

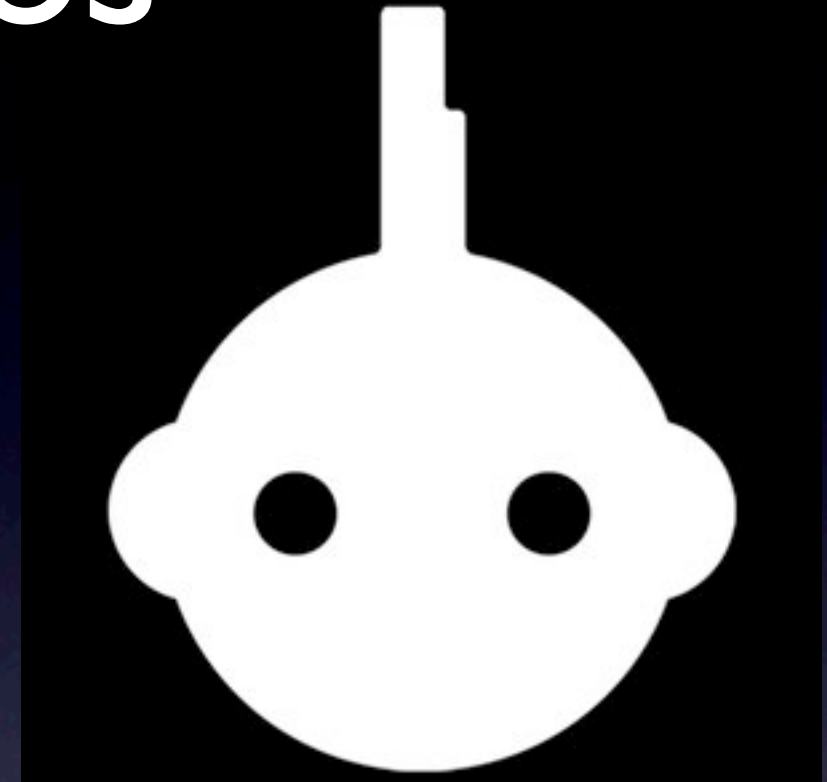
iOS Development @ Cohab

Brent Nycum
brentnycum@gmail.com

SMUG Meeting

- Shreveport Mac Users Group
- Tomorrow night, August 24th
 - 6:30 - 8:00
- Cohabitat
- Pizza will be served
- Register on Facebook

Moonbot Studios



- Looking for iOS Developer
- <http://moonbotstudios.com/jobs.html>
- C++ and Objective C experience and have shipped one or more software products. Ideal candidate has Python and Unity experience.

Baby News

- Our daughter will now be here by September 20th
 - Either naturally or induced
- If you have an updated email address make sure I have it
 - Also check out Cohab's Facebook for any news
- We will skip a week and continue on

Schedule

- Tonight we will discuss Table Views and the Navigation Controller

Schedule

- Tab based applications
- Split View based applications (iPad)
 - Popovers

Schedule

- Gesture Recognizers
- Scroll Views
- Web Views
- Image Views

Schedule

- Media
 - Images, Audio, Video
- Core Location
- Map Kit

Schedule

- Alerts
- Action Sheets
- Submitting to the App Store
 - Handling certificates

- Table Views
- Navigation Controller

But first...

Lets build an app

- Temperature Convertor
 - To °F
 - To °C

Temp Converter App

- Create a New Project in Xcode
- Use a View Based Application as the template
 - iPhone based for simplicity

Temp Converter App

```
@interface TempConvertViewController :  
UIViewController {  
  
}
```

```
@end
```

Temp Converter App

- What we will need?
 - Input field
 - 2 Buttons that will convert input
 - To °F
 - To °C
 - Output Label with converted value

Temp Converter App

- Add our IBOutlets to our View Controller header file
 - IBOutlet UITextField *inputField;
 - IBOutlet UILabel *outputField;

Temp Converter App

```
@interface TempConvertViewController :  
UIViewController {  
    IBOutlet UITextField *inputField;  
    IBOutlet UILabel *outputField;  
}  
  
@end
```

Temp Converter App

- Why not create outlets for our Buttons?
 - Our buttons will call separate actions
 - No need for additional references

Temp Converter App

- We also need 2 actions
- Convert actions when either button is pressed
 - `-(IBAction)convertToF:(UIButton *)sender;`
 - `-(IBAction)convertToC:(UIButton *)sender;`

Temp Converter App

```
@interface TempConvertViewController :  
UIViewController {  
    IBOutlet UITextField *inputField;  
    IBOutlet UILabel *outputField;  
}  
  
-(IBAction)convertToF:(UIButton *)sender;  
-(IBAction)convertToC:(UIButton *)sender;  
  
@end
```

Temp Converter App

- Add these to our Interface Builder file (.xib)
 - 1x UITextField
 - 2x UIButton
 - 1x UILabel

