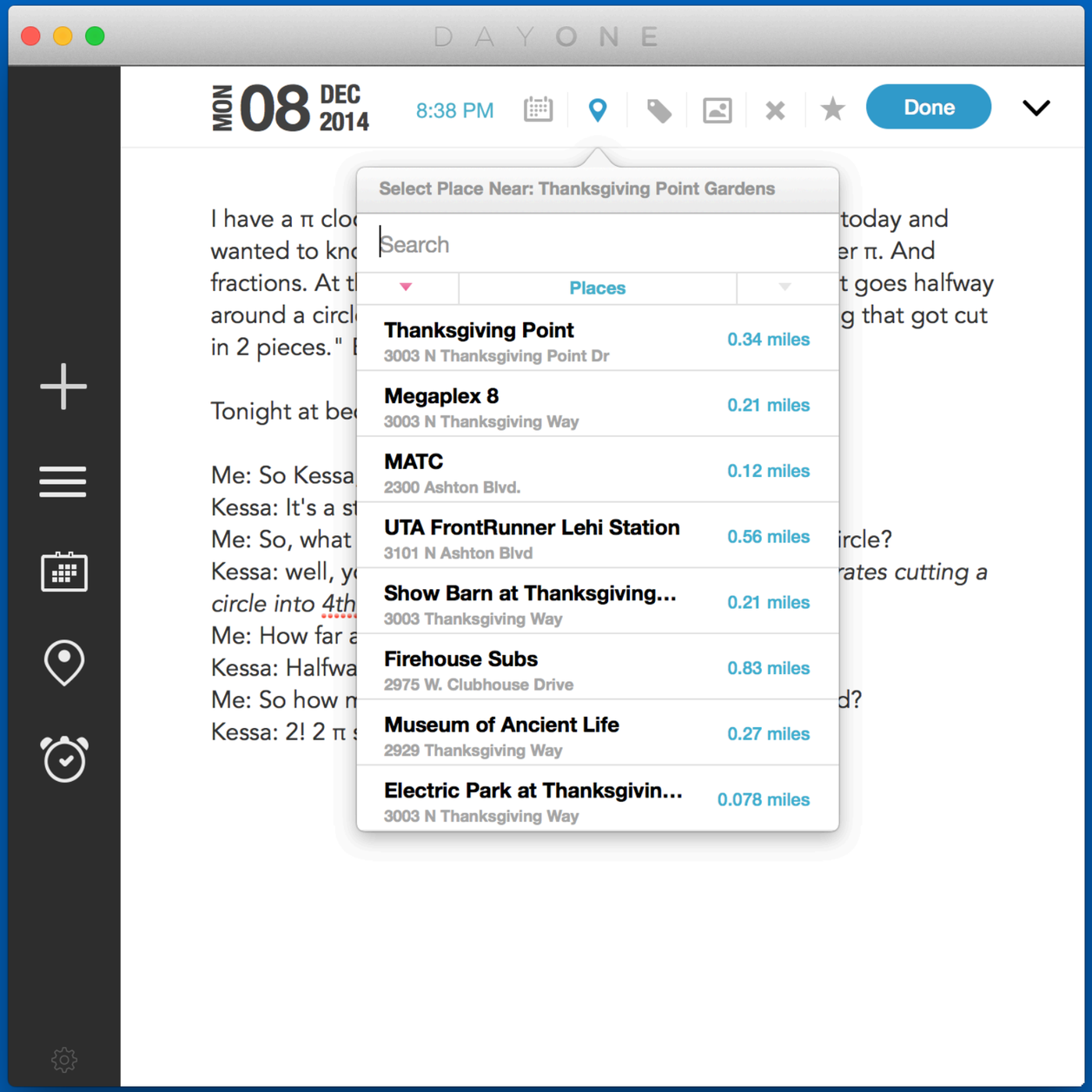
`EntryViewController`

- Editing Text
- Date



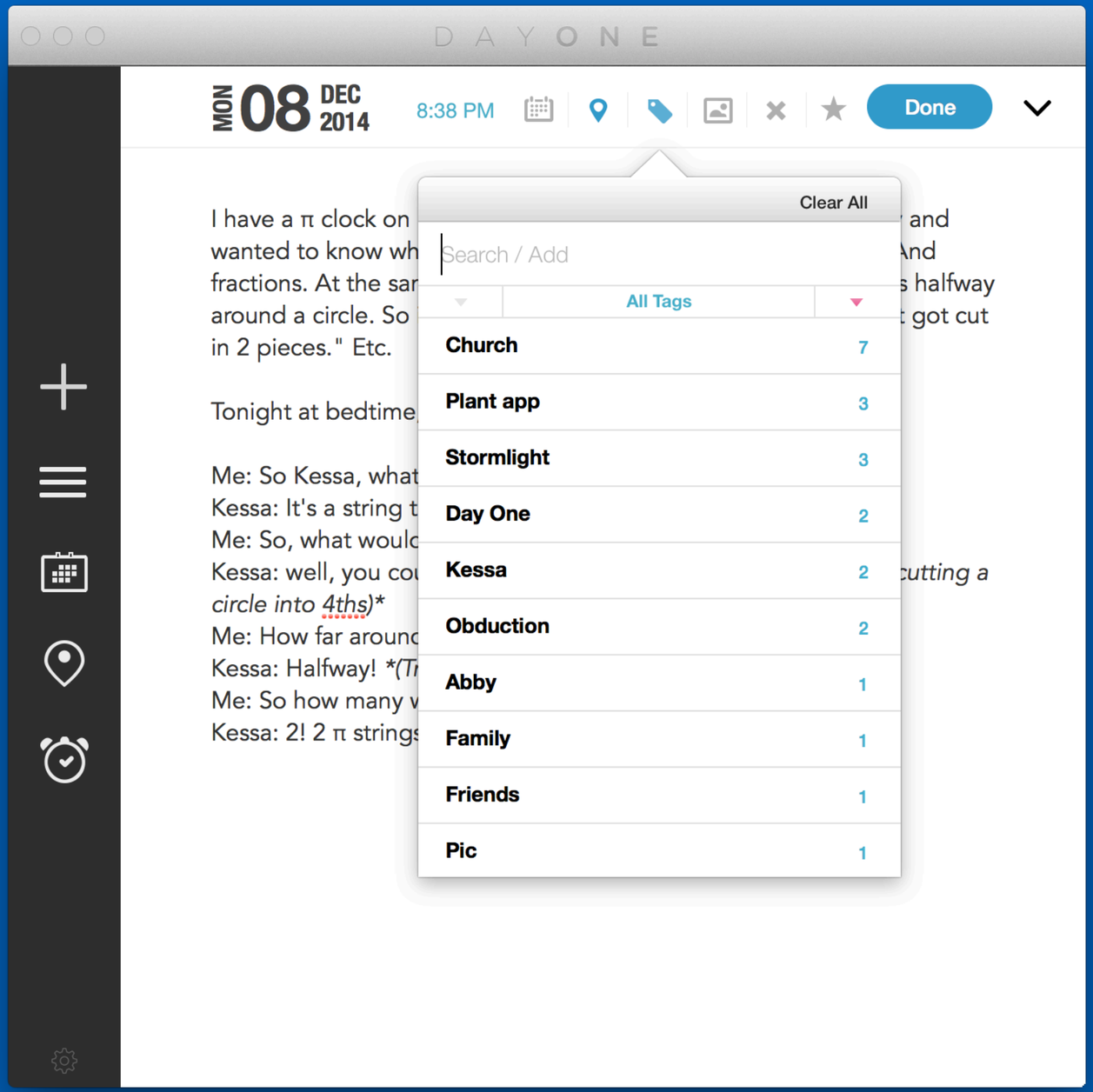
`EntryViewController`

- Editing Text
- Date
- Location



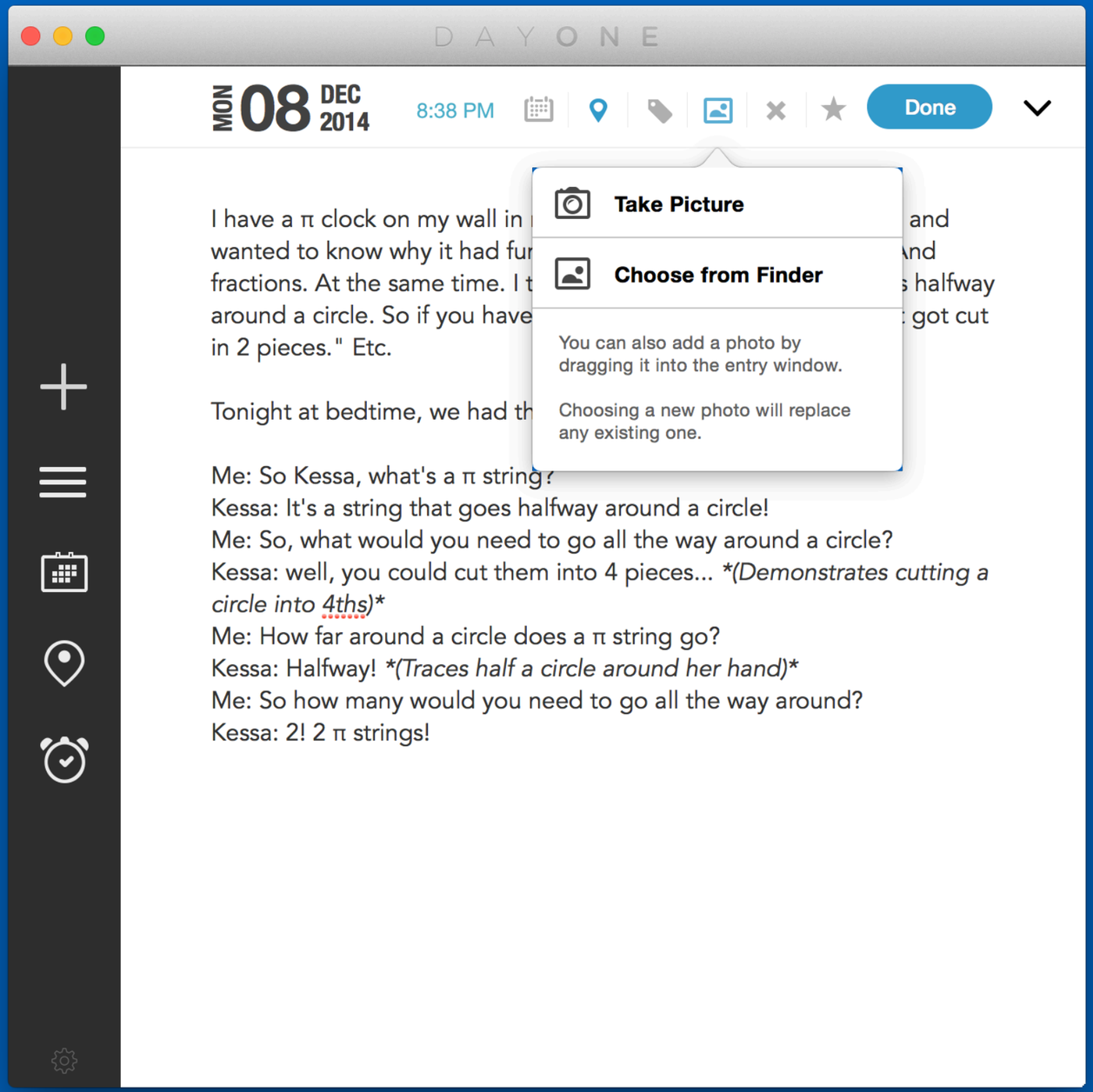
`EntryViewController`

- Editing Text
- Date
