# iPhone App Dev

Lesson 6

# Source Codes

https://github.com/makzan/ios-dev-course-example

# Contact

makzan@42games.net

# Exercise

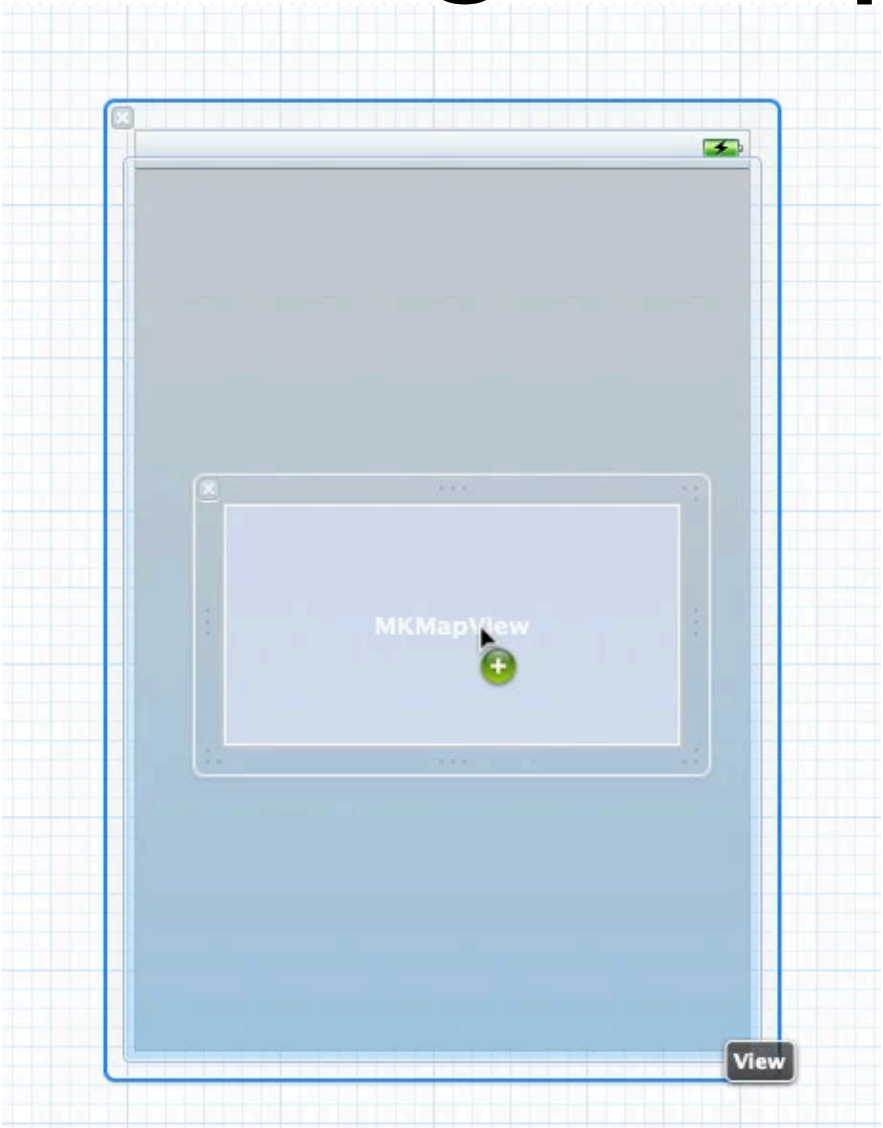
- What app are you developing?
- Any ideas worth discussing?
- Any great app design worth sharing?

# Summary

- Showing a map
- Configuring frameworks
- Displaying a map region
- Adding pin annotation to map
- Using NSArray
- Using custom pin image
- Detecting pin interaction
- Finding current location
- Custom annotation class