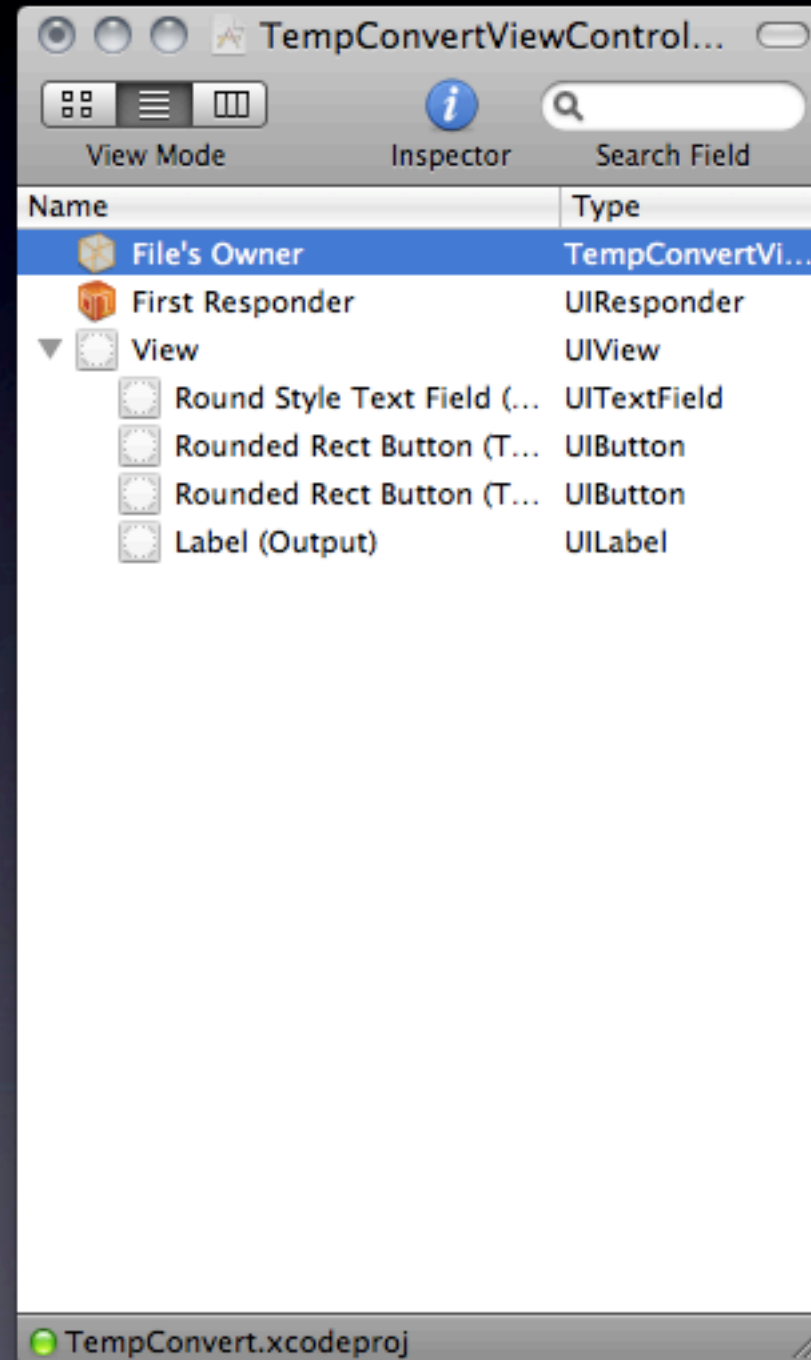
Temp Convertor App

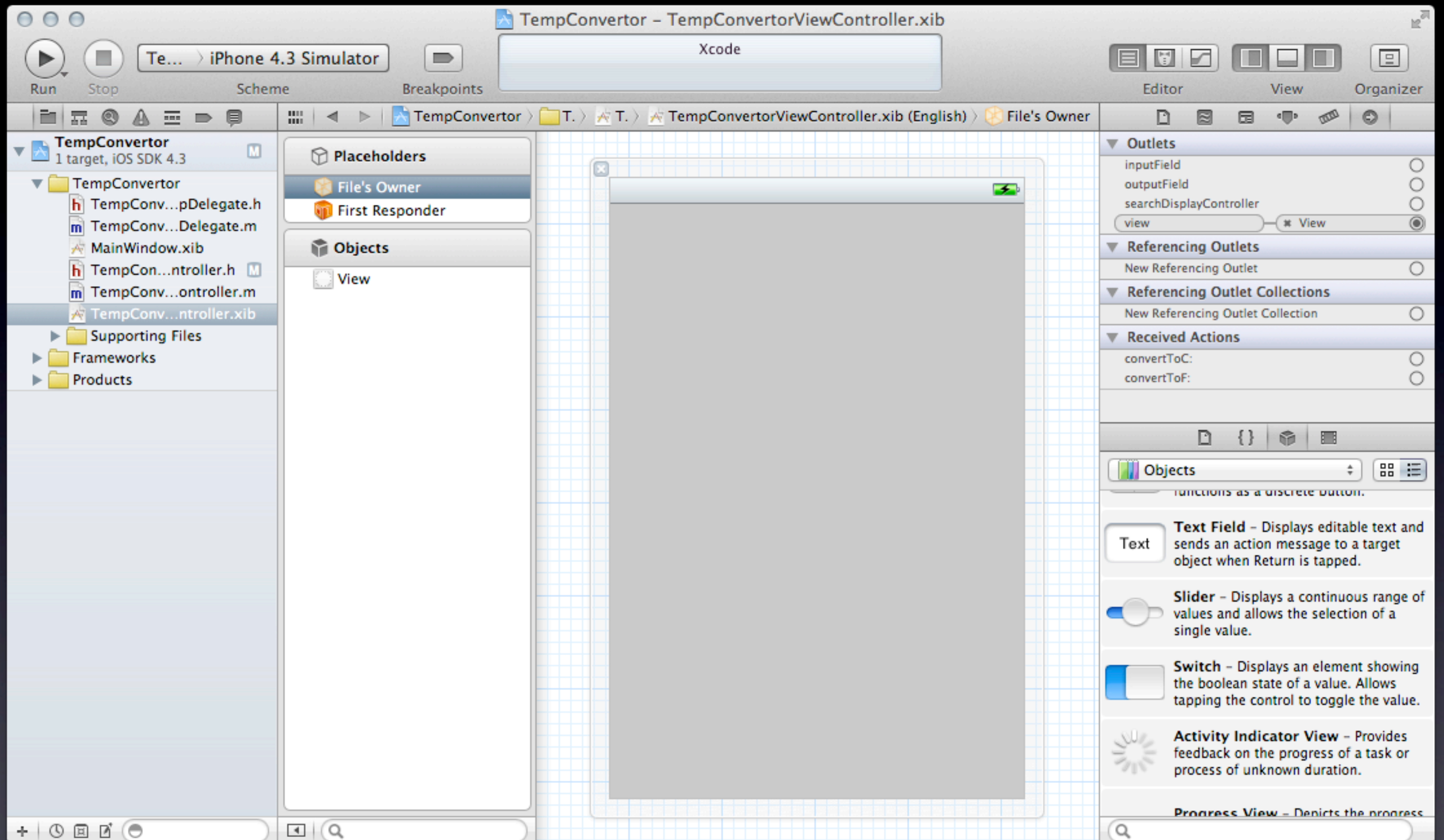
The image shows a graphical user interface for a temperature conversion application. The window has a title bar with the text 'View' and standard macOS window controls (red, yellow, and green buttons). Below the title bar, there is a text input field labeled 'Input'. Underneath the input field are two buttons: 'To °F' and 'To °C'. Below these buttons is a label 'Output'. The entire interface is set against a light gray background.