- Location
- Foursquare



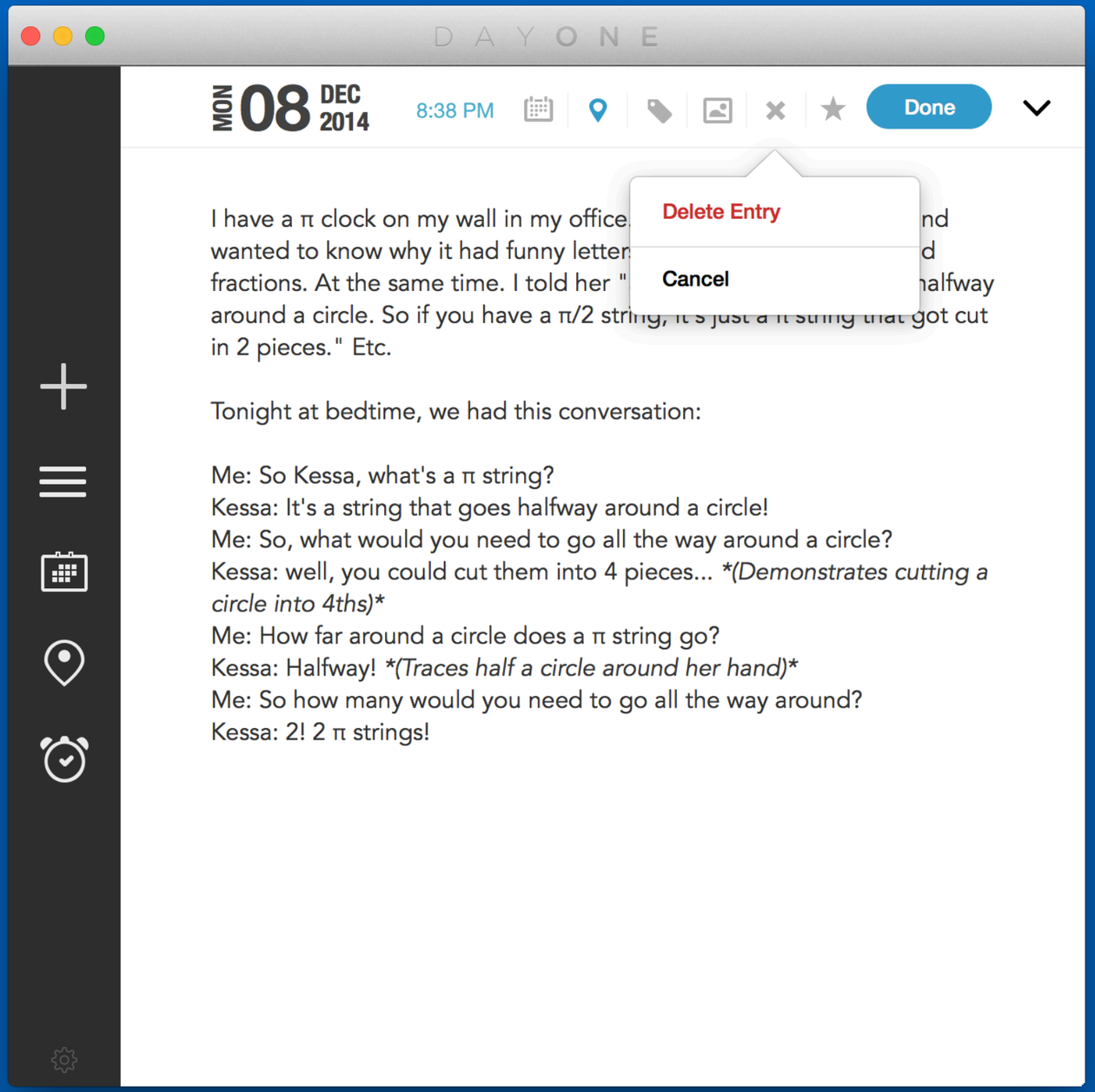
`EntryViewController`

- Editing Text
- Date
- Location
- Foursquare
- Tags



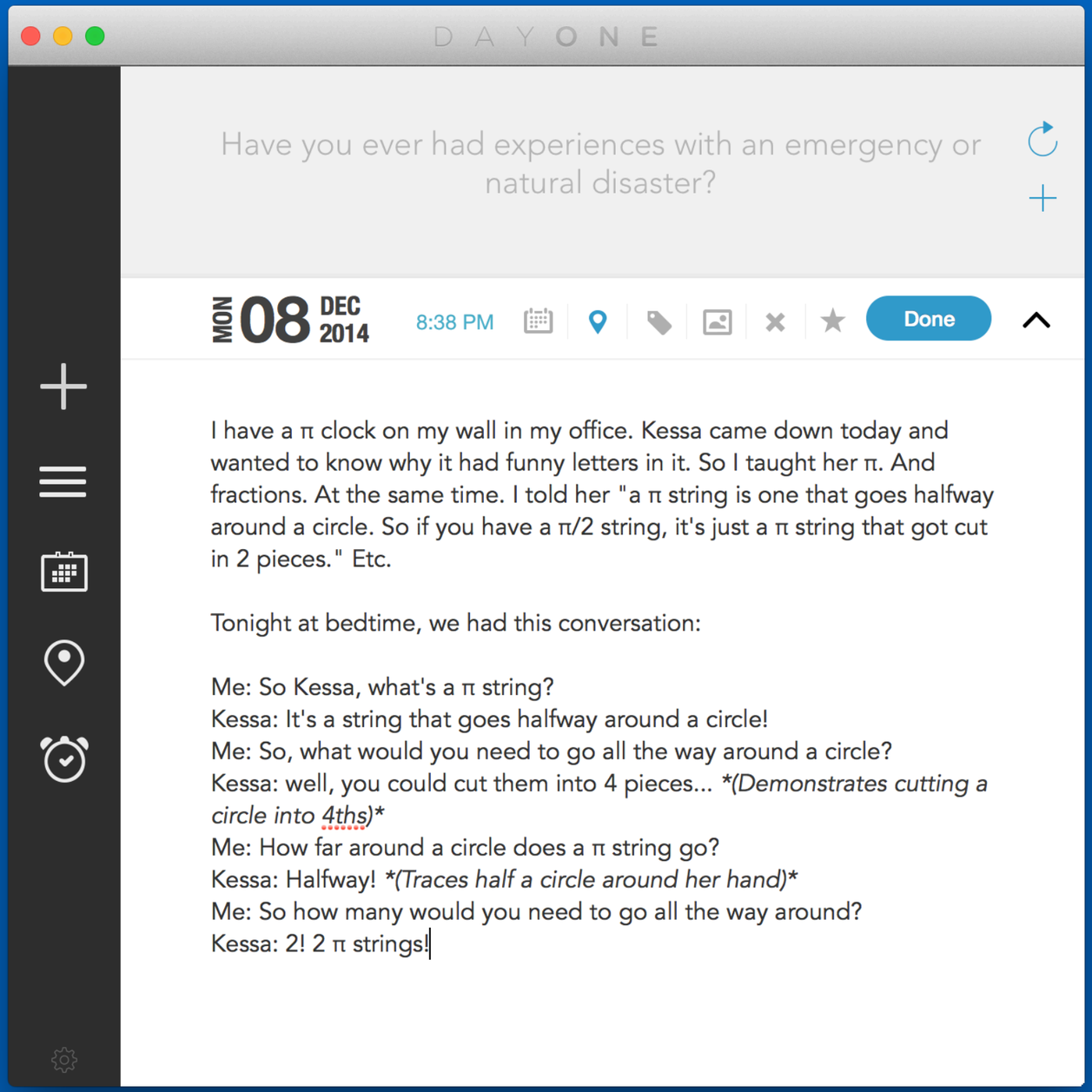
`EntryViewController`

- Editing Text
- Date
- Location
- Foursquare
- Tags
- Photos