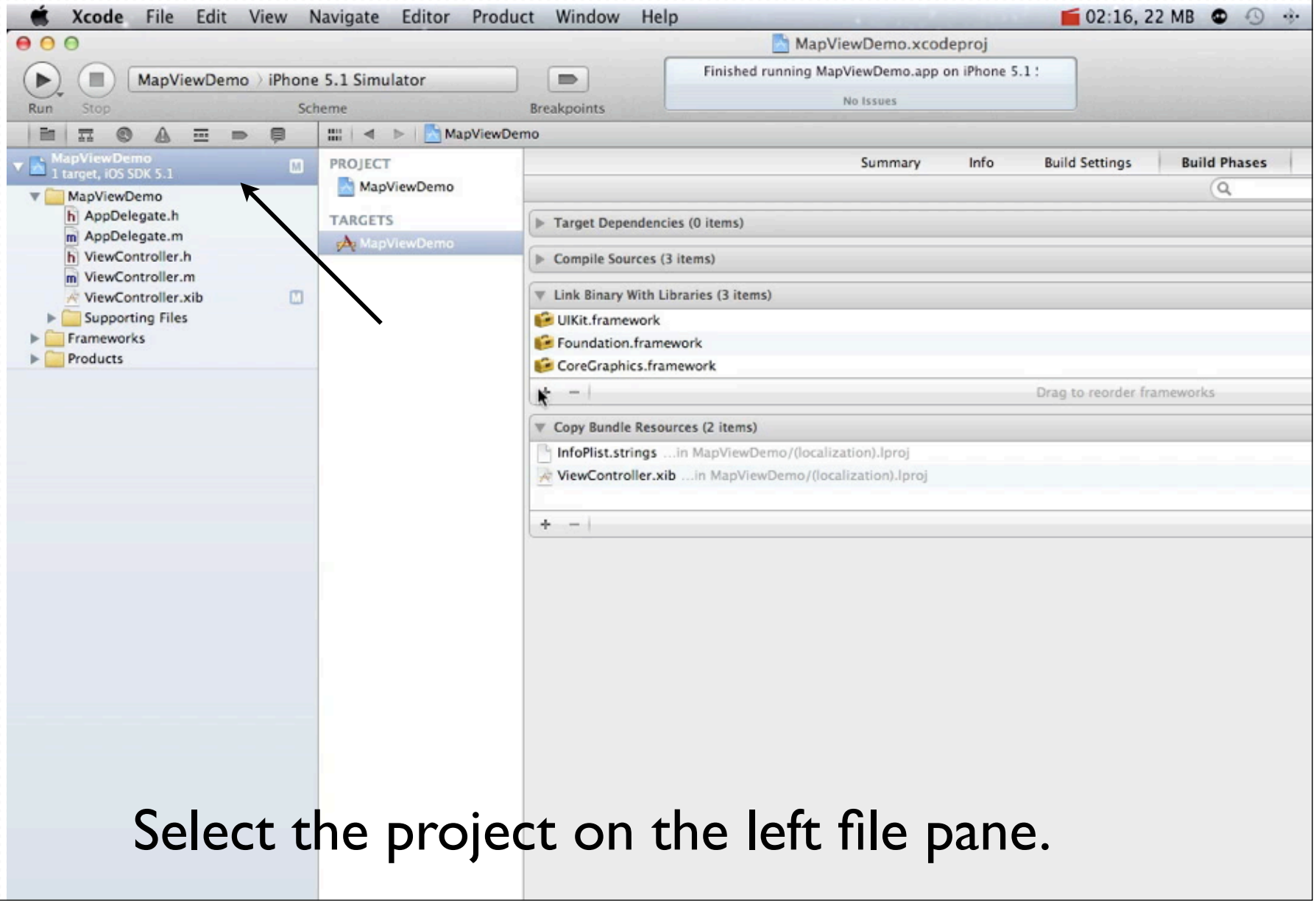
# Showing a map

# Showing a map



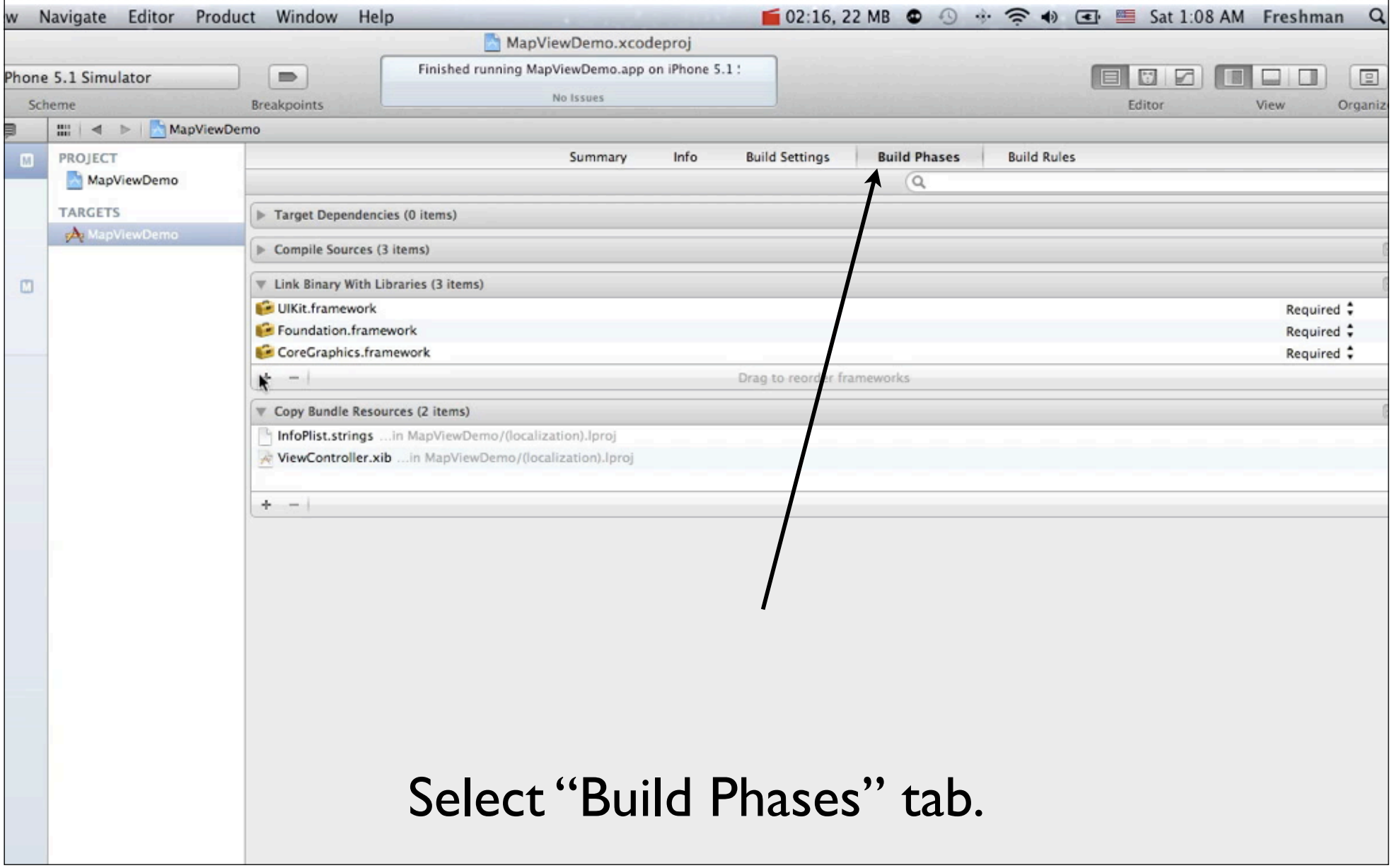

Drag the map to the view in xib.

# Showing a map



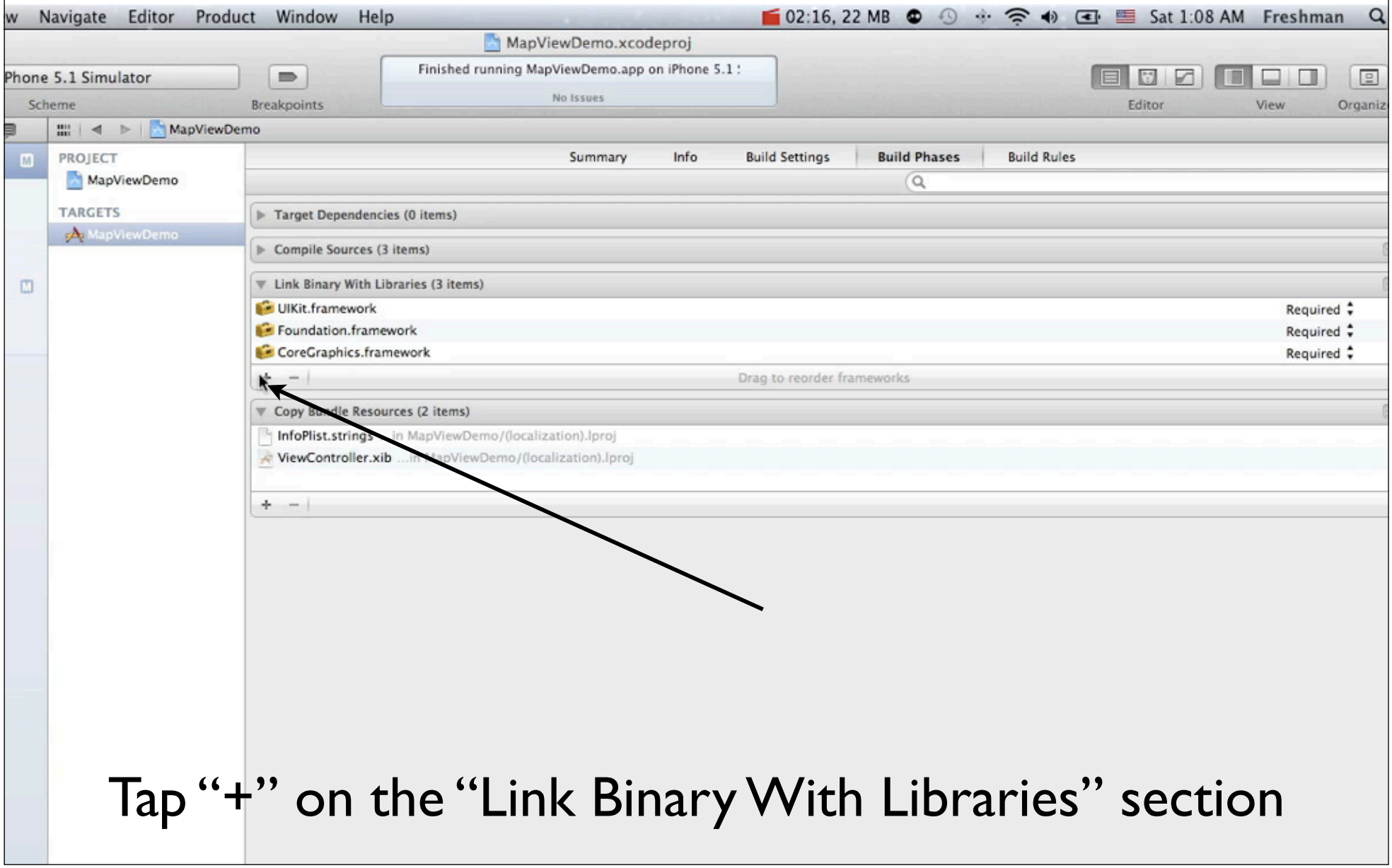

Select the project on the left file pane.

# Showing a map



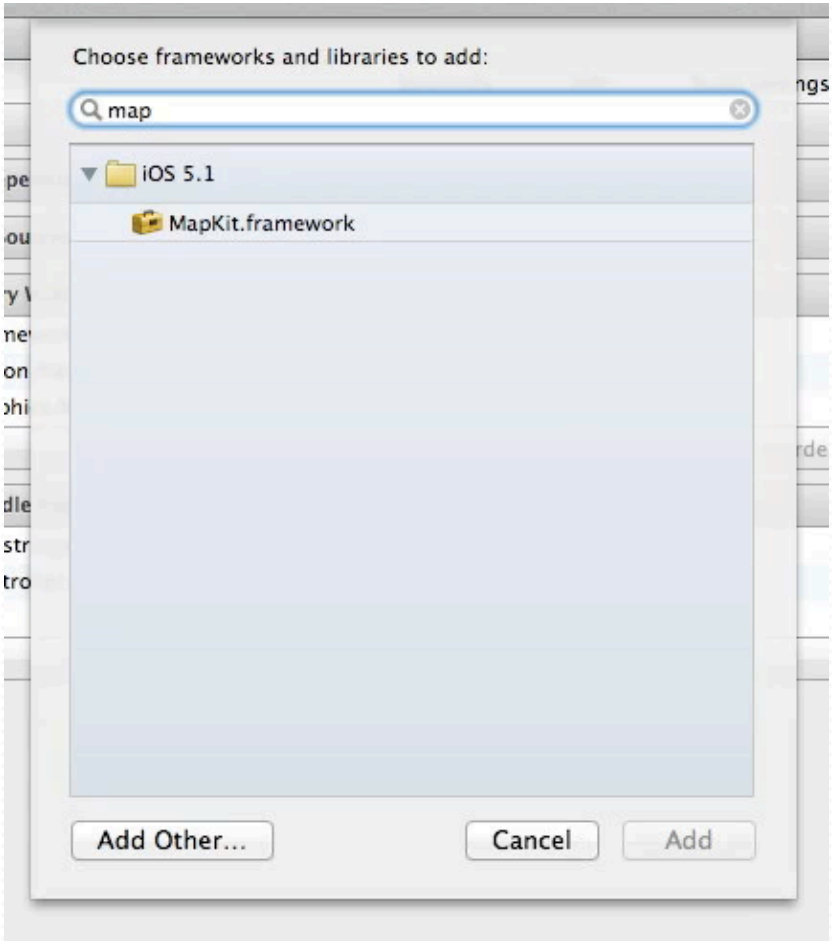

Select "Build Phases" tab.