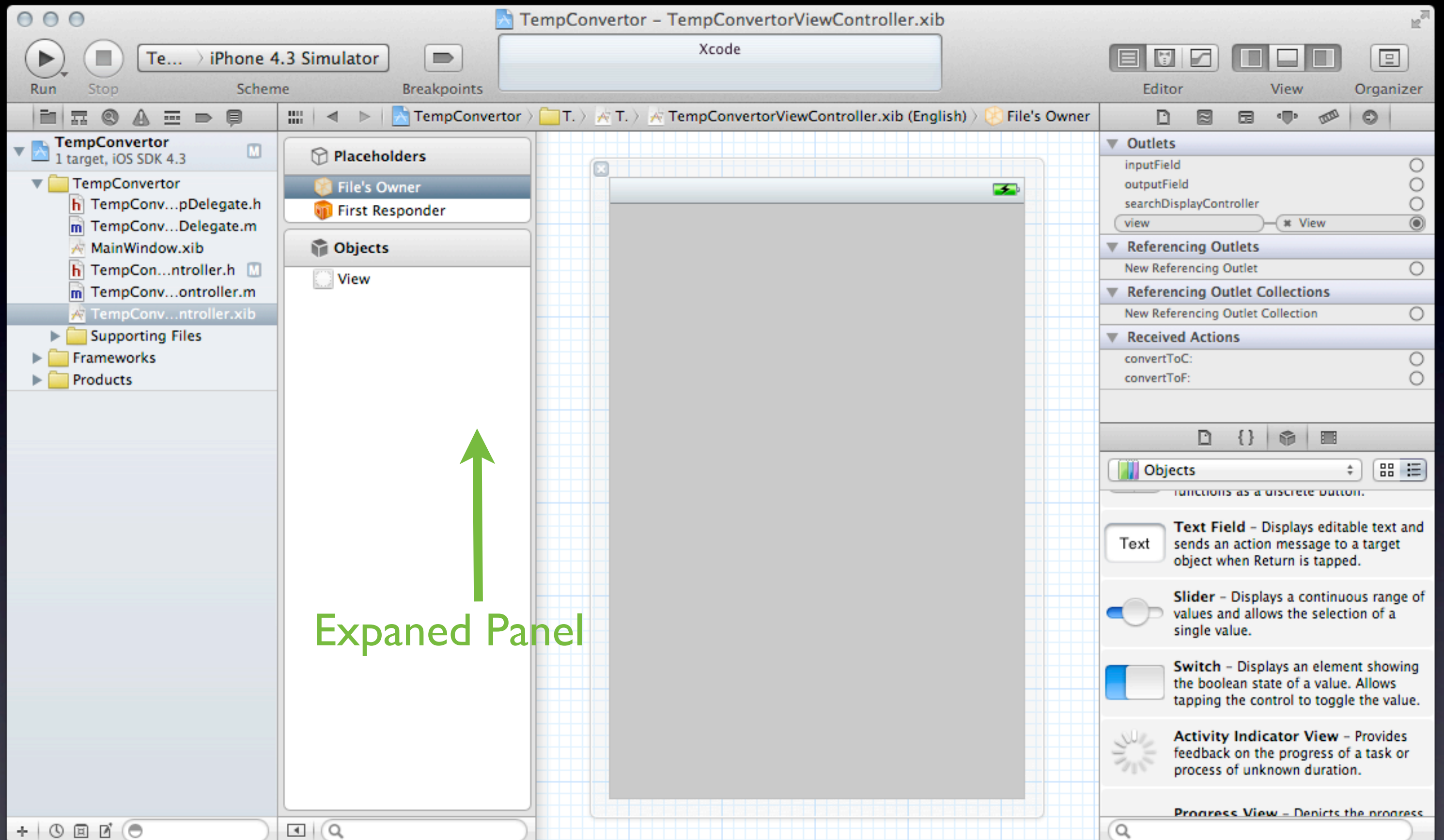
Temp Converter App

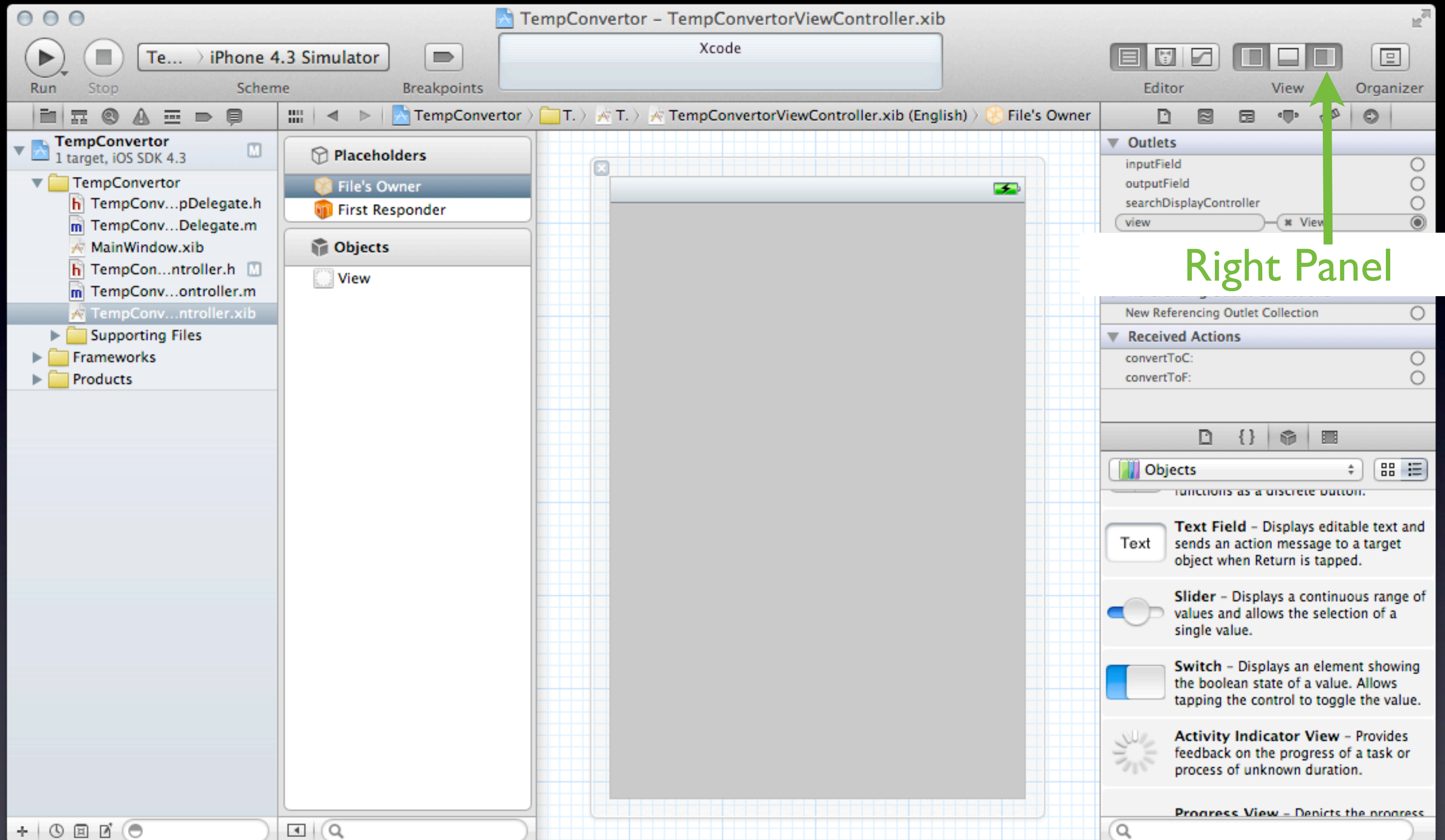
- Lets wire up our Outlets and Actions

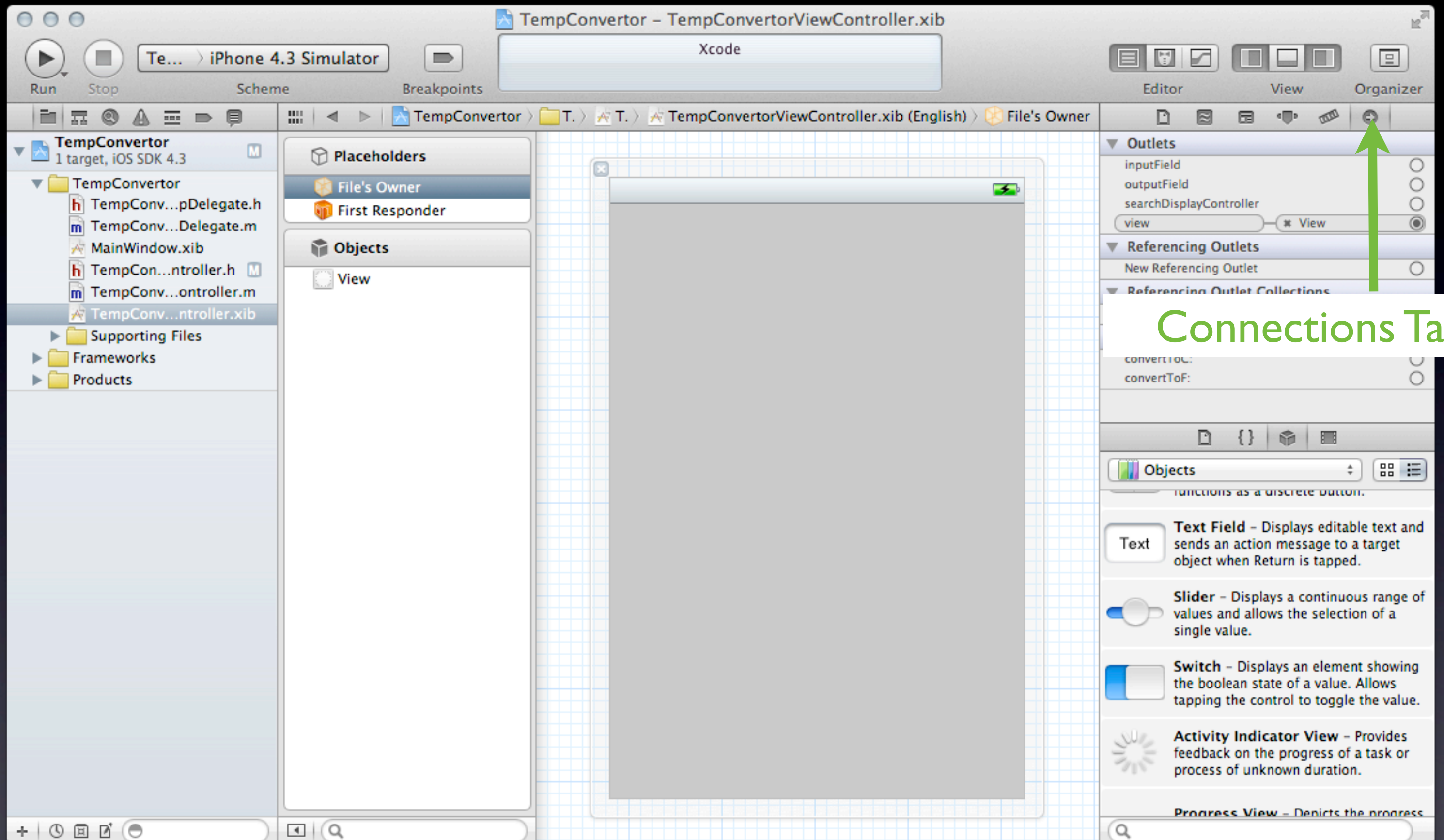
Temp Convertor App

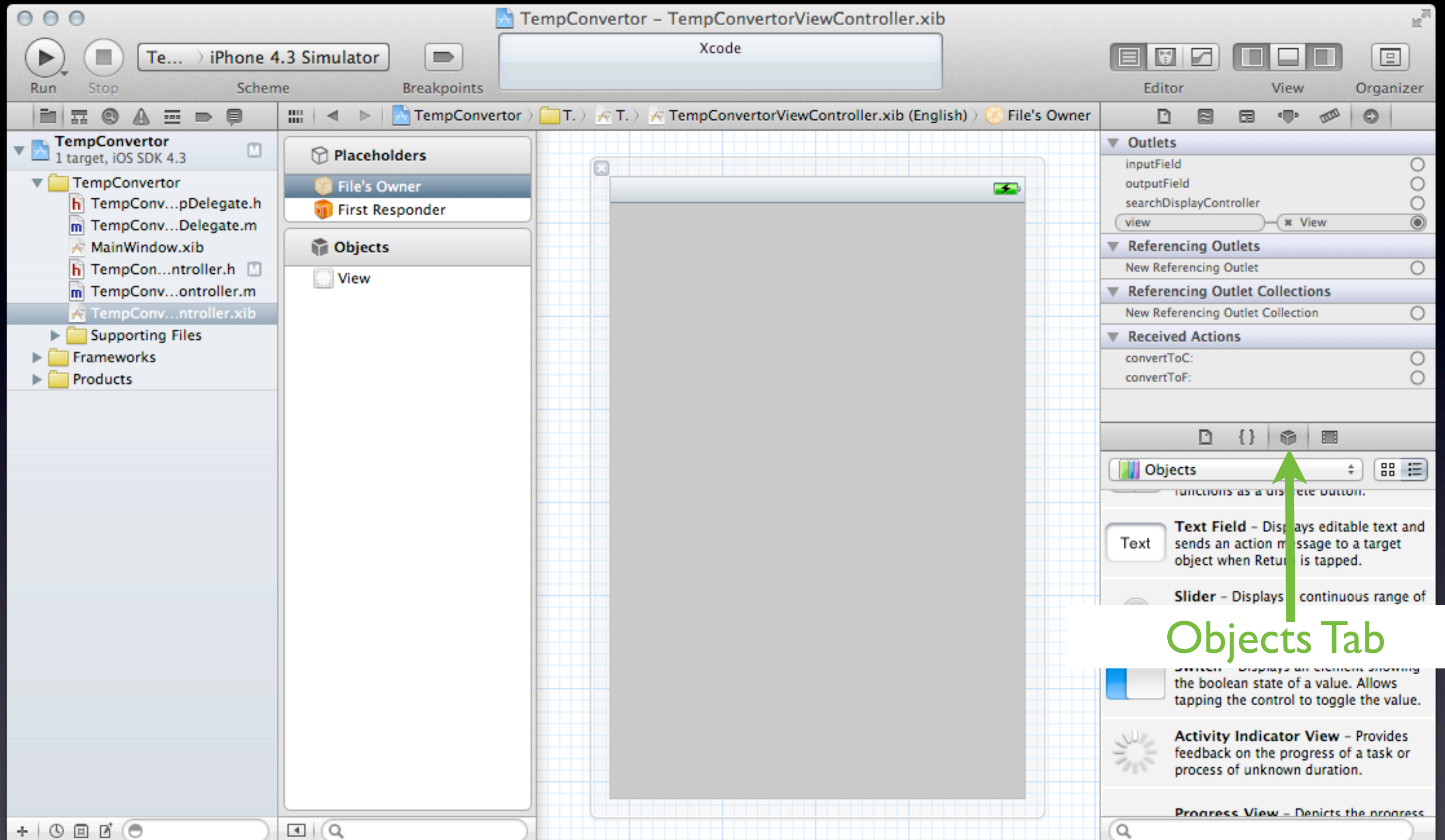


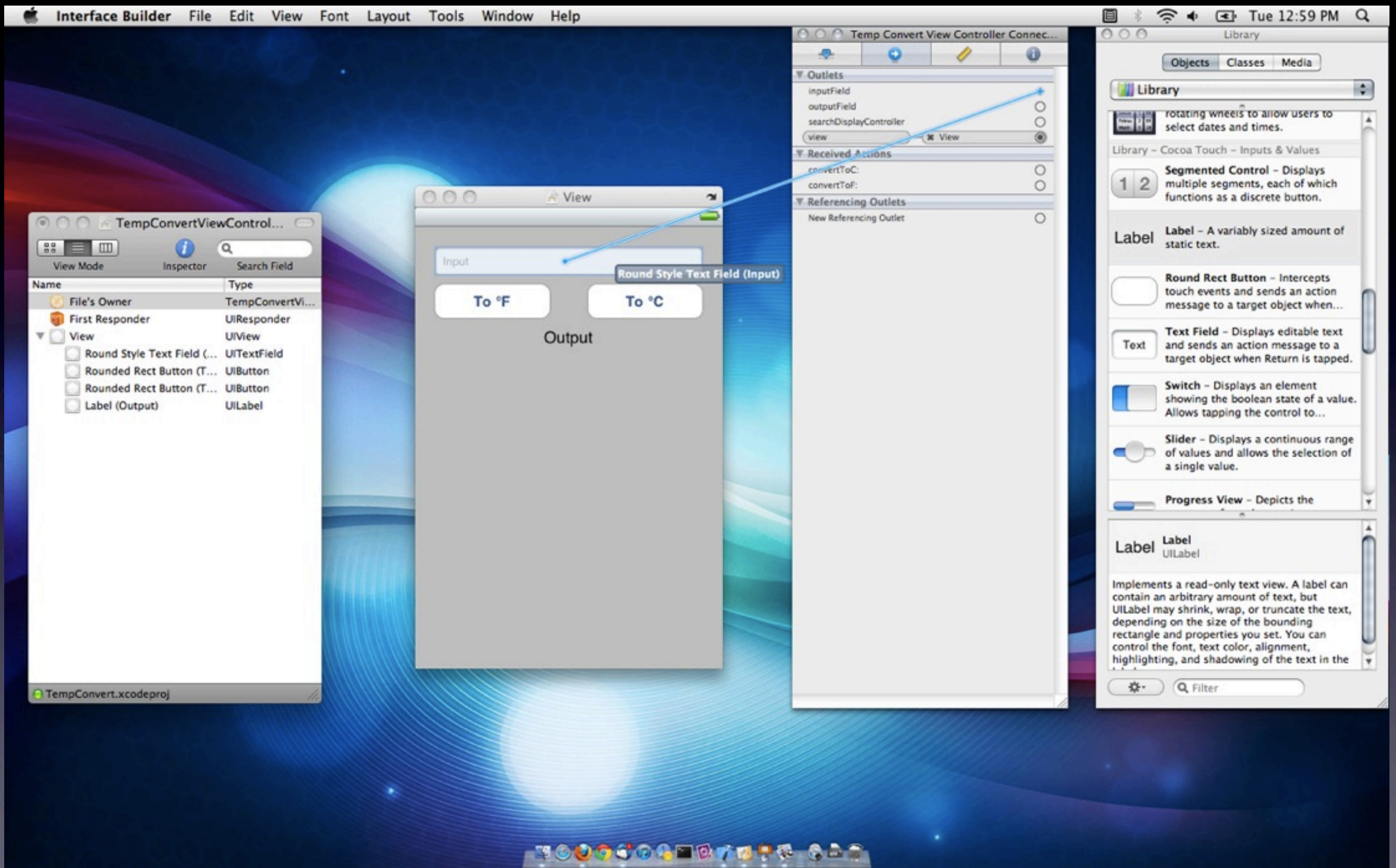












TempConvertViewControl...

View Mode Inspector Search Field

Name	Type
File's Owner	TempConvertVi...
First Responder	UIResponder
View	UIView
Round Style Text Field (...)	UITextField
Rounded Rect Button (T...	UIButton
Rounded Rect Button (T...	UIButton
Label (Output)	UILabel

TempConvert.xcodeproj

View

Input

To °F To °C

Output

Temp Convert View Controller Connec...

Outlets

inputField Round Style Text Fi... outputField searchDisplayController view View

Received Actions

convertToC: convertToF:

Referencing Outlets

New Referencing Outlet

Library

Objects Classes Media

Library

rotating wheels to allow users to select dates and times.

Library - Cocoa Touch - Inputs & Values

1 2 Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Label Label - A variably sized amount of static text.

Round Rect Button - Intercepts touch events and sends an action message to a target object when...

Text Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

Switch - Displays an element showing the boolean state of a value. Allows tapping the control to...

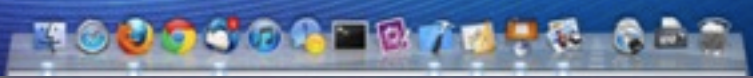
Slider - Displays a continuous range of values and allows the selection of a single value.