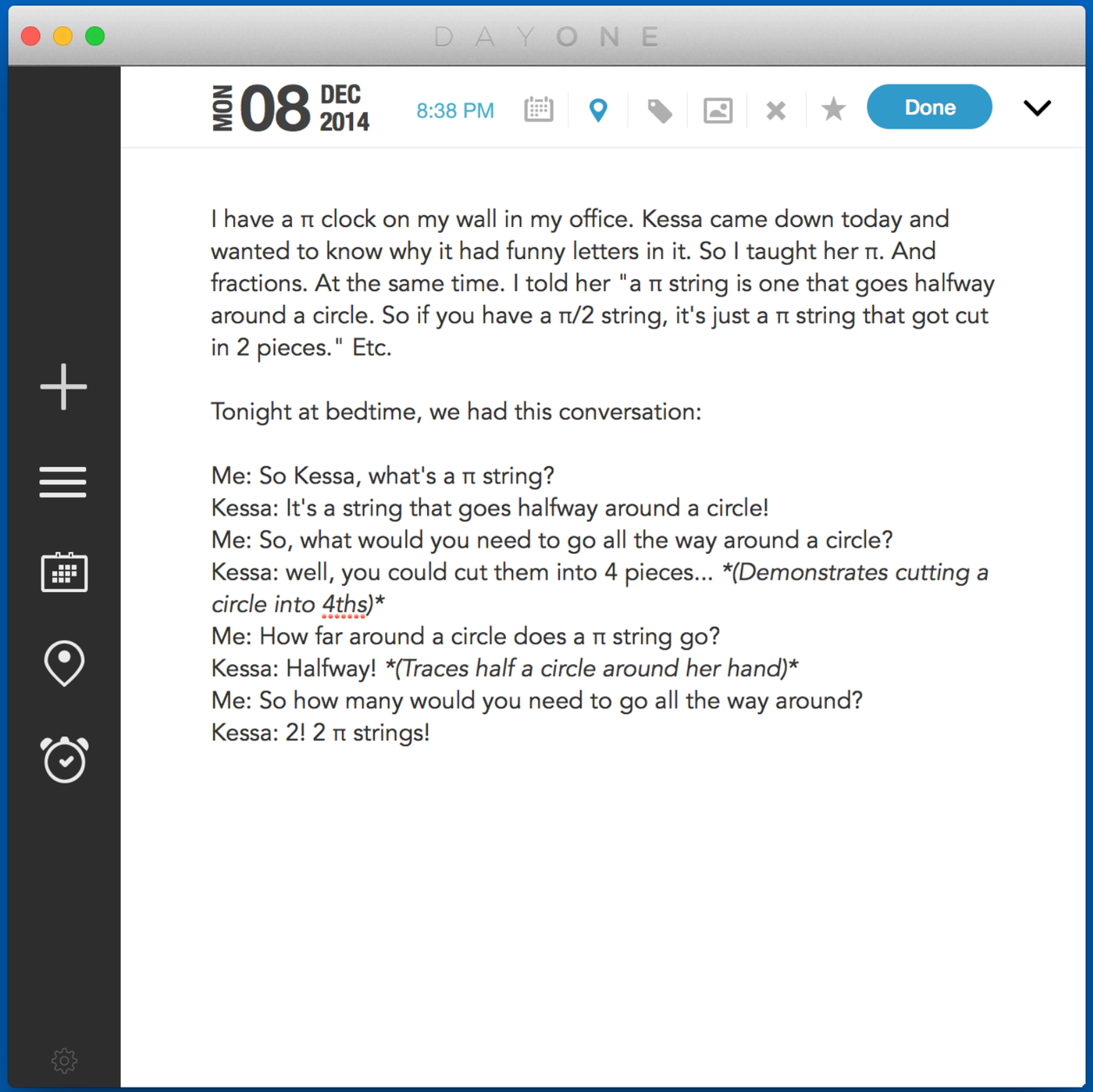
`EntryViewController`

- Editing Text
- Date
- Location
- Foursquare
- Tags
- Photos
- Deleting Entries



`EntryViewController`

- Editing Text
- Date
- Location
- Foursquare
- Tags
- Photos
- Deleting Entries
- Inspirational Quotes