# Showing a map



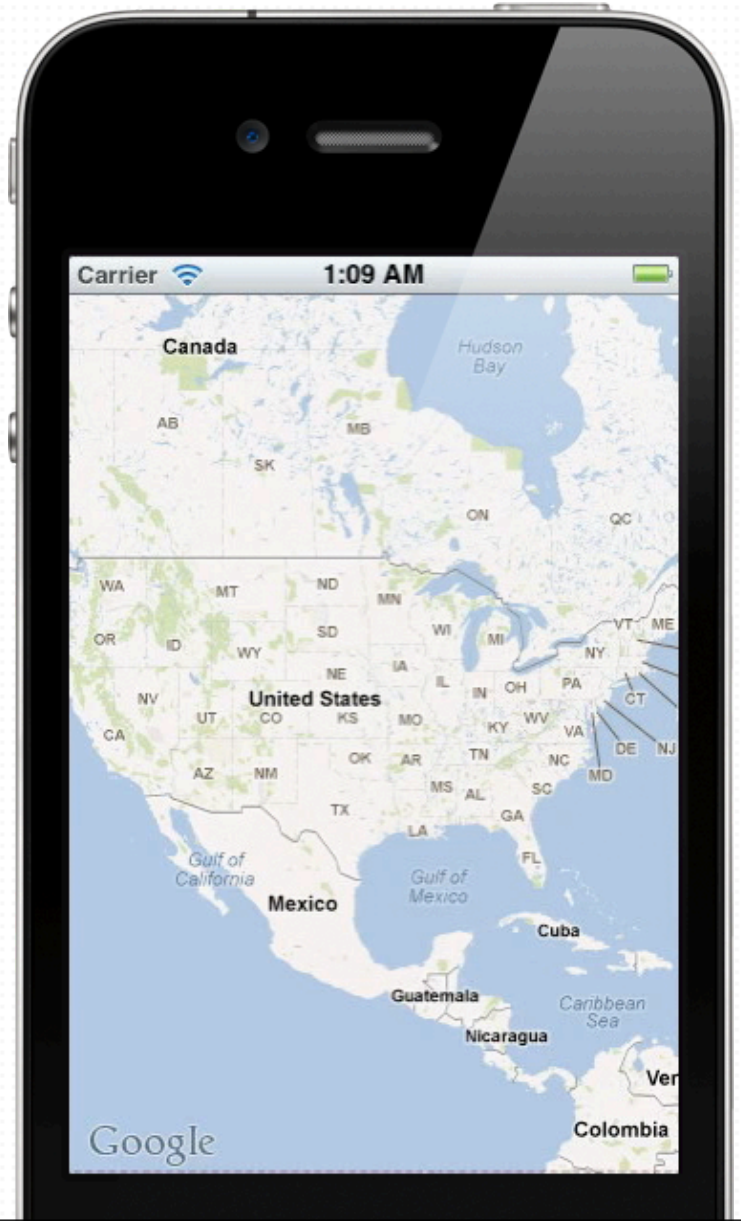

Tap "+" on the "Link Binary With Libraries" section

# Showing a map



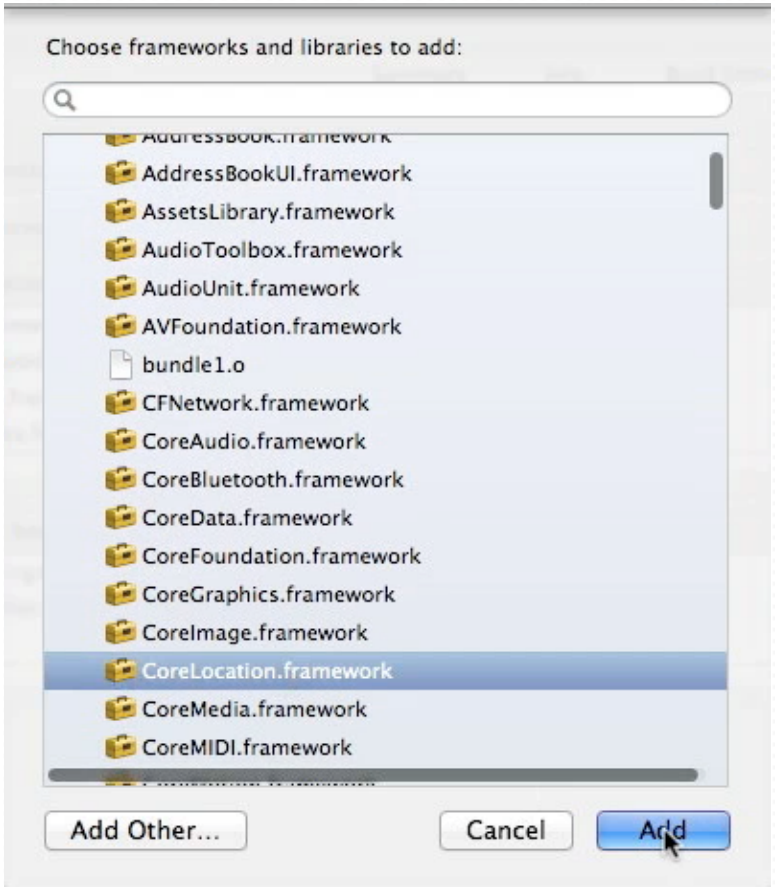

Select and add the MapKit.framework

# Showing a map



# Configuring frameworks

# Frameworks




There are many frameworks that we may use.

# Frameworks

- Some framework examples:
- GameKit - Game Center.
- CFNetwork - Network related.
- MapKit - Map views
- CoreLocation - Location, GPS, compass

# Displaying a map region

# Displaying map region

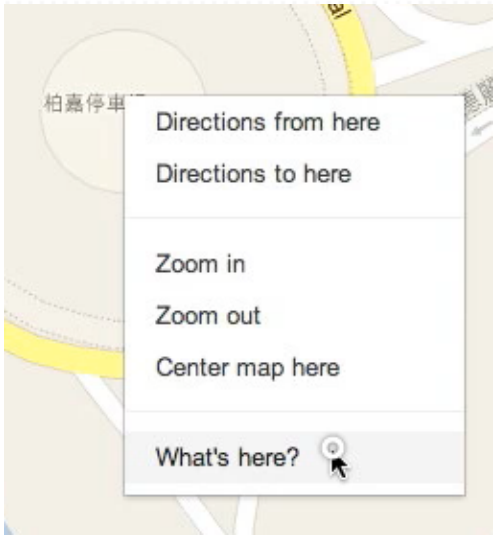
Make sure we included the MapKit header

```
#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>

@interface ViewController : UIViewController
@property (retain, nonatomic) IBOutlet MKMapView *mapView;

@end
```

Link the map view to an IBOutlet

# Displaying map region

- How we know the latitude and longitude of a place in map?



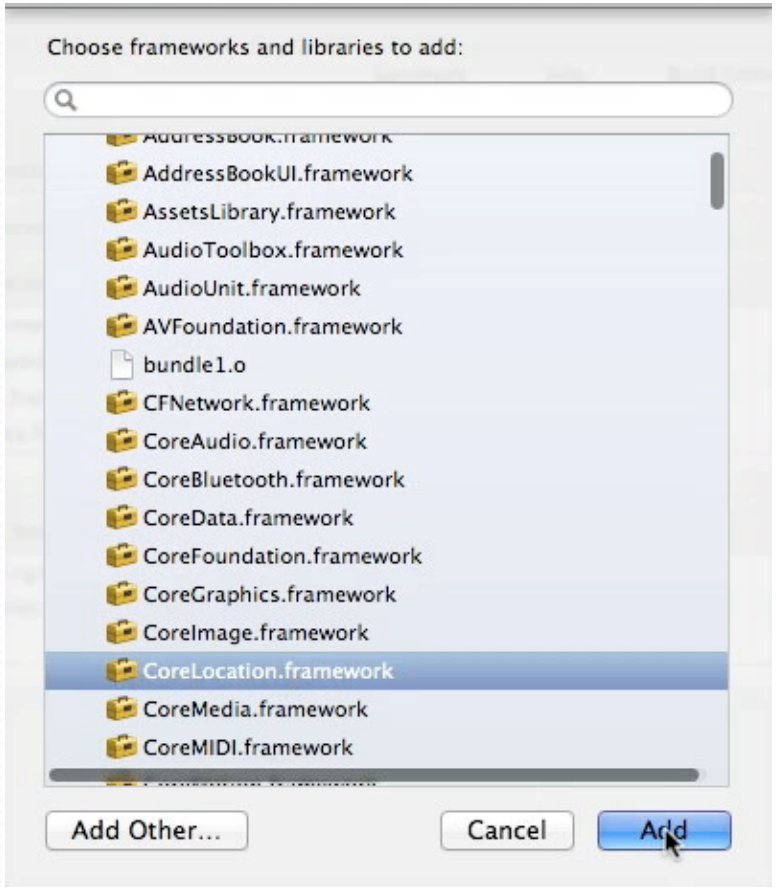

Use Google Map, right click and choose “What’s here?”