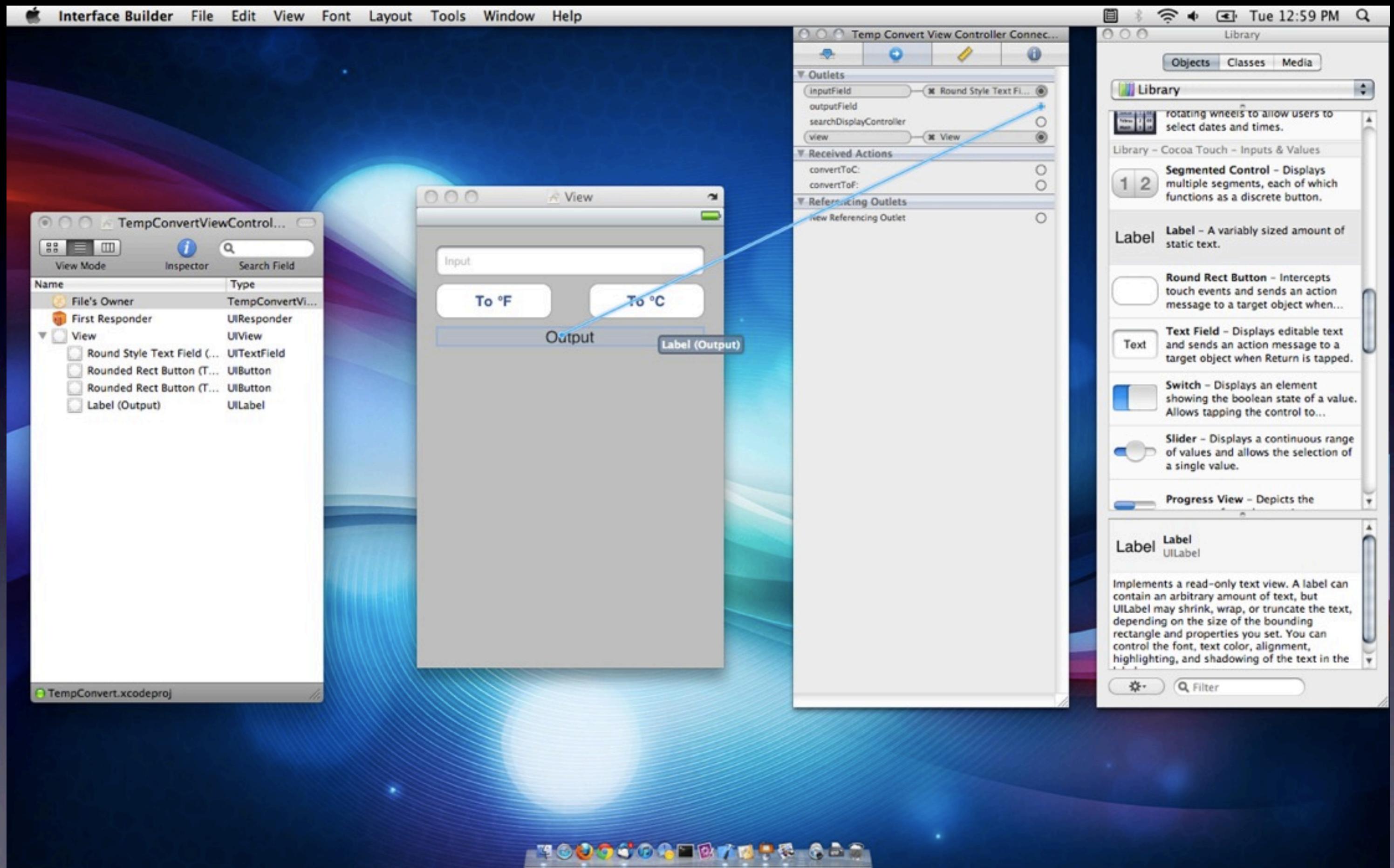
Progress View - Depicts the

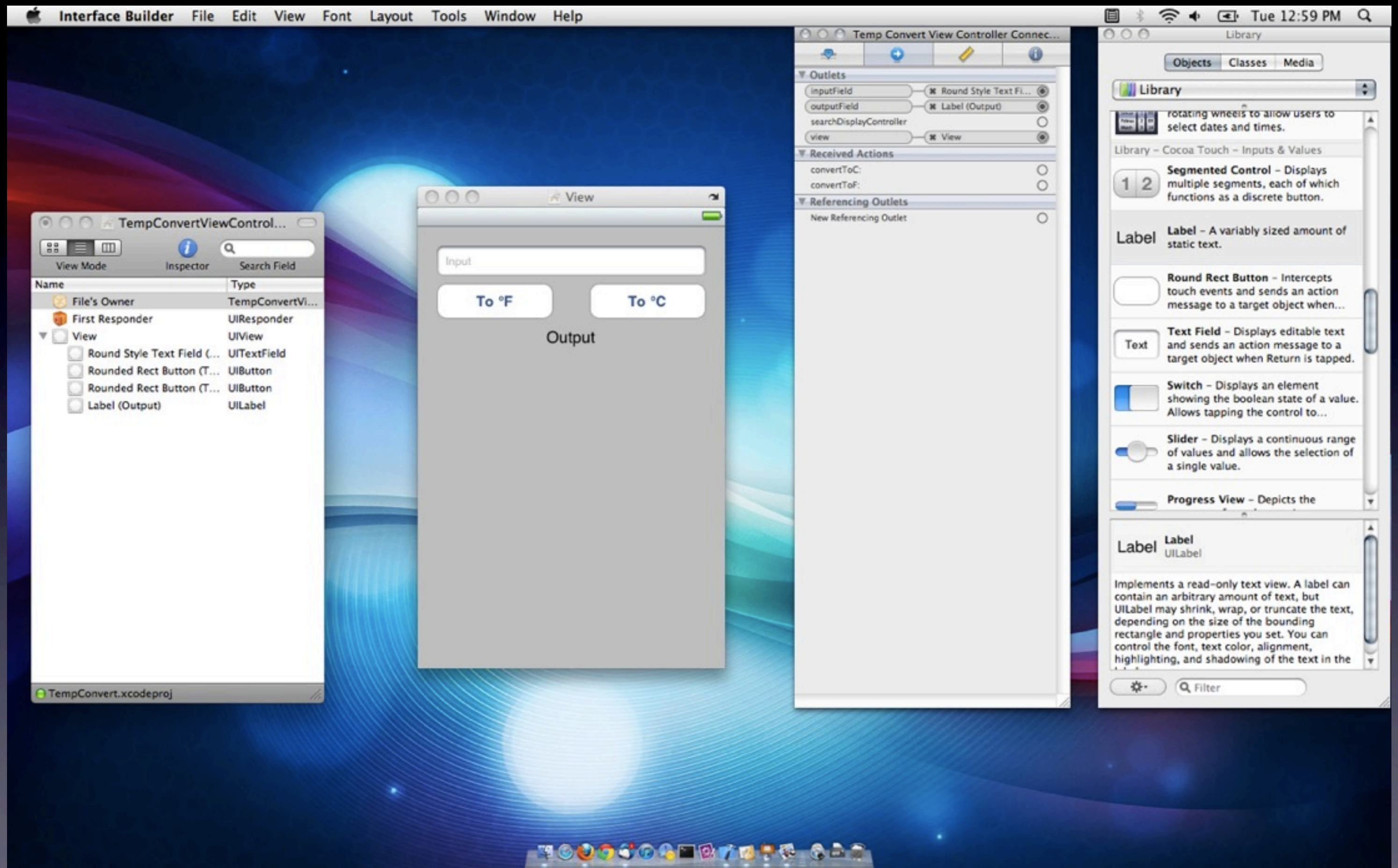
Label UILabel

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the

Filter







TempConvertViewControl...

View Mode Inspector Search Field

Name	Type
File's Owner	TempConvertVi...
First Responder	UIResponder
View	UIView
Round Style Text Field (...)	UITextField
Rounded Rect Button (T...	UIButton
Rounded Rect Button (T...	UIButton
Label (Output)	UILabel

TempConvert.xcodeproj

View

Input

To °F To °C

Rounded Rect Button (To °F)

Output

Temp Convert View Controller Connec...

Outlets

- inputField Round Style Text Fi...
- outputField Label (Output)
- searchDisplayController
- view View

Received Actions

- convertToC:
- convertToF:

Referencing Outlets

- New Referencing Outlet

Library

Objects Classes Media

Library

rotating wheels to allow users to select dates and times.

Library - Cocoa Touch - Inputs & Values

1 2 Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Label Label - A variably sized amount of static text.

Round Rect Button - Intercepts touch events and sends an action message to a target object when...

Text Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

Switch - Displays an element showing the boolean state of a value. Allows tapping the control to...

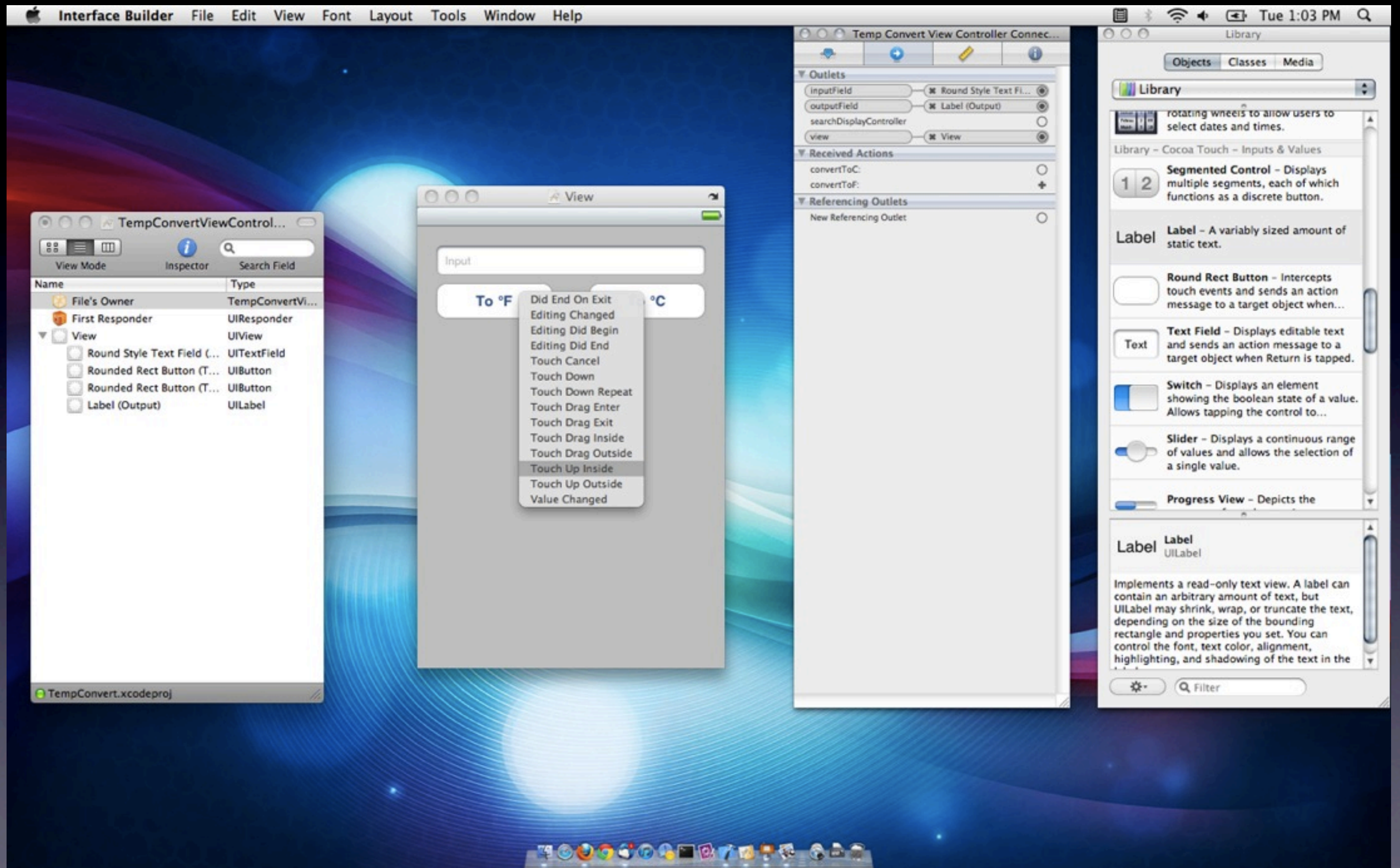
Slider - Displays a continuous range of values and allows the selection of a single value.

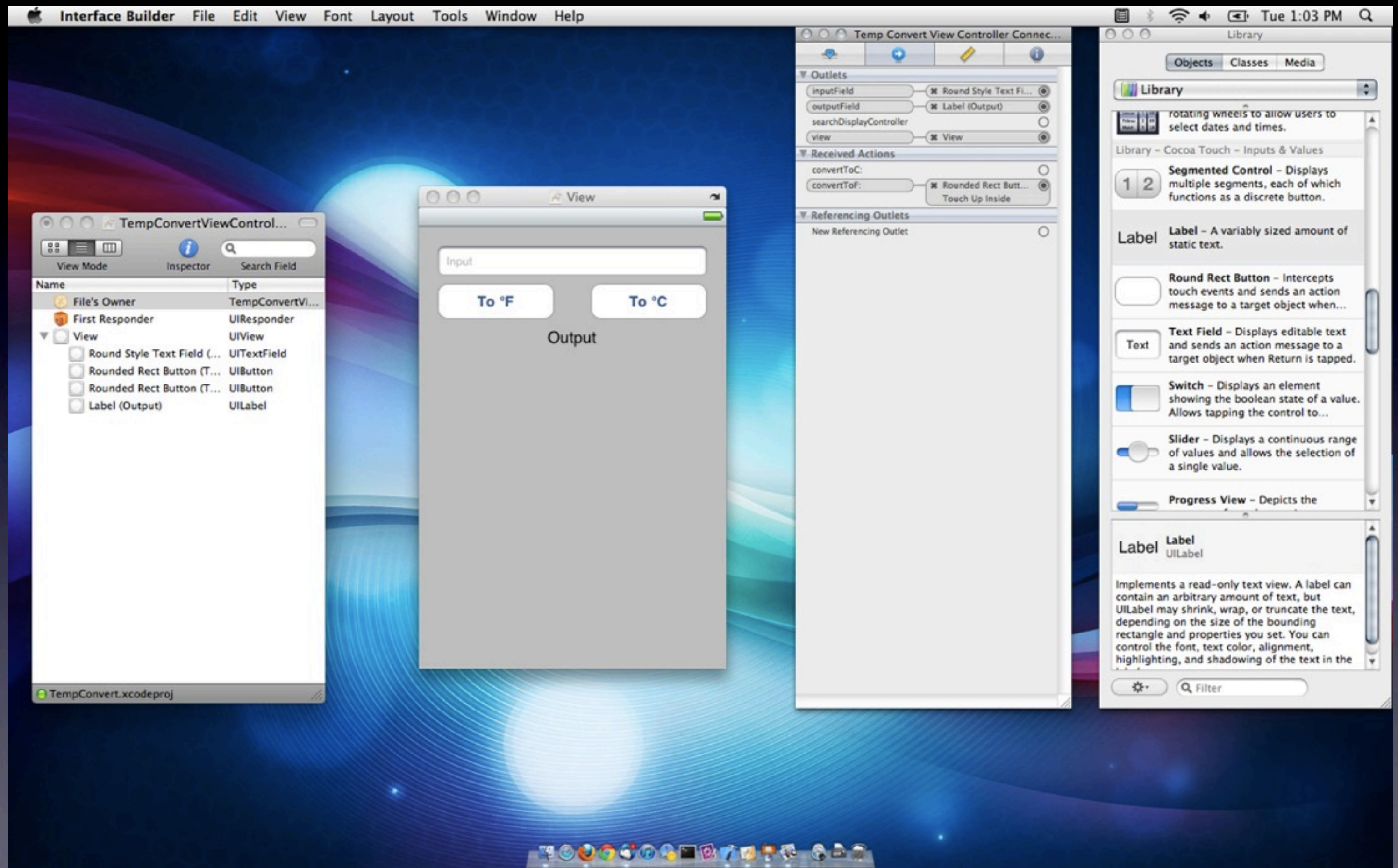
Progress View - Depicts the

Label UILabel

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the

Filter





TempConvertViewControl...