`EntryViewController`

All managed in one file

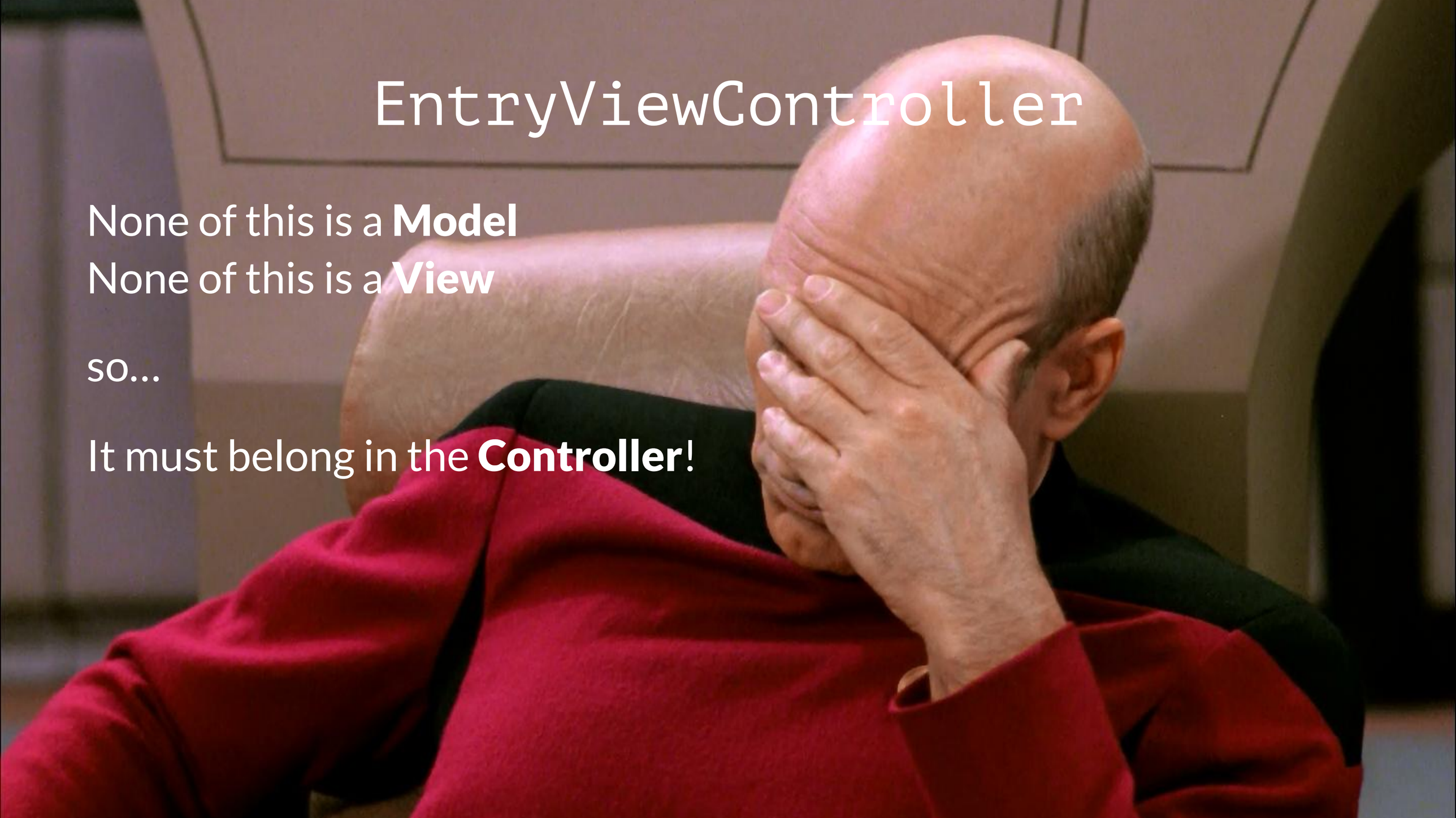
1,970 lines!

`EntryViewController`

None of this is a **Model**
None of this is a **View**

so...

It must belong in the **Controller**!



`EntryViewController`

None of this is a **Model**
None of this is a **View**

so…

It must belong in the **Controller**!

Problem #1:

The Controller

What *is* a controller?

Honestly, I have no idea.