# Displaying map region

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // we use CLLocationCoordinate2D to store a coordinate.
    CLLocationCoordinate2D location = CLLocationCoordinate2DMake(22.191856, 113.543186);

    // We not only need a point but also a region with distance in order to so it on map.
    MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance(location, 500.0f, 500.0f);

    self.mapView.region = region;
```

Set the location and region of the map with the code

# Displaying map region



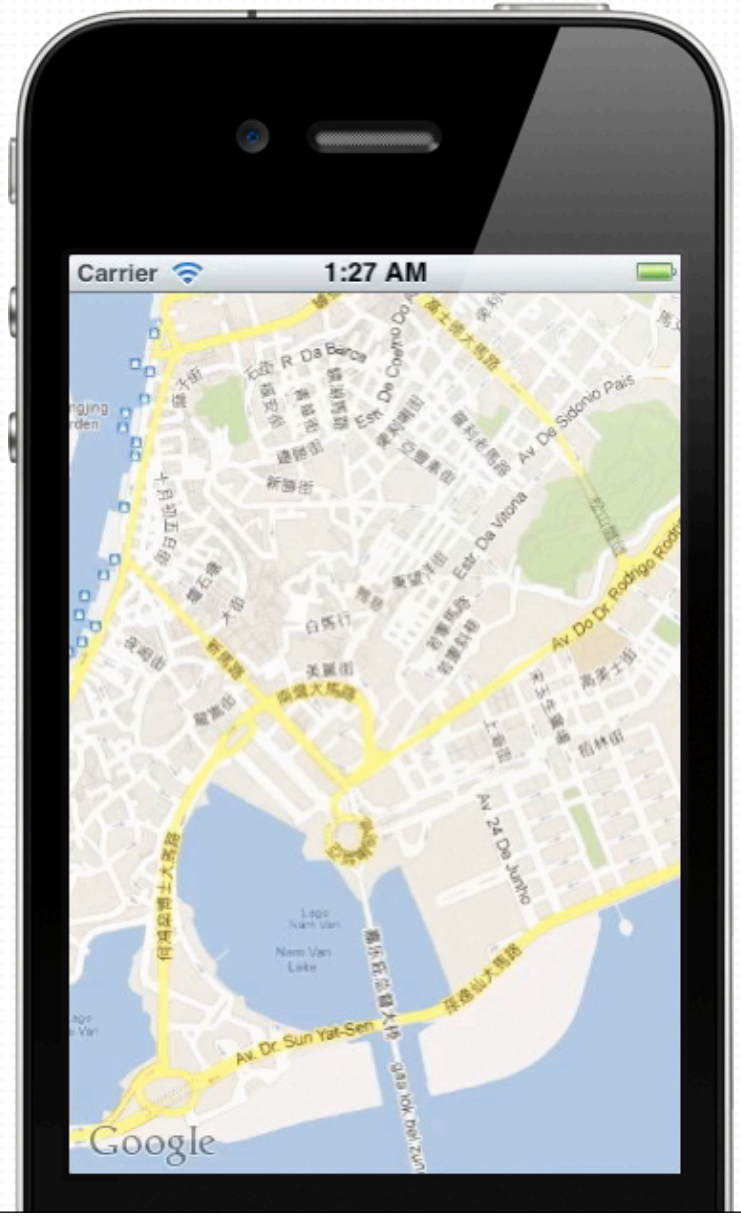

We'll need CoreLocation framework too.

# Displaying map region

```
#import "ViewController.h"
#import <CoreLocation/CoreLocation.h>
```

Include the CoreLocation in ViewController.m

# Displaying map region

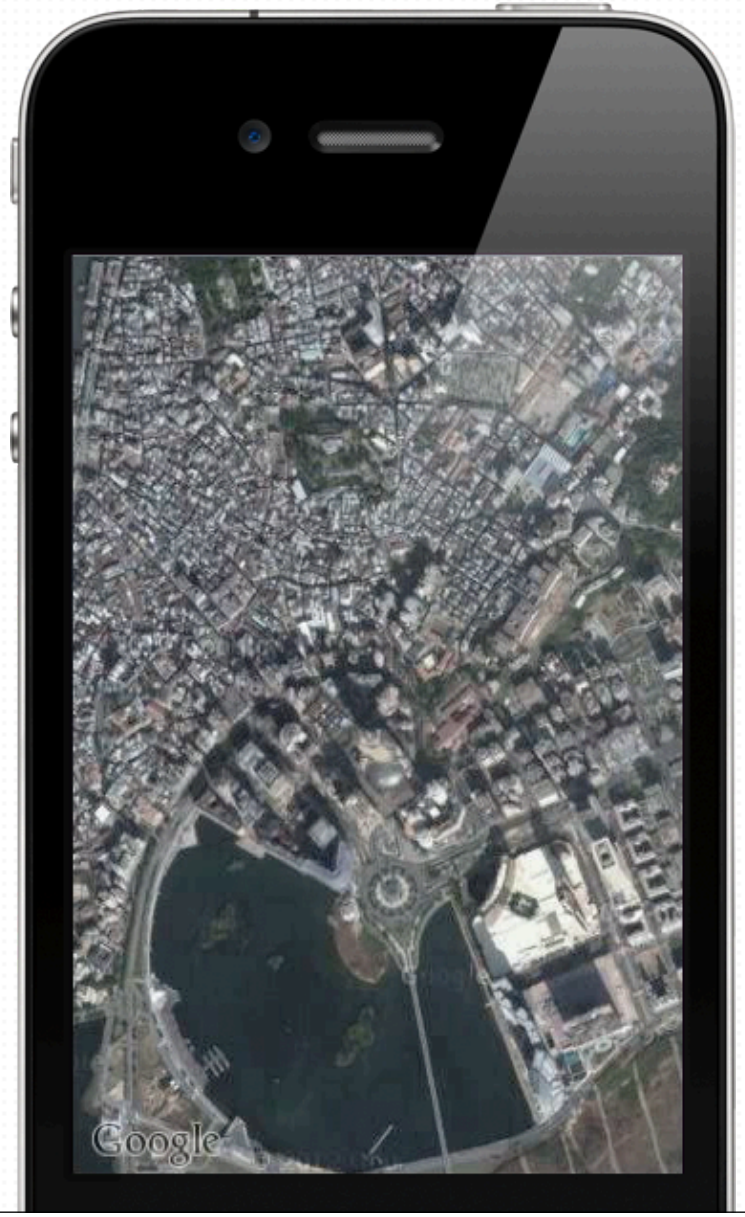


# Displaying map region

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // we use CLLocationCoordinate2D to store a coordinate.
    CLLocationCoordinate2D location = CLLocationCoordinate2DMake(22.191856, 113.543186);

    // We not only need a point but also a region with distance in order to so it on map.
    MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance(location, 1000, 1000);

    mapView.region = region;


    // and we may also set the map type.
    mapView.mapType = MKMapTypeSatellite;
}
```

And we can set the map type to Satellite.

# Displaying map region



# Adding pin annotation

# Adding pin annotation

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // we use CLLocationCoordinate2D to store a coordinate.
    CLLocationCoordinate2D location = CLLocationCoordinate2DMake(22.191856, 113.543186);

    // We not only need a point but also a region with distance in order to so it on map.
    MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance(location, 1000, 1000);

    mapView.region = region;

    // add an map annotation (pin)
    MKPointAnnotation *pin = [[MKPointAnnotation alloc] init];
    pin.coordinate = CLLocationCoordinate2DMake(22.188826, 113.550729);
    pin.title = @"宋玉生公園";
    pin.subtitle = @"澳門新口岸填海區";

    [mapView addAnnotation:pin];
    [pin release];
}
```

We can use MKPointAnnotation to add pin to the map.

# Adding pin annotation

# Using NSArray

# Using NSArray

- Creating a mutable array

  [[NSMutableArray alloc] initWithCapacity: 3];

# Using NSArray

- Adding object into array

  [array addObject:something];

# Using NSArray

- Accessing object from array

  [array objectAtIndex:3];

# (Bonus) New Syntax

Previously:

```
number = [NSNumber numberWithInt:100];
```

Now:

```
number = @100;
```

# (Bonus) New Syntax

Previously:

```
number = [NSNumber numberWithFloat:123.45];
```

Now:

```
number = @123.45;
```

# (Bonus) New Syntax

Previously:

```
number = [NSNumber numberWithBool:YES];
```

Now:

```
number = @YES;
```

# (Bonus) New Syntax

Previously:

```
array = [NSArray arrayWithObjects: a, b, c];
```

Now:

```
array = @[ a, b, c];
```

# (Bonus) New Syntax

Previously:

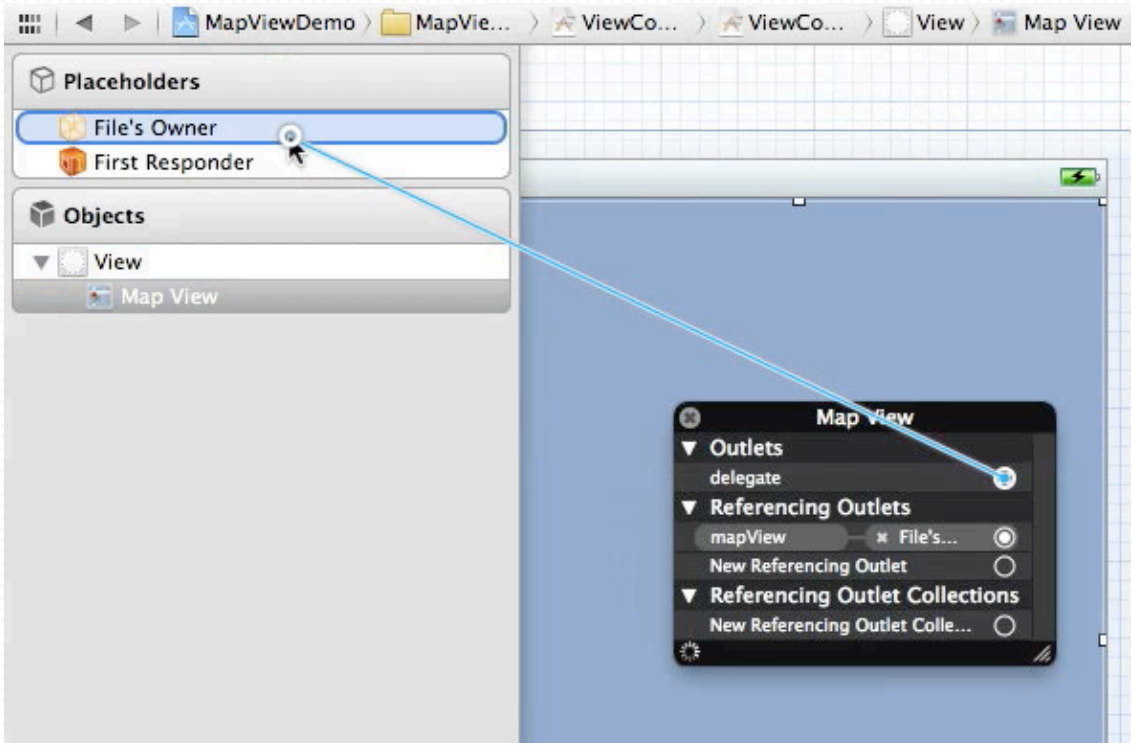
```
dictionary = [NSDictionary
dictionaryWithObjects:@[obj1, obj2, obj3] for
keys:@[key1, key2, key3]];
```

Now:

```
dictionary = @{ key1:obj1, key2:obj2, key3:obj3};
```

# Setting pin color

# Setting pin color




First, we need to set the map view delegate to ViewController

# Setting pin color

```
#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>

@interface ViewController : UIViewController <MKMapViewDelegate>
@property (retain, nonatomic) IBOutlet MKMapView *mapView;

@end
```

Add the <MKMapViewDelegate> to ViewController.h

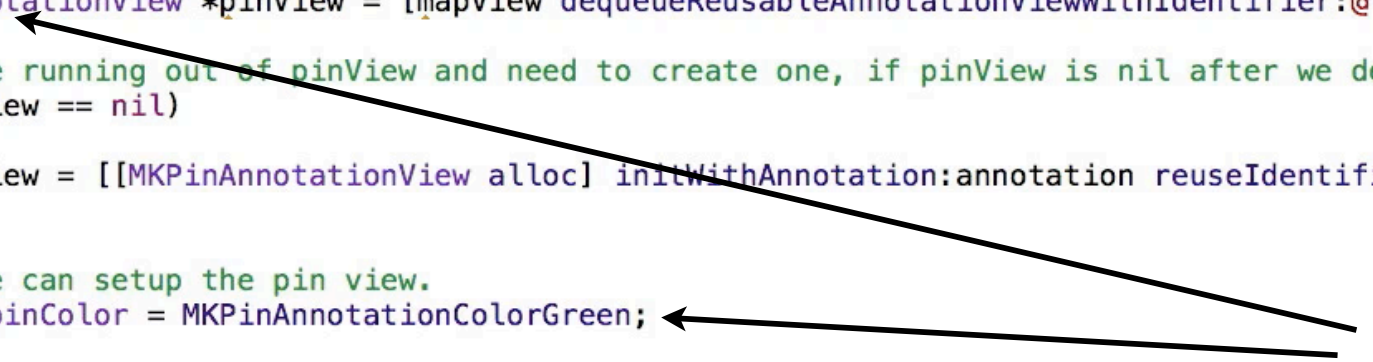
# Setting pin color

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView viewForAnnotation:(id <MKAnnotation>)annotation
{
    // reuse any not-using Pin View with identifier "greenPin"
    MKPinAnnotationView *pinView = [mapView dequeueReusableAnnotationViewWithIdentifier:@"greenPin"];

    // we are running out of pinView and need to create one, if pinView is nil after we dequeue one.
    if (pinView == nil)
    {
        pinView = [[MKPinAnnotationView alloc] initWithAnnotation:annotation reuseIdentifier:@"greenPin"];
    }

    // now we can setup the pin view.
    pinView.pinColor = MKPinAnnotationColorGreen;

    // we need to assign the annotation information to the view.
    // Otherwise it will not display the information.
    pinView.annotation = annotation;

    return pinView;
}
```

Add the **mapView:viewForAnnotation:** delegate method

# Setting pin color

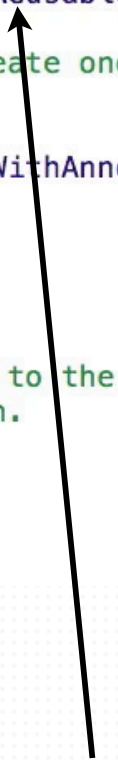
```
- (MKAnnotationView *)mapView:(MKMapView *)mapView viewForAnnotation:(id <MKAnnotation>)annotation
{
    // reuse any not-using Pin View with identifier "greenPin"
    MKPinAnnotationView *pinView = [mapView dequeueReusableAnnotationViewWithIdentifier:@"greenPin"];

    // we are running out of pinView and need to create one, if pinView is nil after we dequeue one.
    if (pinView == nil)
    {
        pinView = [[MKPinAnnotationView alloc] initWithAnnotation:annotation reuseIdentifier:@"greenPin"];
    }

    // now we can setup the pin view.
    pinView.pinColor = MKPinAnnotationColorGreen;

    // we need to assign the annotation information to the view.
    // Otherwise it will not display the information.
    pinView.annotation = annotation;

    return pinView;
}
```



MKPinAnnotationView is a subclass of MKAnnotationView.
It contains the default pin image.
It allows us to change color among green, purple and red.

# Setting pin color

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView viewForAnnotation:(id <MKAnnotation>)annotation
{
    // reuse any not-using Pin View with identifier "greenPin"
    MKPinAnnotationView *pinView = [mapView dequeueReusableAnnotationViewWithIdentifier:@"greenPin"];

    // we are running out of pinView and need to create one, if pinView is nil after we dequeue one.
    if (pinView == nil)
    {
        pinView = [[MKPinAnnotationView alloc] initWithAnnotation:annotation reuseIdentifier:@"greenPin"];
    }

    // now we can setup the pin view.
    pinView.pinColor = MKPinAnnotationColorGreen;

    // we need to assign the annotation information to the view.
    // Otherwise it will not display the information.
    pinView.annotation = annotation;

    return pinView;
}
```



The dequeue calling let us reuse the pin view.

# Setting pin color

- Annotation is the data
- MKAnnotationView is the view

# Setting pin color

- MKAnnotationView contains annotation property to display the information in view.

# Setting pin color

- Annotation is lightweight, such bytes of data.
- MKAnnotationView is heavyweight. They are images!!

# Setting pin color

- We will have a pool of MKAnnotationView instances.
- We mark it as ‘using’ when user can see it.
- We mark it as ‘not-using’ when it is out of screen.
- So we can reuse ‘not-using’ view.

# Setting pin color

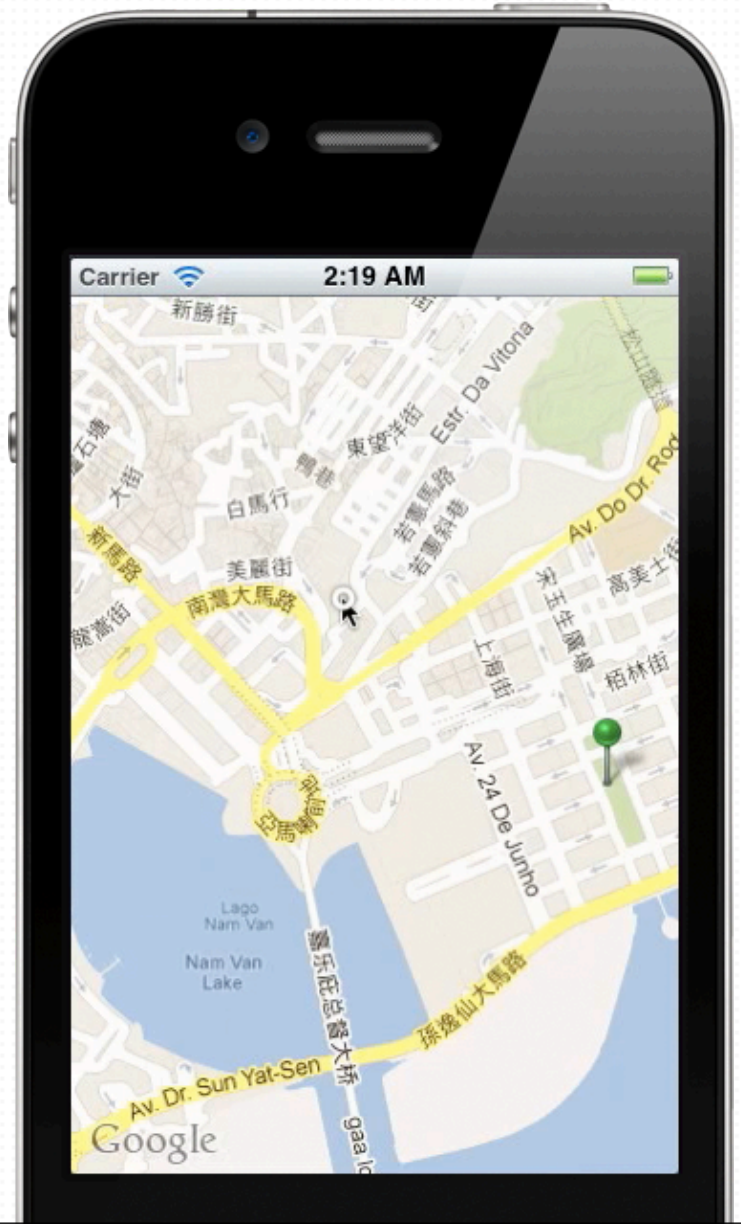
- Why need reuse?
- It's because creating new image instance is slow.
- It's because most of them are very similar.
- It's because why iPhone is smooth and consume less CPU.

# Setting pin color