View Mode Inspector Search Field

Name	Type
File's Owner	TempConvertVi...
First Responder	UIResponder
View	UIView
Round Style Text Field (...)	UITextField
Rounded Rect Button (T...	UIButton
Rounded Rect Button (T...	UIButton
Label (Output)	UILabel

TempConvert.xcodeproj

View

Input

To °F To °C

Output

Rounded Rect Button (To °C)

Temp Convert View Controller Connec...

Outlets

- inputField → Round Style Text Fi...
- outputField → Label (Output)
- searchDisplayController
- view → View

Received Actions

- convertToC: → Rounded Rect butt... Touch Up Inside
- convertToF: → Rounded Rect butt... Touch Up Inside

Referencing Outlets

- New Referencing Outlet

Library

Objects Classes Media

Library

rotating wheels to allow users to select dates and times.

Library - Cocoa Touch - Inputs & Values

1 2

Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Label - A variably sized amount of static text.

Round Rect Button - Intercepts touch events and sends an action message to a target object when...

Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

Switch - Displays an element showing the boolean state of a value. Allows tapping the control to...

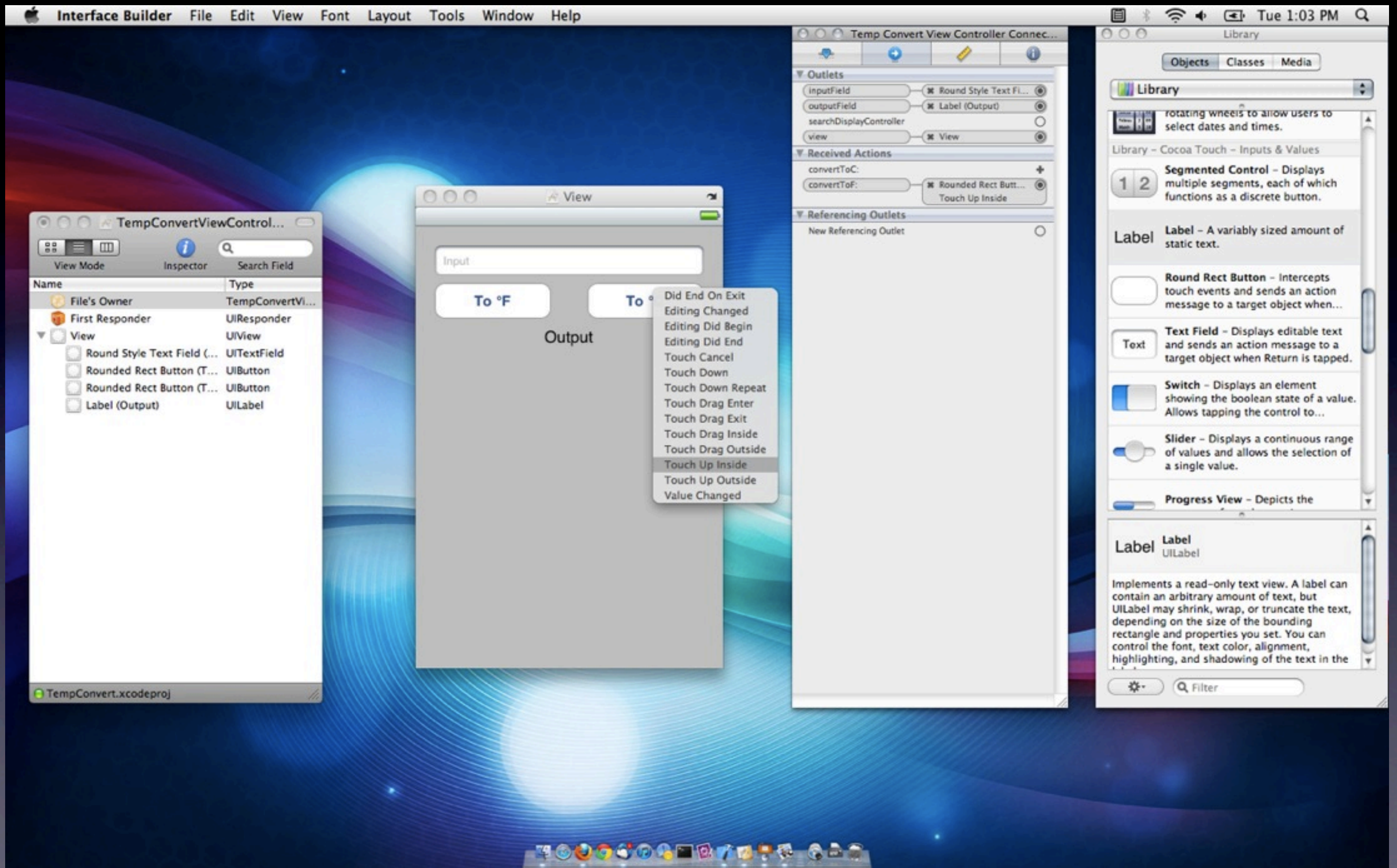
Slider - Displays a continuous range of values and allows the selection of a single value.

Progress View - Depicts the

Label UILabel

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the

Filter



TempConvertViewControl...

View Mode Inspector Search Field

Name	Type
File's Owner	TempConvertVi...
First Responder	UIResponder
View	UIView
Round Style Text Field (...)	UITextField
Rounded Rect Button (T...	UIButton
Rounded Rect Button (T...	UIButton
Label (Output)	UILabel

TempConvert.xcodeproj



Temp Convert View Controller Connec...

Outlets

- inputField - Round Style Text Fi...
- outputField - Label (Output)
- searchDisplayController
- view - View

Received Actions

- convertToC: - Rounded Rect Butt... Touch Up Inside
- convertToF: - Rounded Rect Butt... Touch Up Inside

Referencing Outlets

- New Referencing Outlet

Library

Objects Classes Media

Library

rotating wheels to allow users to select dates and times.

Library - Cocoa Touch - Inputs & Values

1 2 Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Label Label - A variably sized amount of static text.

Round Rect Button - Intercepts touch events and sends an action message to a target object when...

Text Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

Switch - Displays an element showing the boolean state of a value. Allows tapping the control to...

Slider - Displays a continuous range of values and allows the selection of a single value.

Progress View - Depicts the

Label UILabel

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the

Filter

Temp Convertor App

- Now we need to fill in our convert code
 - $[^{\circ}\text{C}] = ([^{\circ}\text{F}] - 32) \times 5/9$
 - $[^{\circ}\text{F}] = [^{\circ}\text{C}] \times 9/5 + 32$

Print Formatters

- Haven't talked about this before
- A format string is given to a method that provides a description of the output
- Have placeholder values that are designated by a % escape character

Print Formatters

- %@ - Object
- %i - signed int
- %d - double
- %f - float
- %u - unsigned int

Print Formatters

- `@” %i + %i = %i”, 2, 2, (2+2)`
- `2 + 2 = 4`

Print Formatters

- We will be using the `stringWithFormat` method from `NSString`
- `[NSString stringWithFormat:@"%d °C",
temperature];`
- 12 °C

Temp Converter App

```
-(IBAction)convertToF:(UIButton *)sender {  
    int toConvert = [inputField.text intValue];  
  
}
```

Temp Converter App

```
-(IBAction)convertToF:(UIButton *)sender {  
    int toConvert = [inputField.text intValue];  
  
    float convertedValue = toConvert * 9.0 / 5.0 + 32;  
  
}
```

Temp Converter App

```
-(IBAction)convertToF:(UIButton *)sender {  
    int toConvert = [inputField.text intValue];  
  
    float convertedValue = toConvert * 9.0 / 5.0 + 32;  
  
    outputField.text = [NSString  
stringWithFormat:@"%f", convertedValue];  
}
```

Table Views

Table Views

- Don't worry about memorizing method names, Xcode will provide all this for us

Table Views

- Vertical list of Cells
- Single column, designed for a device with a small screen
 - No multiple columns without custom programming
 - So nothing like a Grid

Table Views

- Made up of zero or more Sections
- Each section containing it's own rows, Cells

Table Views

- Sections have an index within the Table View
- Rows have an index within their section
- Remember: Everything is 0 Based