Model

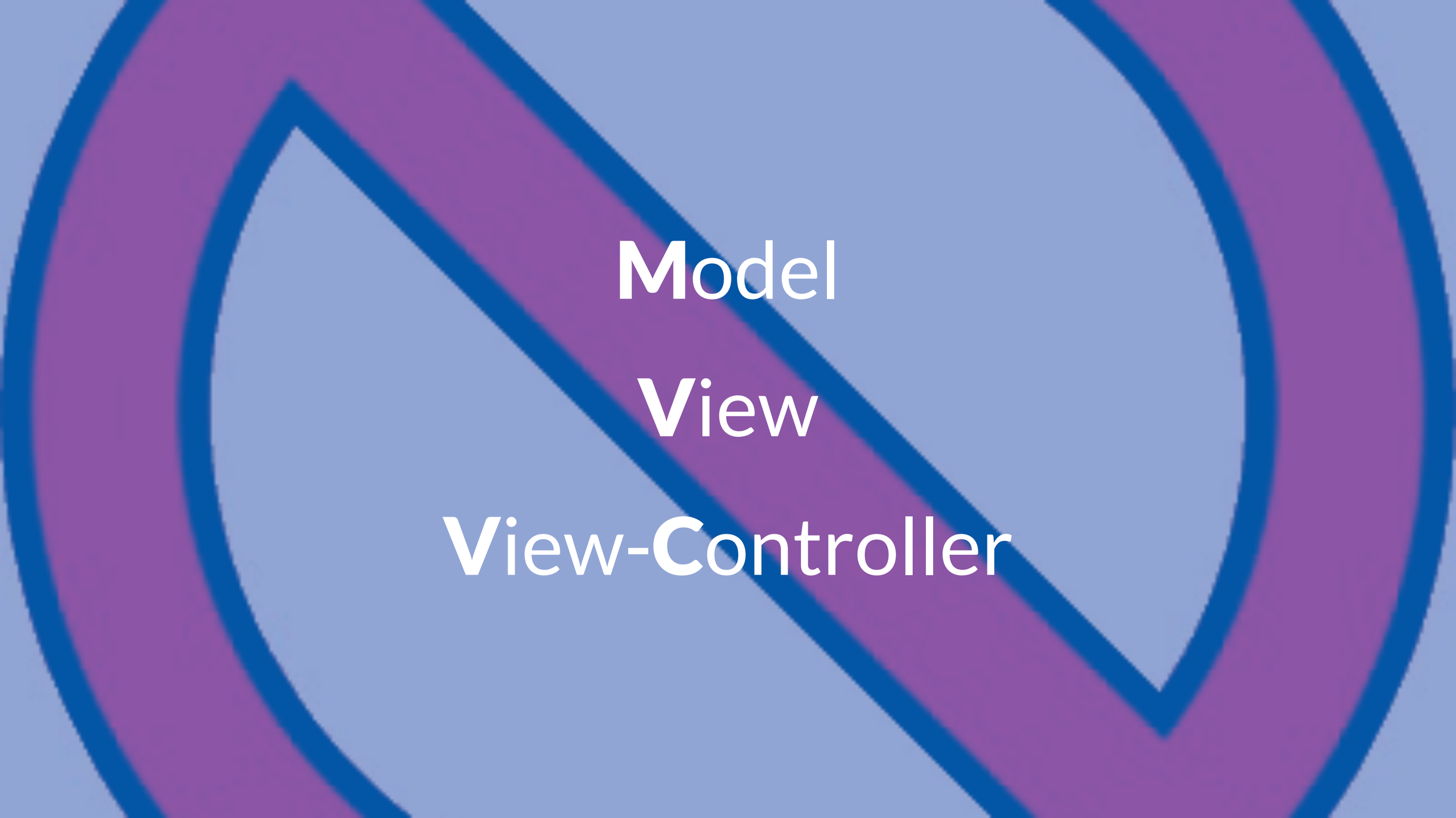
View

Controller

Model

View

View-**C**ontroller

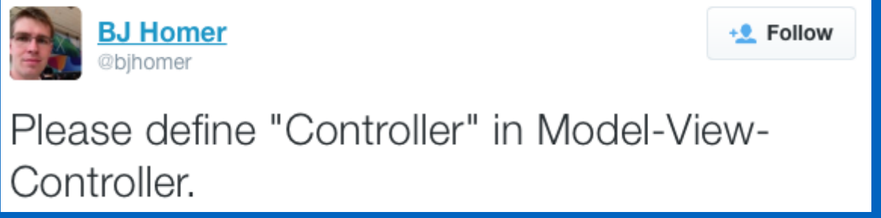

Model
View
View-Controller

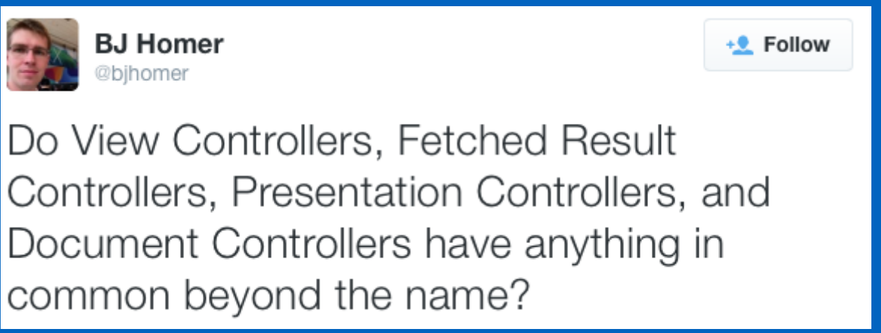
Kinds of Controllers

- View Controller
- Interaction Controller
- Presentation Controller
- Document Controller
- Fetched Results Controller

What do these have in common?

- View Controller
- Interaction Controller
- Presentation Controller
- Document Controller
- Fetched Results Controller


BJ Homer
@bjhomer
Follow
Please define "Controller" in Model-View-Controller.


BJ Homer
@bjhomer
Follow
Do View Controllers, Fetched Result Controllers, Presentation Controllers, and Document Controllers have anything in common beyond the name?

Julio Carrettoni ()
@dev_jac

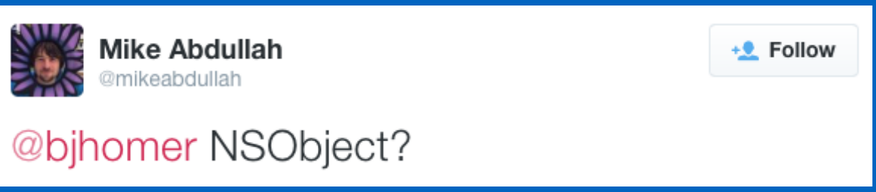
Follow

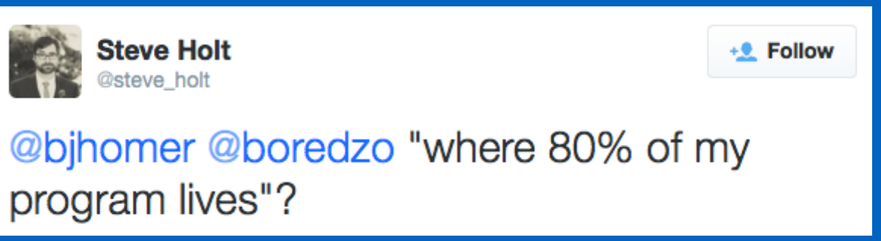
@bjhomer a thing that define lots of behavior of other thing but not all of it as some behavior is still defined in that other thing itself

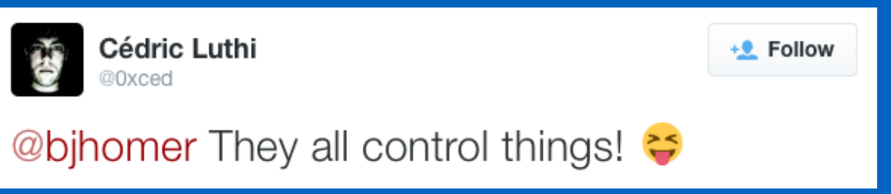
Kyle
@kydare

Follow

@bjhomer sadly, if it's not a model or a view, it's controller!


Mike Abdullah
@mikeabdullah
Follow
@bjhomer NSObject?


Steve Holt
@steve_holt
Follow
@bjhomer @boredzo "where 80% of my program lives"?


Cédric Luthi
@0xced
Follow
@bjhomer They all control things!

Nobody really knows what a "controller" is.

Nobody really knows what a "controller" is.

It's just a made up name.

Maybe we need better words

Names

- Model
- View
- Controller

I asked for some help

Names

Manager
Delegate
DataSource
Driver
Strategy
Processor
Provider
Generator
RepairShop

Names

Index
Cache
Table
ViewModel
Hub
Operation
Factory
Collection
Action

Names

Data
Info
Context
Options
Result
Normalizer
Dispatcher
Executor
Workflow

Names

Transformer
Formatter
Helper
Logger
Notifier
Preparer
Attempter
Utils
Client

Names

Proxy
Analyzer
Populator
Handler
Animator
Importer

MVCMDDDSPPGRICTVHOFCADICORNDE
WTFHLNPAUCPAPHAI

Invent your own kinds of objects!

View Controllers

```
@implementation MyController

#pragma mark - Lifecycle
- (id) initWithNibName:bundle:
- (void) viewDidLoad
- (void) dealloc

#pragma mark - Custom Setters
- (void) setMyProperty:

#pragma mark - IBActions
- (IBAction) clickedAddButton
- (IBAction) clickedNextButton
- (IBAction) clickedPrevButton

#pragma mark - Private Helpers
- (void) prepareTableRows
- (void) presentImagePicker
- (UIImage *) processImage:
```

```
@implementation MyController

#pragma mark - UITableViewDataSource
- (UITableViewCell *) tableView:cellForRowAtIndexPath:
- (NSUInteger) numberOfSectionsInTableView:
- (NSUInteger) tableView:numberOfRowsInSection:

#pragma mark - UITableViewDelegate
- (void) tableView:didSelectRowAtIndexPath:
- (void) tableView:willDisplayCell:forRowAtIndexPath:
- (CGFloat) tableView:heightForRowAtIndexPath:
- (UIView *) tableView:viewForHeaderInSection:
// ...

#pragma mark - NSFetchedResultsControllerDelegate
// ...

#pragma mark - UIImagePickerControllerDelegate
// ...
```

```
@implementation MyController

#pragma mark - UIAlertViewDelegate
// ...

#pragma mark - Notification handling
// ...

#pragma mark - Rotation handling
// ...

// ...

// ...

// ...

// 1,200 lines later
@end
```

Where does this new code go?

"All the stuff it needs is in the View Controller;
it would be weird to have this somewhere else."

Where does this new code go?