```
// reuse any not-using Pin View with identifier "greenPin"
MKPinAnnotationView *pinView = [mapView dequeueReusableAnnotationViewWithIdentifier:@"greenPin"];

// we are running out of pinView and need to create one, if pinView is nil after we dequeue one.
if (pinView == nil)
{
    pinView = [[MKPinAnnotationView alloc] initWithAnnotation:annotation reuseIdentifier:@"greenPin"];
}
```

Here is the re-use code again.

# Setting pin color

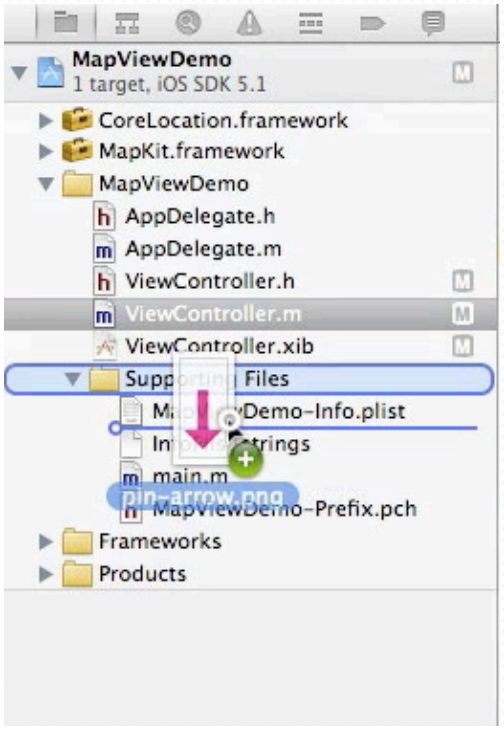


# Using custom pin image

# Using custom pin image


First, let's prepare an image for the pin.

# Using custom pin image



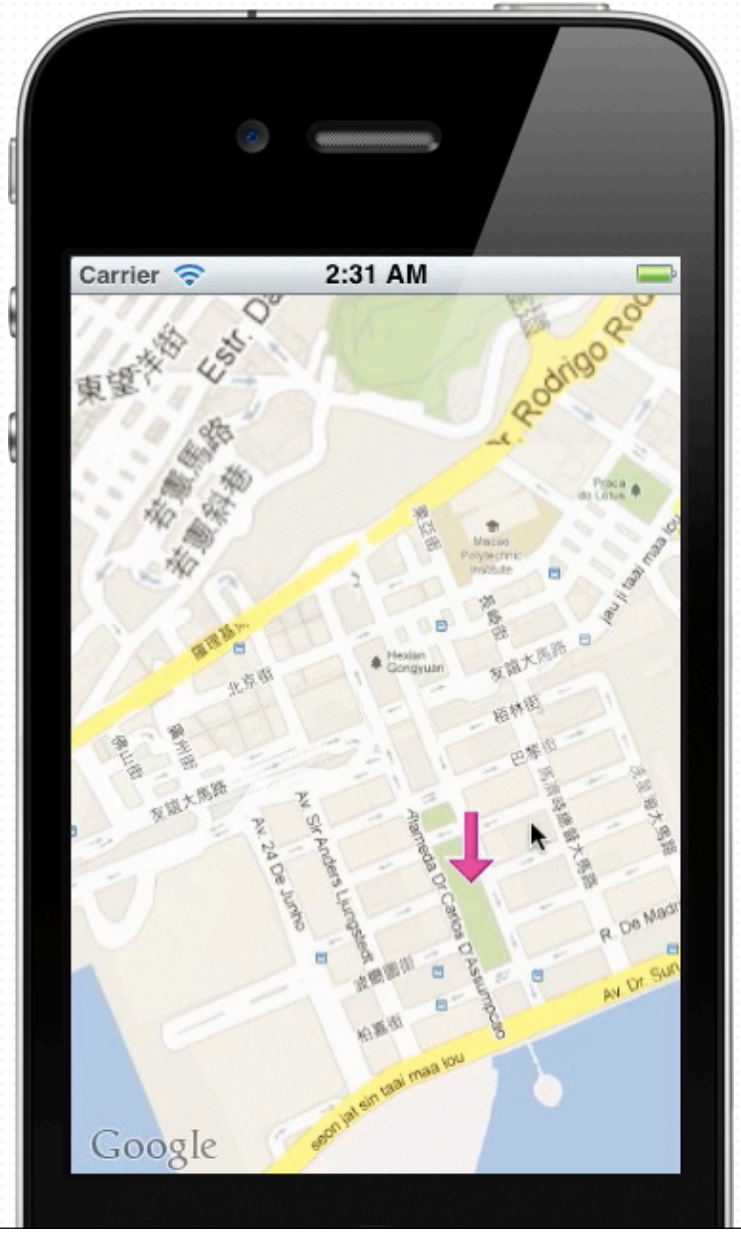

Then add the image file to the project.

# Using custom pin image

```
// we can set the image of the pin.
pinView.image = [UIImage imageNamed:@"pin-arrow.png"];
```

Instead of setting pin color, now we set the image of the annotation.

# Using custom pin image



# Adding multiple pins

# Adding multiple pins

```
NSMutableArray *pinsArray = [[NSMutableArray alloc] initWithCapacity:10];

{
    // add an map annotation (pin)
    MKPointAnnotation *pin = [[MKPointAnnotation alloc] init];
    pin.coordinate = CLLocationCoordinate2DMake(22.188826, 113.550729);
    pin.title = @"宋玉生公園";
    pin.subtitle = @"澳門新口岸填海區";

    [pinsArray addObject:pin];
}

{
    // add another pin
    MKPointAnnotation *pin = [[MKPointAnnotation alloc] init];
    pin.coordinate = CLLocationCoordinate2DMake(22.188473, 113.543454);
    pin.title = @"亞馬喇前地";
    pin.subtitle = @"澳門舊大橋橋口";

    [pinsArray addObject:pin];
}
```

Let's create an array to store our annotation pins.
And we have two annotation data that is added to the array.

# Adding multiple pins

```
[mapView addAnnotations:pinsArray];
```

Now we use **addAnnotations:** to add multiple annotations from array.

# Adding multiple pins

# Detecting pin interaction

# Detecting pin interaction

```
- (void)mapView:(MKMapView *)mapView didSelectAnnotationView:(MKAnnotationView *)view
{
    NSLog(@"Tapped Pin with title: %@", view.annotation.title);
}
```

We can use this delegate method to know which annotation is selected.

# Detecting pin interaction

```
// add a button
pinView.rightCalloutAccessoryView = [UIButton buttonWithType:UIButtonTypeDetailDisclosure];
```

To make the pop up tappable, we need to assign the **rightCalloutAccessoryView** of the annotation view to a button.

# Detecting pin interaction

```
- (void)mapView:(MKMapView *)mapView annotationView:(MKAnnotationView *)view
    calloutAccessoryControlTapped:(UIControl *)control
{
    NSLog(@"Detail button is tapped on pin: %@", view.annotation.title);
}
```

Now we can use this delegate method to know the pop up is tapped.

# Detecting pin interaction

# Finding current location

# Finding current location