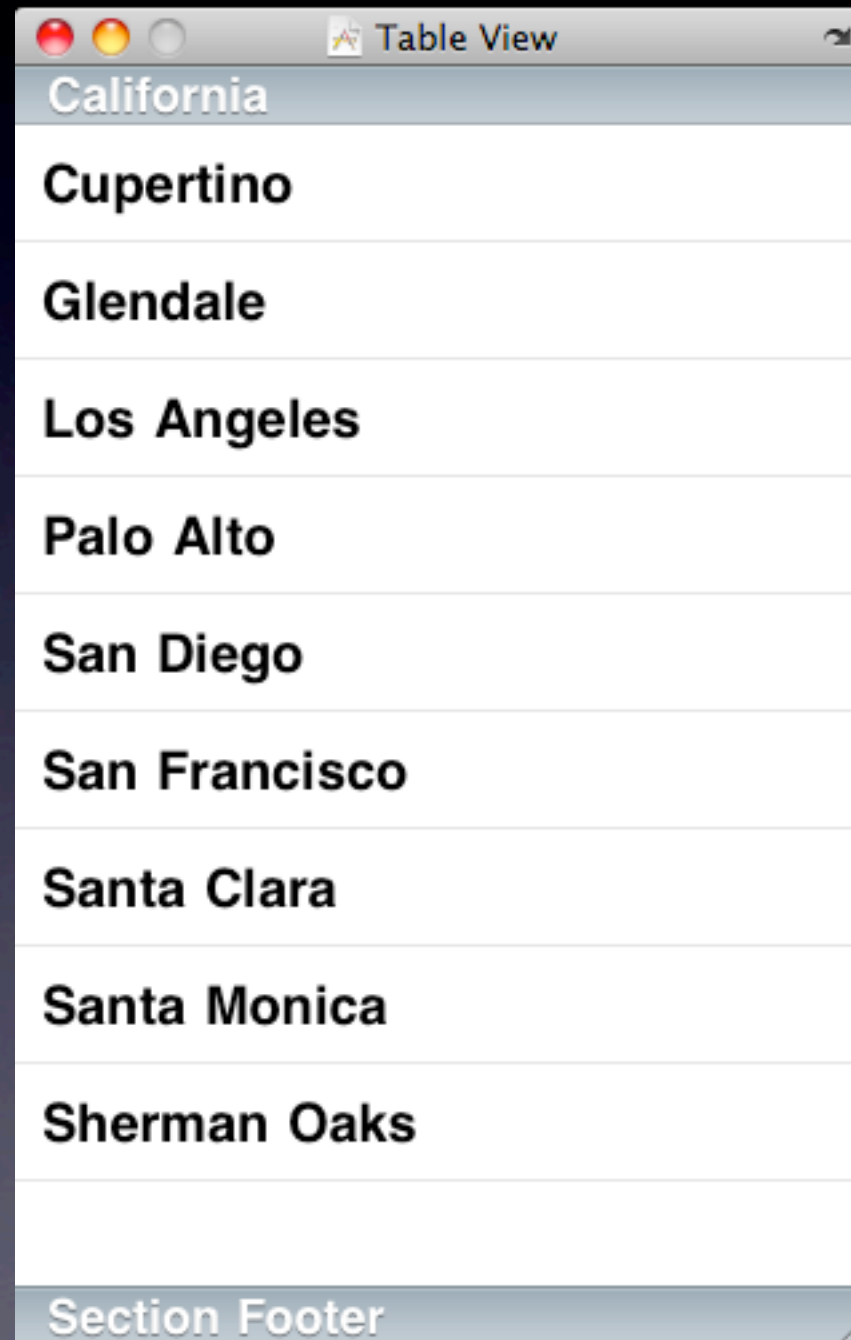
Table Views

- Sections can optionally have a header or a footer

Table Views

- Table Views have 2 different styles
 - Plain
 - Grouped

TableView Plain



California
Cupertino
Glendale
Los Angeles
Palo Alto
San Diego
San Francisco
Santa Clara
Santa Monica
Sherman Oaks
Section Footer

Table View Grouped

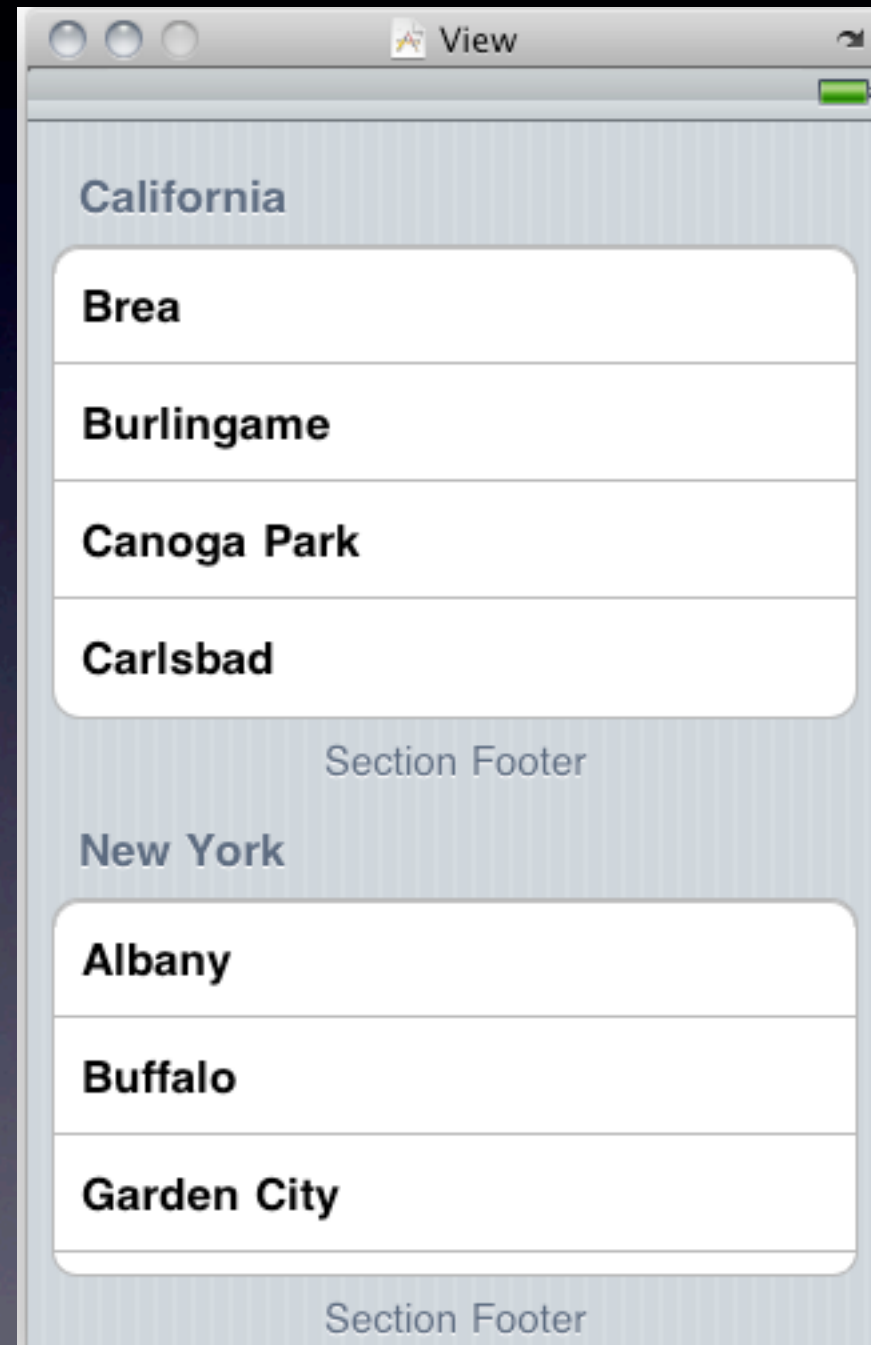


Table View

Section 0

Section 1

California	
Brea	
Burlingame	
Canoga Park	
Carlsbad	
Section Footer	
New York	
Albany	
Buffalo	
Garden City	
Section Footer	

Row 0

Row 1

Row 2

Row 3

Row 0

Row 1

Row 2

Table View

Section 0 Header

Section I Header



The screenshot shows a mobile application window titled "View" with a standard iOS-style title bar (three buttons on the left, a close button on the right, and a battery indicator in the top right corner). The content area has a light blue background with vertical stripes. It displays two sections of data. The first section is titled "California" in a medium blue font. Below the title is a white rounded rectangle containing four rows of city names: "Brea", "Burlingame", "Canoga Park", and "Carlsbad". Below this rounded rectangle is the text "Section Footer" in a light blue font. The second section is titled "New York" in a medium blue font. Below the title is a white rounded rectangle containing three rows of city names: "Albany", "Buffalo", and "Garden City". Below this rounded rectangle is the text "Section Footer" in a light blue font.

California
Brea
Burlingame
Canoga Park
Carlsbad
Section Footer
New York
Albany
Buffalo
Garden City
Section Footer

Section 0 Footer

Section I Footer

Table Views

- How do we tell our Table View how many Sections and Rows?
 - Our Table View will ask us
 - Calls our Table View Data Source
 - Remember Delegates from last week

Table Views

- – (NSInteger)numberOfSectionsInTableView:(UITableView *) tableView;
- – (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section;

Table Views

- Our Data Source methods pass in a UITableView
 - Our Data Source can control more than one Table View
 - 99% of the time they won't

Table View

- – (NSInteger)numberOfSectionsInTableView:
 (UITableView *)tableView {
 return 2;
 }
- Tells our Table View we have 2 sections

Table Views

- ```
- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section {
 if (section == 0) {
 return 2;
 } else if (section == 1) {
 return 4;
 }
}
```

- 2 Rows in Section 0
- 4 Rows in Section 1

# Table Views

- – (NSString \*)tableView:(UITableView \*)  
tableView titleForHeaderInSection:(NSInteger)  
section;
- – (NSString \*)tableView:(UITableView \*)  
tableView titleForFooterInSection:(NSInteger)  
section;

# Table View

- ```
- (NSString *)tableView:(UITableView *)  
tableView titleForHeaderInSection:(NSInteger)  
section {  
    if (section == 0) {  
        return @"California";  
    } else if (section == 1) {  
        return @"New York";  
    }  
}
```

Table Views

- Each row of our table is a Cell
 - Remember single column
- Cells are what display our data

TableView Cells

- Cells have content:
 - Title
 - Subtitle
 - Image

TableView Cells



TableView Cells

- Cells can also have accessory views
 - Disclosure Indicator
 - Detail Disclosure Indicator
- Along the right edge of the cell

Table View Cells



Table View Cells

- How do we fill in our Table Cells?
- Table View Data Source
- – (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath

Table View Cells

- Note:
 - iOS likes to reuse Table View Cells for performance reasons
 - We will fetch a previously created cell that is marked for reuse
 - These previously created cells are normally cells that have been scrolled off screen

Table View Cells

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:
(NSIndexPath *)indexPath {

    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier] autorelease];
    }

    return cell;
}
```

TableView Cell

- In this method we can customize our TableView Cell
- `cell.textLabel.text = @"Hello World";`
- `cell.detailTextLabel.text = @"Subtitle";`
- `cell.accessoryType =
UITableViewCellAccessoryDisclosureIndicator;`

Table View Cells

- Now that we have cells created, how can we do something once a cell is tapped?
- Table View Delegate
 - – (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath

Navigation Controllers

Navigation Controllers

- Since we setup our project as a Navigation based application, all the hard work is done for us
- Setting up the Navigation Controller, etc

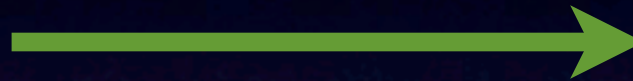
Navigation Controllers

- Easy way to push and pop views onto the screen
 - Push to go forward
 - Pop to go back

Navigation Controller

- Keep in mind we are pushing and popping View Controllers
 - Navigation Controller will get the View from the View Controller and display it for us

Navigation Controller



Navigation Controller



Navigation Controller



View Controller has a title property

Navigation Controller



View Controller's view

Navigation Controller

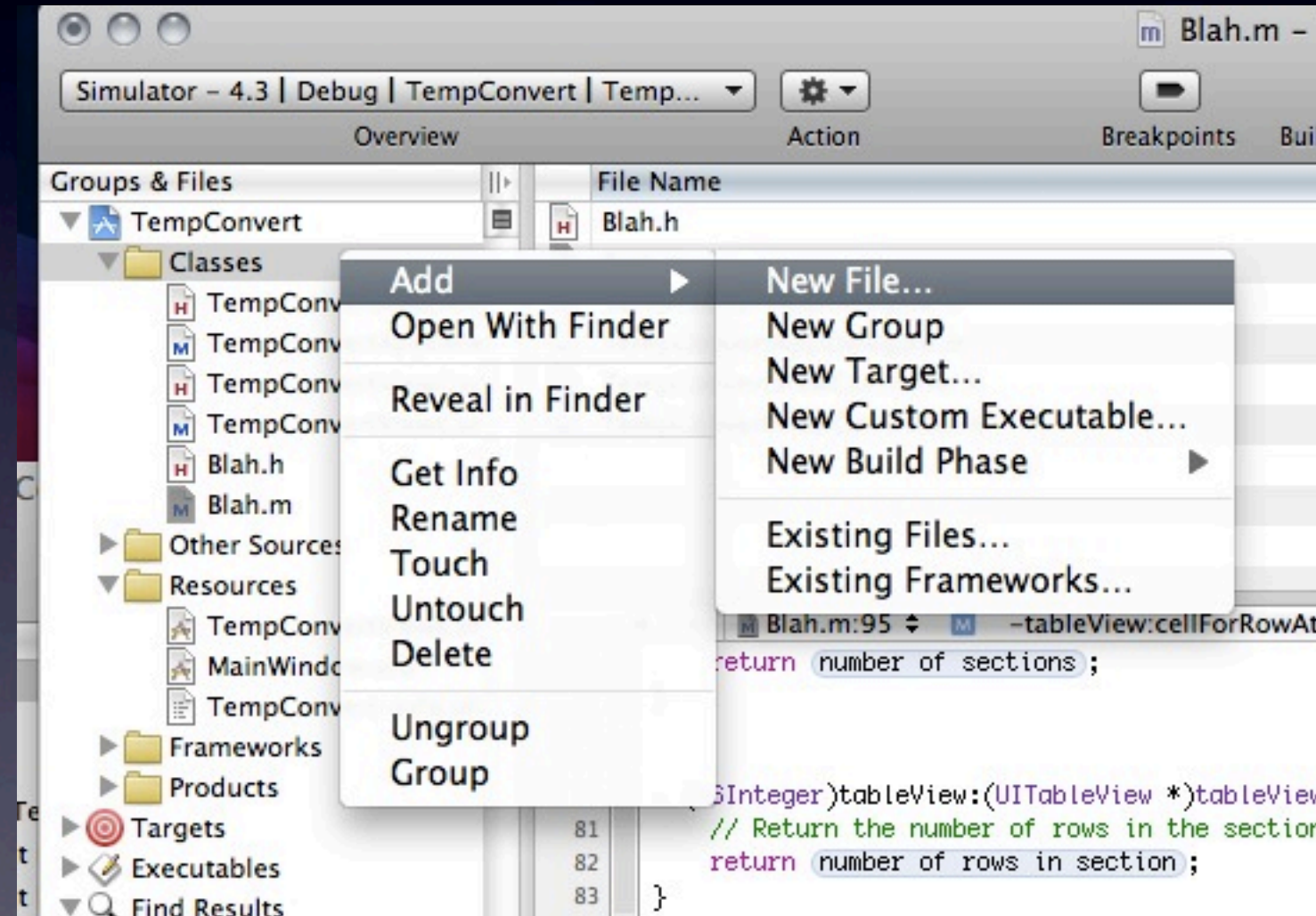


Back Button will use title from previous View Controller

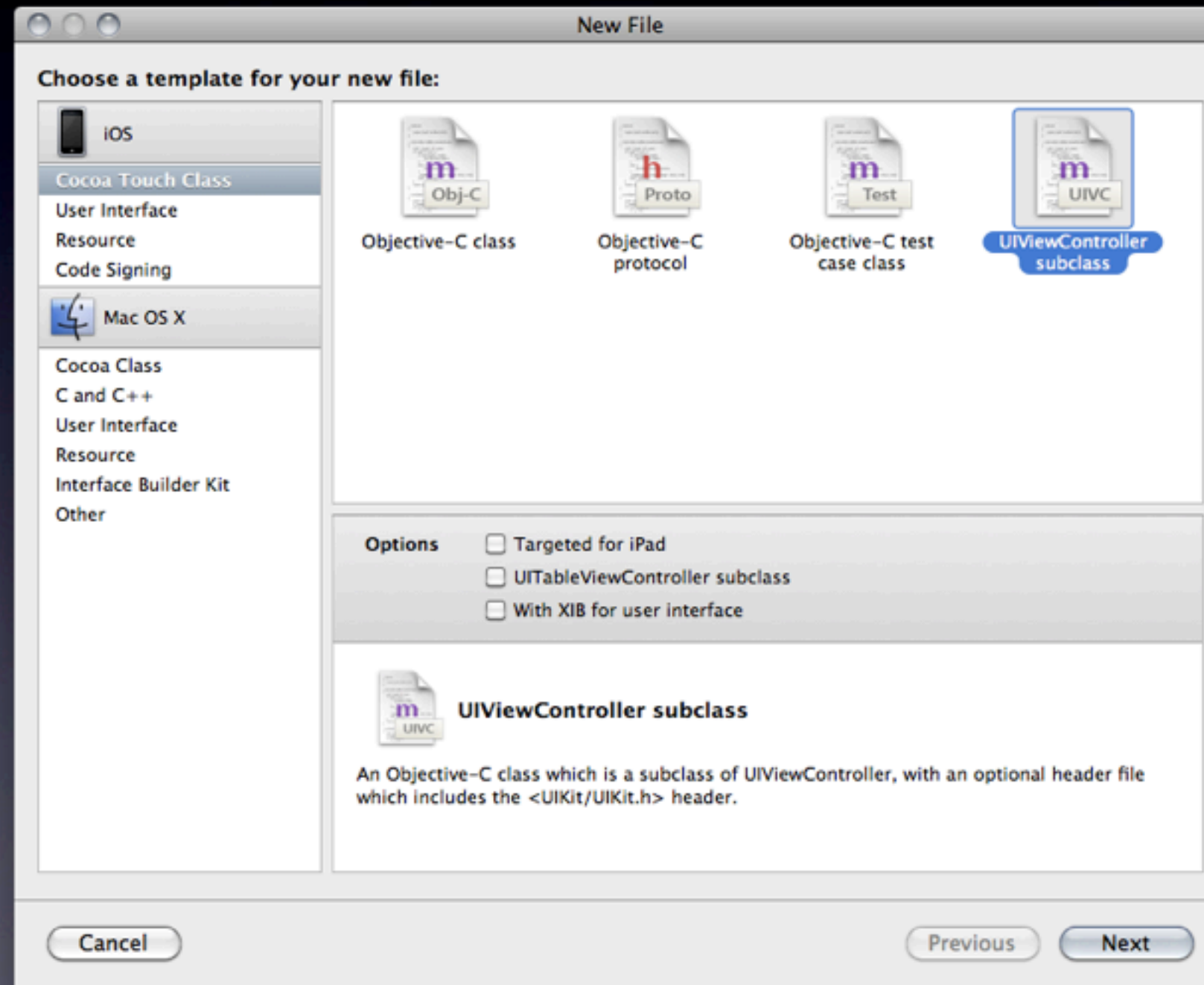
Navigation Controller

- First we need to create other views to actually go to

Navigation Controller



Navigation Controller



Navigation Controller

- Now how do we “push” to our new View Controller
 - `UIViewController *vcToPush = [[UIViewController alloc] initWithNibName:@"Interface" bundle:nil];`
 - `[self.navigationController pushViewController:vc animated:YES];`

Navigation Controller

- What if we want to pop back?
 - `[self.navigationController popViewControllerAnimated:YES];`

Navigation Controller

- Keep in mind that the Navigation Controller will create our back button for us, we will hardly ever want to pop back ourselves
- Let the user go back on their own

Questions?

brentnycum@gmail.com