"All the stuff it needs is in the View Controller;
it would be weird to have this somewhere else."

```
// 1,300 lines later
@end
```


We need simple tools.

```objc
- (IBAction)clickedTrashButton:(id)sender {
    NSAlert *alert = [NSAlert new];
    NSString *alertTemplate = NSLocalizedString(@"Are you sure you want to delete %ld entries?",
                                                @"Text in a confirmation dialog.");
    alert.messageText = [NSString stringWithFormat:alertTemplate, (long)self.entries.count];
    alert.informativeText = NSLocalizedString(@"This operation cannot be undone.", nil);

    NSButton *deleteButton = [alert addButtonWithTitle:NSLocalizedString(@"Delete", @"Text on a button")];

    NSButton *dontDeleteButton = [alert addButtonWithTitle:NSLocalizedString(@"Don't delete", @"Text on a
        button for not deleting a journal")];

    deleteButton.keyEquivalent = @"";
    dontDeleteButton.keyEquivalent = @"\e";

    NSManagedObjectContext *context = self.filter.context;
    NSWindow *window = self.view.window;
    [alert beginSheetModalForWindow:self.view.window completionHandler:^(NSModalResponse returnCode) {
        if (returnCode == NSAlertFirstButtonReturn) {
            for (DOEntry *entry in self.entries) {
                [context deleteObject:entry];
            }

            NSError *error;
            if (![context save:&error]) {
                NSAlert *alert = [NSAlert new];
                alert.messageText = NSLocalizedString(@"There was an error deleting your journal.", nil);
                alert.informativeText = error.localizedDescription;
                [alert beginSheetModalForWindow:window completionHandler:nil];
                NSLog(@"Error deleting journal: %@", error);
                [context reset];
            }
        }
        else {
            // Nothing. The sheet will be dismissed.
        }

    }];
}
```

```objc
- (IBAction)clickedTrashButton:(id)sender {
    DOXDeleteEntriesConfirmationAlert *alert = [[DOXDeleteEntriesConfirmationAlert alloc]
                                                initWithEntries:self.entries];
    [alert presentConfirmationInWindow:self.view.window];

}
```

```objc
@interface DOXDeleteEntriesConfirmationAlert : NSAlert
@property (readonly) NSArray *entries;

/// @param entries An array of DOEntry objects
- (instancetype)initWithEntries:(NSArray *)entries NS_DESIGNATED_INITIALIZER;

/// Convenience method for presenting the confirmation window and deleting entries if accepted
- (void)presentConfirmationInWindow:(NSWindow *)sheetWindow;

@end
```

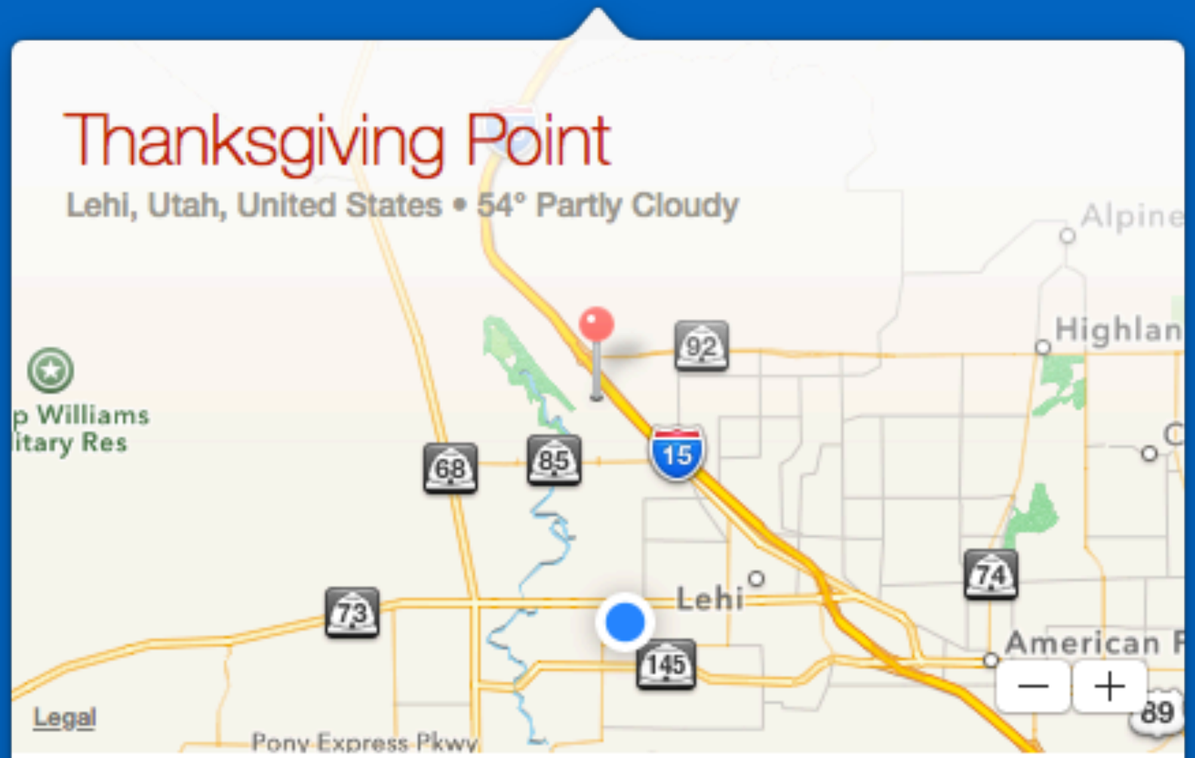
```objc
- (IBAction)clickedTrashButton:(id)sender {
    DOXDeleteEntriesConfirmationAlert *alert = [[DOXDeleteEntriesConfirmationAlert alloc]
                                                initWithEntries:self.entries];
    [alert presentConfirmationInWindow:self.view.window];

}
```

Is this better?

Is this better?