```
// listen to the current location
CLLocationManager *locationManager = [[CLLocationManager alloc] init];
[locationManager startUpdatingLocation];
```

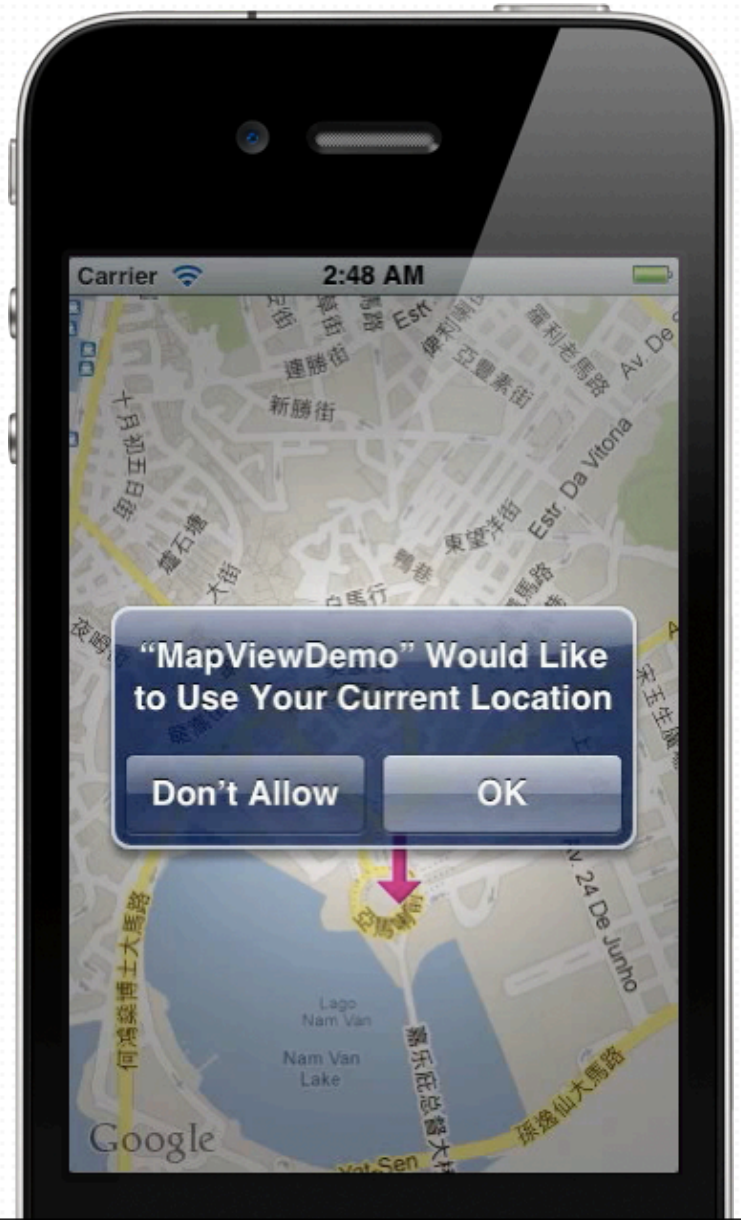
Add the location manager in to viewDidLoad

# Finding current location

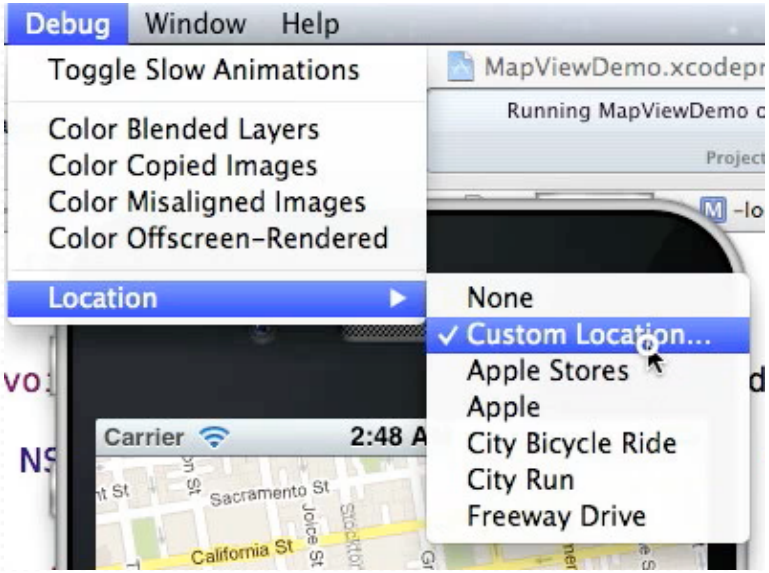
```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation
{
    mapView.centerCoordinate = newLocation.coordinate;

    MKPointAnnotation *pin = [[MKPointAnnotation alloc] init];
    pin.coordinate = newLocation.coordinate;
    pin.title = @"您的位置";
    pin.subtitle = @"";

    [mapView addAnnotation:pin];