Yes



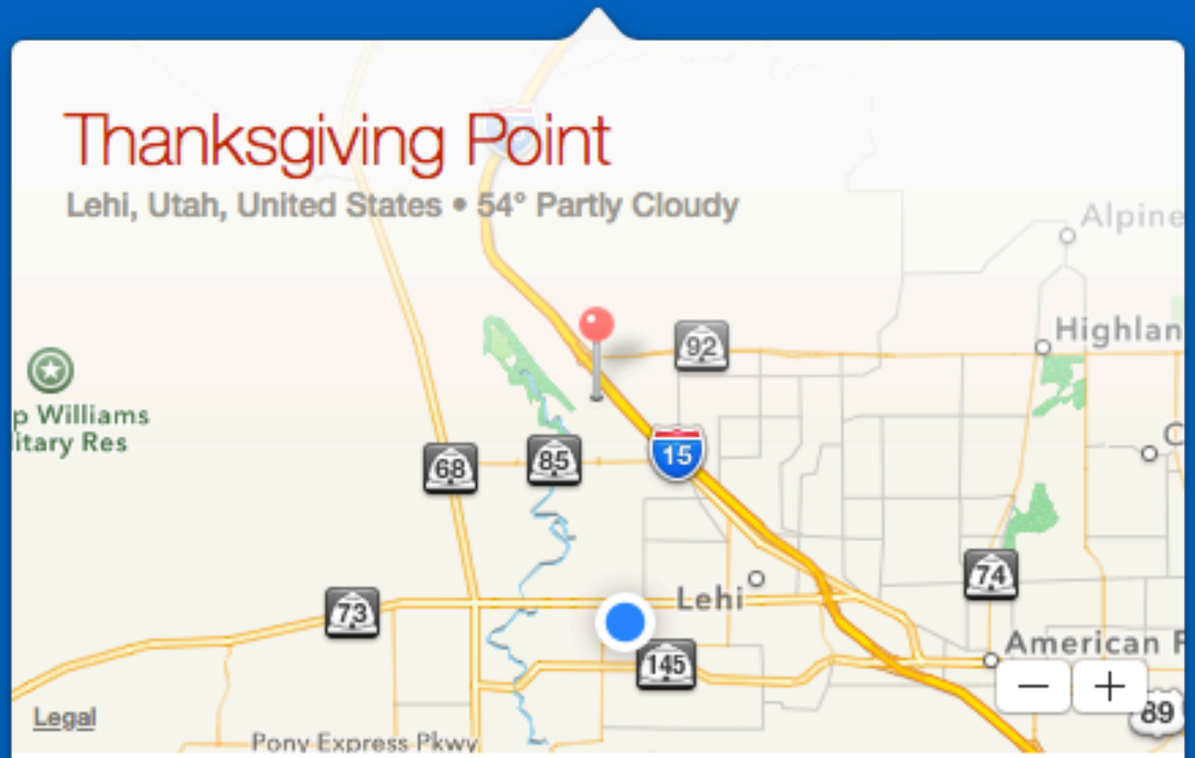
Done

Search

Use Current Location

Select Foursquare Place

Remove Location



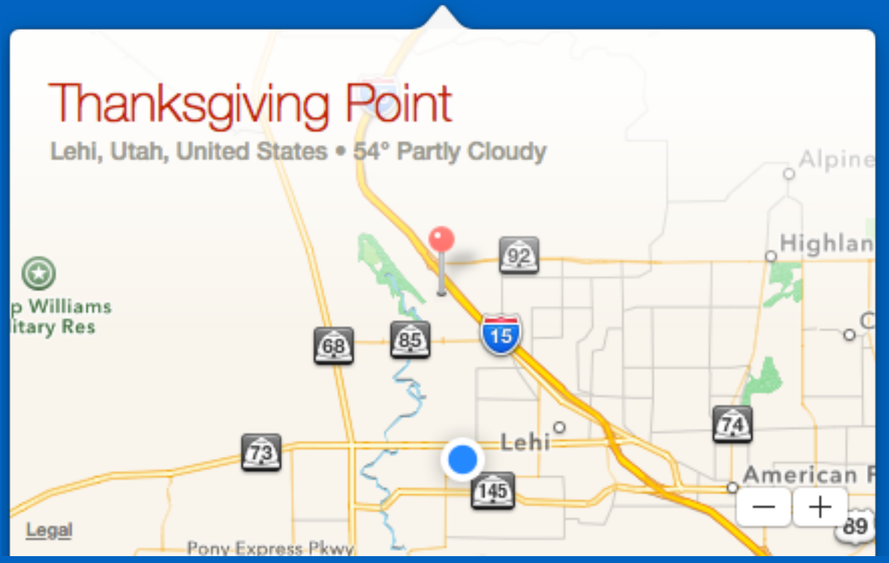
Done

Search

Use Current Location

Select Foursquare Place

Remove Location



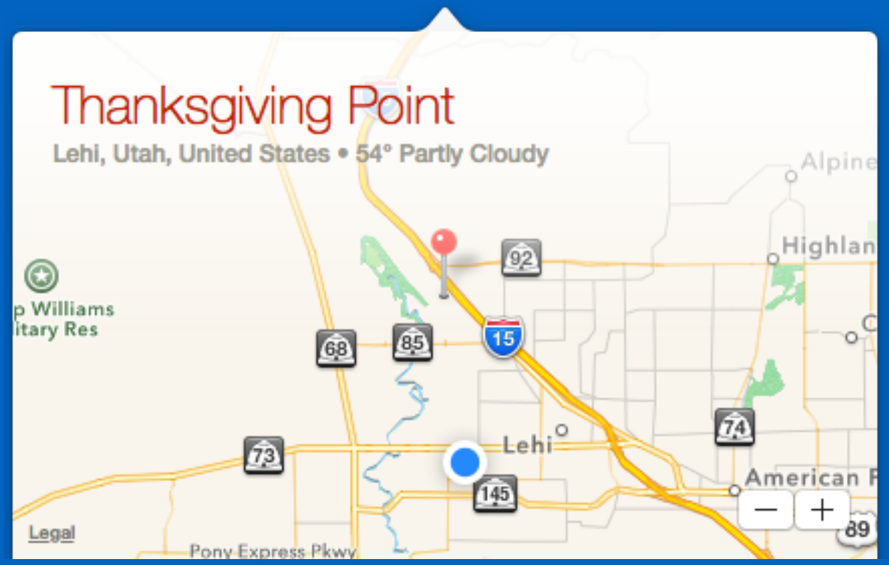
Done

Search

Use Current Location

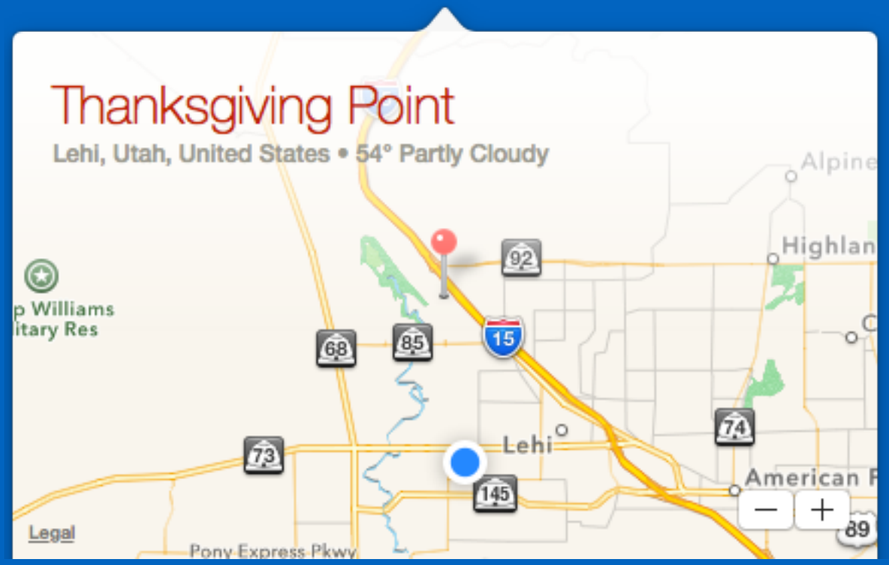
Select Foursquare Place

Remove Location



`EntryLocationMapProvider`

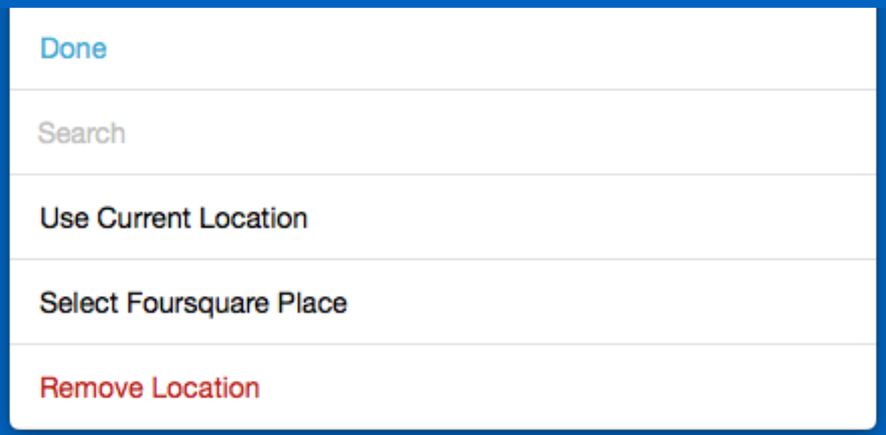
- Adds/removes pins
- Centers, animates, zooms map
- Right-click actions
- Handles pin dragging
- Vends a `Location` object for the current selection



`EntryLocationMapProvider`

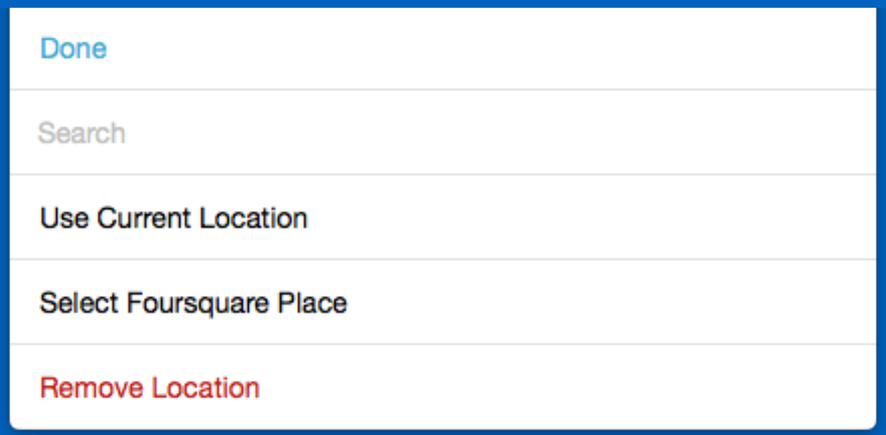
- Adds/removes pins
- Centers, animates, zooms map
- Right-click actions
- Handles pin dragging
- Vends the current `Location` object

331 lines



`EntryLocationTableviewProvider`

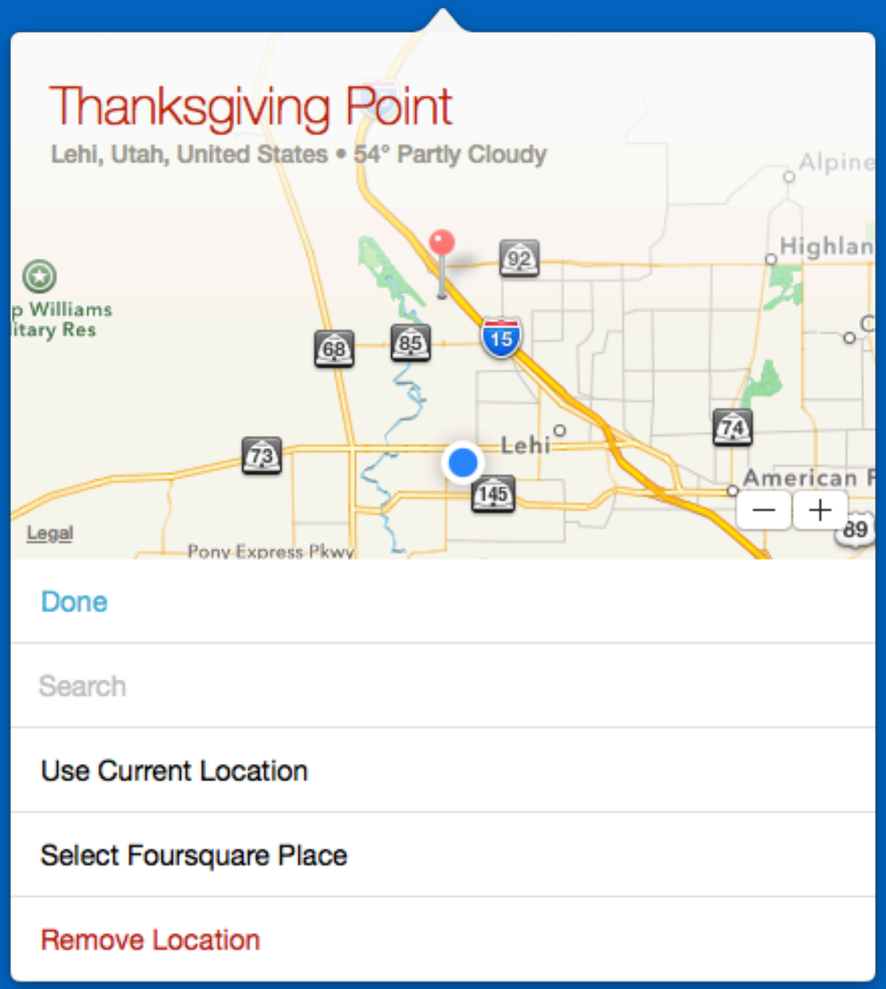
- Table View datasource
- Handles row selections
- Handles search results
- Updates "Current Location" row
- Vends the current `Location` object



`EntryLocationTableviewProvider`

- Table View datasource
- Handles row selections
- Handles search results
- Updates "Current Location" row
- Vends the current `Location` object

600 lines



`EntryLocationMapPopover Controller`

- Coordinator for other providers
- Updates the model

294 lines

Reuse!

View Controllers

What should a view controller (ideally) do?

- Lifecycle
- User input

Pass off everything else

```
- (IBAction)clickedTrashButton:(id)sender {
    DOXDeleteEntriesConfirmationAlert *alert = [[DOXDeleteEntriesConfirmationAlert alloc]
                                                initWithEntries:self.entries];
    [alert presentConfirmationInWindow:self.view.window];

}
```

Aim for **really** small methods

```
- (IBAction)clickedTrashButton:(id)sender {
    DOXDeleteEntriesConfirmationAlert *alert = [[DOXDeleteEntriesConfirmationAlert alloc]
                                                initWithEntries:self.entries];
    [alert presentConfirmationInWindow:self.view.window];

}
```

Aim for **really** small methods

Put all the real work in reusable "tools"

Tip: "Keep-alive" references

```
@property id keepAlive;

- (void)presentTagEditorRelativeToRect:(CGRect)rect ofView:(NSView *)view
{
    [self.popover showRelativeToRect:rect
                              ofView:view
                       preferredEdge:CGRectMinYEdge];

    self.keepAlive = self;
    __weak typeof(self) weakSelf = self;

    self.popover.didCloseBlock = ^(RBLPopover *popover){
        weakSelf.keepAlive = nil;
    };
}
```

Things to put in separate objects

- Things that you repeatedly configure:
  - Menus
  - Image pickers

Things to put in separate objects

- Things that you repeatedly configure:
  - Menus
  - Image pickers
- Calculations

Things to put in separate objects

- Things that you repeatedly configure:
  - Menus
  - Image pickers
- Calculations
- Complex animations

Things to put in separate objects

- Things that you repeatedly configure:
  - Menus
  - Image pickers
- Calculations
- Complex animations
- Anything longer than a couple lines

UICollectionView

Does this really well

UICollectionView

- CollectionView
- CollectionViewCell
- Layout
- DataSource
- Delegate
- Controller

UICollectionView

Make these all separate objects!

Sample in Day One

- CollectionView - **62 lines**
- CollectionViewCell
- Layout
- Layout Helper
- DataSource
- Delegate
- Controller

Sample in Day One

- CollectionView - **62 lines**
- CollectionViewCell - **180 lines**
- Layout
- Layout Helper
- DataSource
- Delegate
- Controller

Sample in Day One

- CollectionView - **62 lines**
- CollectionViewCell - **180 lines**
- Layout - **341 lines**
- Layout Helper
- DataSource
- Delegate
- Controller

Sample in Day One

- CollectionView - **62 lines**
- CollectionViewCell - **180 lines**
- Layout - **341 lines**
- Layout Helper - **207 lines**
- DataSource
- Delegate
- Controller

Sample in Day One

- CollectionView - **62 lines**
- CollectionViewCell - **180 lines**
- Layout - **341 lines**
- Layout Helper - **207 lines**
- DataSource - **81 lines**
- Delegate
- Controller

Sample in Day One

- CollectionView - **62 lines**
- CollectionViewCell - **180 lines**
- Layout - **341 lines**
- Layout Helper - **207 lines**
- DataSource - **81 lines**
- Delegate - **42 lines**
- Controller

Sample in Day One

- CollectionView - **62 lines**
- CollectionViewCell - **180 lines**
- Layout - **341 lines**
- Layout Helper - **207 lines**
- DataSource - **81 lines**
- Delegate - **42 lines**
- Controller - **141 lines**

Sample in Day One

Total: **1054 lines**

Sample in Day One

Total: **1054 lines**

All focused, none overwhelming

"Build friendly tools, not scary controllers"

— BJ Homer

Thank you for coming!