    [pin release];
}
```

Afterwards, we are able to get the updated location.

# Finding current location



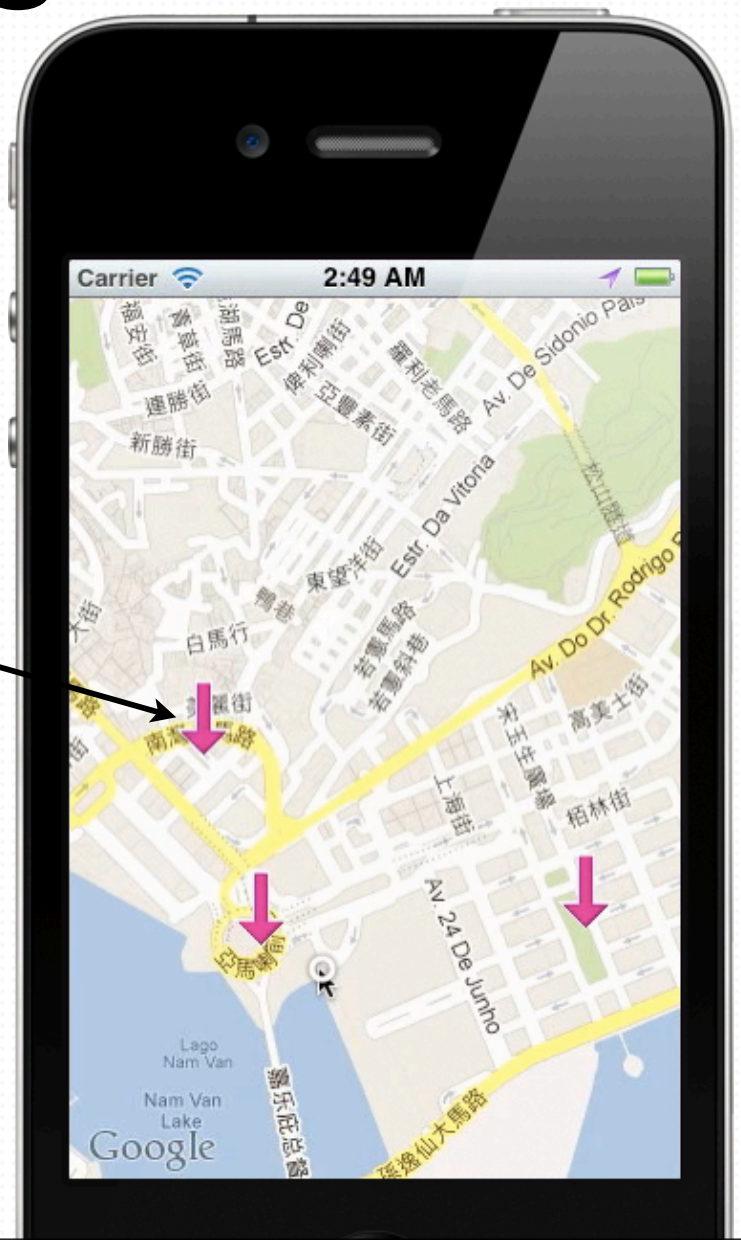
# Finding current location




In simulator, we cannot get the real location. However, we can create a mock location.

# Finding current location

This is our mock current location

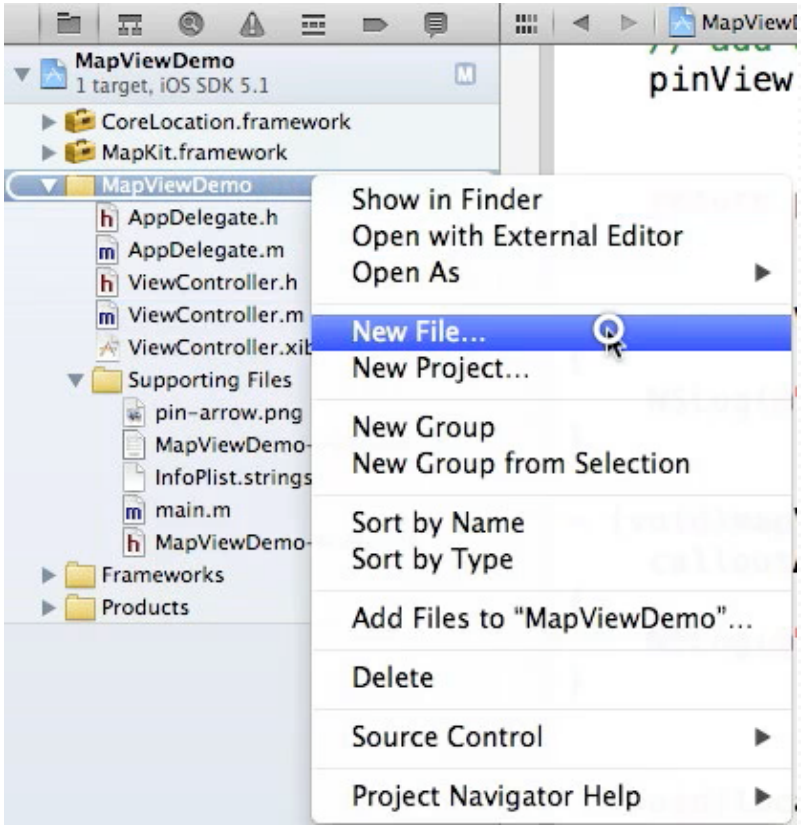


# Custom annotation class

# Custom annotation class

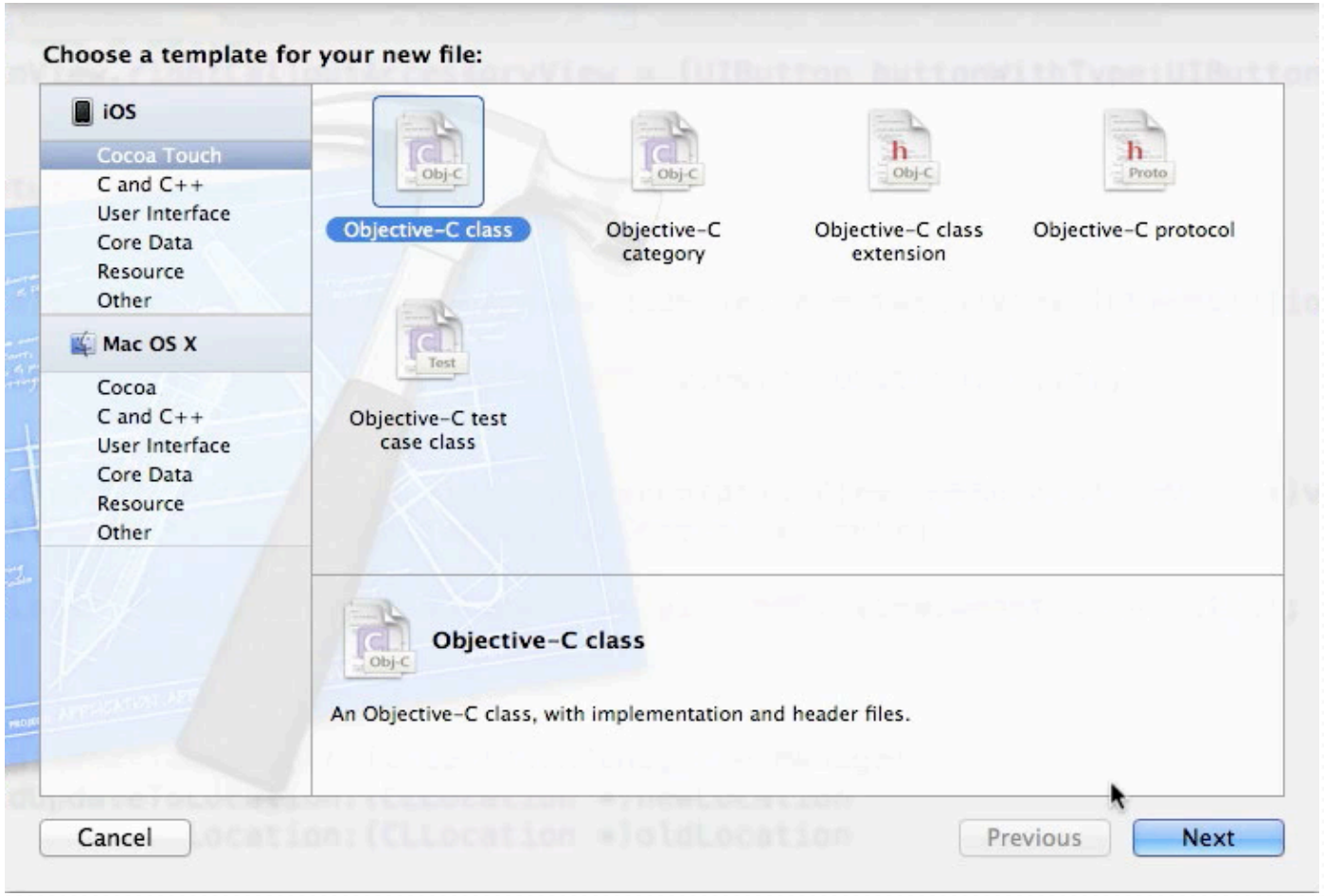
- The default MKPointAnnotation only contains basic data, coordinate / title / subtitle
- Most of the time, we store more information on the location.
- In this case, we need a custom class.

# Custom annotation class



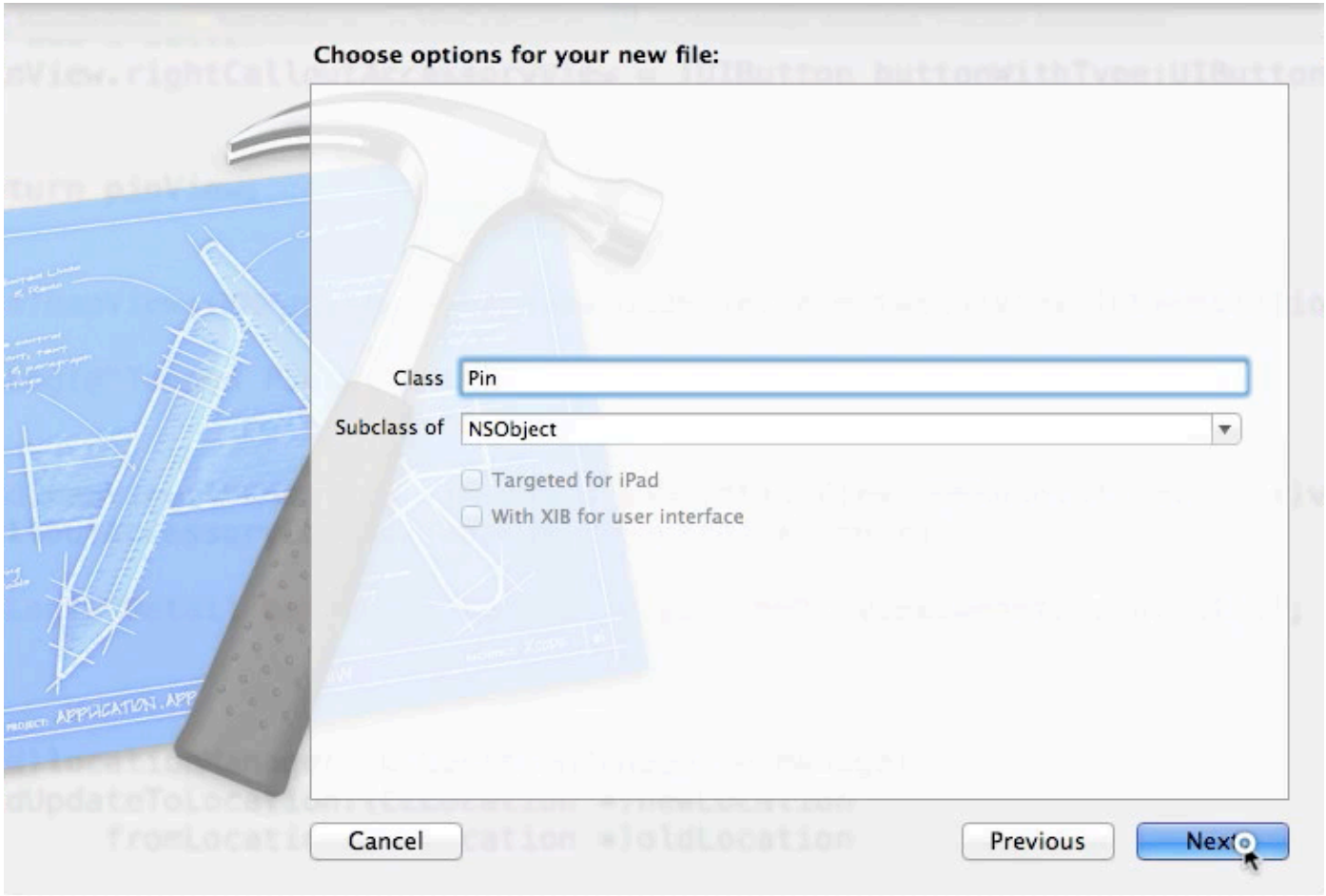

Right click the folder and choose “New File...”

# Custom annotation class




Choose Objective-C class and press Next

# Custom annotation class




Give a name to our annotation class.

# Custom annotation class

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface Pin : NSObject <MKAnnotation>

// required by the MKAnnotation
@property (nonatomic) CLLocationCoordinate2D coordinate;

// optional but we need these
@property (nonatomic, copy) NSString *title;
@property (nonatomic, copy) NSString *subtitle;

// and extra data
@property (nonatomic) BOOL isCurrentLocationMark;

@end
```

In the header, we add <MKAnnotation> to let XCode knows this class provide annotation information.

# Custom annotation class

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface Pin : NSObject <MKAnnotation>

// required by the MKAnnotation
@property (nonatomic) CLLocationCoordinate2D coordinate;

// optional but we need these
@property (nonatomic, copy) NSString *title;
@property (nonatomic, copy) NSString *subtitle;

// and extra data
@property (nonatomic) BOOL isCurrentLocationMark;

@end
```

We must provide the **coordinate** property.

# Custom annotation class

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface Pin : NSObject <MKAnnotation>

// required by the MKAnnotation
@property (nonatomic) CLLocationCoordinate2D coordinate;

// optional but we need these
@property (nonatomic, copy) NSString *title;
@property (nonatomic, copy) NSString *subtitle;

// and extra data
@property (nonatomic) BOOL isCurrentLocationMark;

@end
```

Optionally, we provide title and subtitle.

# Custom annotation class

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface Pin : NSObject <MKAnnotation>

// required by the MKAnnotation
@property (nonatomic) CLLocationCoordinate2D coordinate;

// optional but we need these
@property (nonatomic, copy) NSString *title;
@property (nonatomic, copy) NSString *subtitle;

// and extra data
@property (nonatomic) BOOL isCurrentLocationMark;

@end
```

Then, we can declare whatever extra data we want.

# Custom annotation class

```
#import "Pin.h"


@implementation Pin
@synthesize coordinate;
@synthesize title;
@synthesize subtitle;
@synthesize isCurrentLocationMark;

@end
```

Make sure we have @synthesize configure properly in the .m file.

# Custom annotation class

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation
{
    mapView.centerCoordinate = newLocation.coordinate;

    Pin *pin = [[Pin alloc] init];
    pin.coordinate = newLocation.coordinate;
    pin.title = @"您的位置";
    pin.subtitle = @"";

    pin.isCurrentLocationMark = YES;

    [mapView addAnnotation:pin];

    [pin release];
}
```

Modify our location update code to the above one.
We changed the MKPointAnnotation to Pin.
We set the extra isCurrentLocationMark flag to true.

# Custom annotation class

```
// we will not use custom image on current location.
// Otherwise, we use our custom image.
if (![(Pin*)annotation isCurrentLocationMark])
{
    pinView.image = [UIImage imageNamed:@"pin-arrow.png"];
}
```

When we are preparing the annotation view, we skip the current location one and let it use the default view.

# Custom annotation class