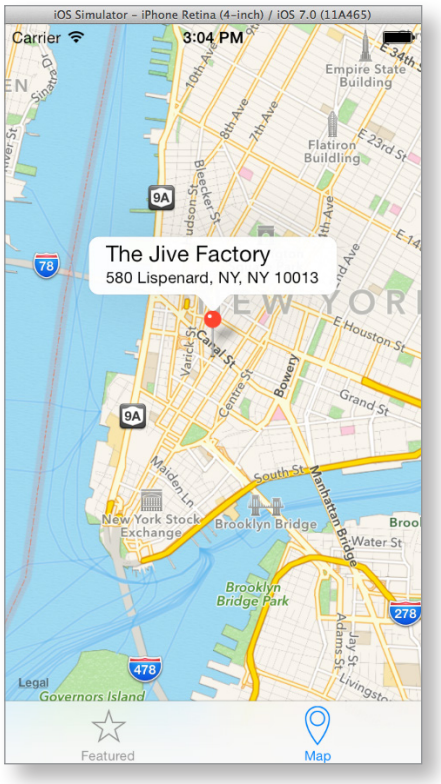
EXERCISE PREVIEW



EXERCISE OVERVIEW

In this exercise, we will add a map to our app showing the location of the Jive Factory.

1. We suggest you complete the previous exercises (2C–4C) before doing this one.

   If you completed the previous exercise, **Jive Factory.xcodeproj** should still be open. If you closed it, re-open as follows:
   – Go to **File > Open.**
   – Navigate to **Desktop > Class Files > yourname-iOS Intro Class > Jive Factory** and double–click on **Jive Factory.xcodeproj.**

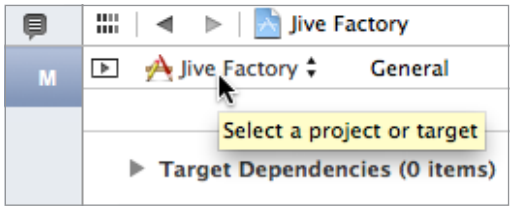
   If you **did not** complete the previous exercises and need a finished version of the file, you can open one that we've provided for you. To open it:
   – Go to **File > Open.**
   – Navigate to **Desktop > Class Files > yourname-iOS Intro Class > Jive Factory-Tab Bar Done** and double–click on **Jive Factory.xcodeproj.**

ADDING THE MAPKIT FRAMEWORK

We need to add an Apple-provided framework, called the **MapKit.framework,** to our project. It will give us additional functionality for manipulating the map such as zooming and pinpointing locations. A framework can be a collection of libraries or a collection of code, and it can contain new UI elements or images.

1. Click on the **Jive Factory** project name in the Project Navigator. This opens the **Project Settings Editor**.

2. Slightly to the right of the Project Navigator, to the left of General, in the menu select **Jive Factory** as a Target.



3. Click on the **Build Phases** tab at the top.
4. Expand the **Link Binary With Libraries** section. Here you'll see the frameworks that were already included when we created our project in Xcode.
5. Below the Frameworks list, click on the **Add items** button ( ).
6. In the search bar at the top, type **map**
7. In the results, double–click on **MapKit.framework** to add it.
8. In the Project Navigator, expand the **Frameworks** folder (it's near the bottom).
9. Notice Xcode automatically added **MapKit.framework** into this folder. In earlier versions of Xcode, you would have to do this manually. Progress!

ADDING A MAP VIEW

1. In the Project Navigator click on **Main.storyboard.**
2. In the **Object library** ( ) search type in **map**
3. From the **Object library** ( ), drag **Map View** onto the **View Controller - Map** in the Editor area, positioning it so it fills all the empty white space in the view.
4. Make sure the **Map View** is still selected in the Editor.

5. In the Utilities area, click on the **Attributes inspector** tab ( ).
6. Next to **Behavior,** check **Shows User Location.**
7. Let's test this out. Click the **Run** button ( ).
8. When the iOS Simulator finishes loading, click on the **Map** tab.
9. If you get an alert asking to use your current location, click **OK.**

   Because we are using the Simulator, it will not show our current location. On a phone, it will show the user's actual location.
10. Let's test out the zooming. To simulate zooming in the iOS Simulator, hold **Option.** Two gray circles (representing two fingertips in a pinching action) will appear. Click and drag the mouse to simulate pinching.

SETTING THE JIVE FACTORY LOCATION

What we actually want is to show the location of the Jive Factory (not the user's current location). For that we'll have to write some code. First we need to create a new class.

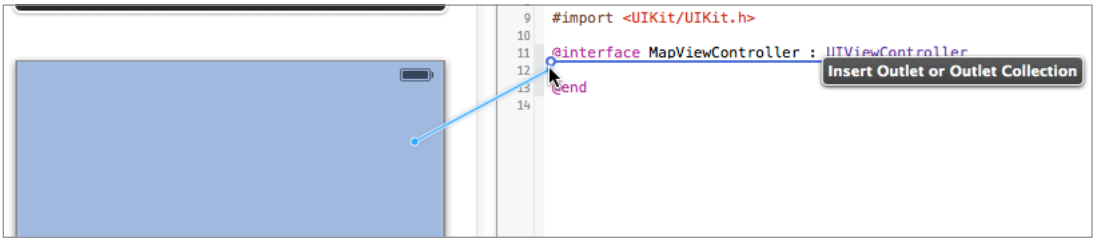
1. Switch back to Xcode.
2. In the Project Navigator select **BandDetail.m.** (We want the file to be added after this file, that's why we had you select it.)
3. Go to **File > New > File.**
4. Double–click **Objective-C class** to choose it.
5. From the **Subclass of** menu choose **UIViewController** (or start typing it and let Xcode autocomplete it for you).
6. Edit the name of the **Class** to be **MapViewController**
7. Click **Next.**
8. You should already be in the **Jive Factory** folder, so click **Create.**
9. Notice **MapViewController.h** and **MapViewController.m** have been added in the Project Navigator. Now we have a class we can work with.
10. In the Project Navigator click on **Main.storyboard.**
11. Select the **View Controller - Map** by clicking on the **battery** icon ( ), giving it a blue outline.
12. In the Utilities area, click on the **Identity inspector** tab ( ).
13. Under Custom Class, next to **Class**, start typing **Map.** Xcode should autocomplete to **MapViewController** and you can hit **Return** to apply it. (If it doesn't autocomplete, just type it.)

14. If the Document Outline is open, click **Hide Document Outline** ( ).

15. At the top right of the window, click on the **Assistant editor** button ( ).

16. **MapViewController.h** should be shown on the right. If **MapViewController.h** isn't showing on the right side, click the menu bar at the top of the file and choose it.

17. As shown below, hold **Ctrl** and drag from the **MKMapView** in the **Map View Controller** into the code below @interface:



18. In the menu that pops up, set the following:
    Connection: **Outlet**
    Name: **myMapView**
    Type: **MKMapView**
    Storage: **Strong**

19. Click **Connect.**

20. Still in **MapViewController.h,** below the #import statement (around line 10) add the following bold code:

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

@interface MapViewController : UIViewController
@property (strong, nonatomic) IBOutlet MKMapView *myMapView;

@end
```

    NOTE: The syntax to import frameworks uses **less than** and **greater than brackets <>** and when you import classes you use **quotes " ".**

21. Switch back to the **Standard editor** ( ).

22. In the Project Navigator click on **MapViewController.m.**

23. Add the following bold code below the @implementation line (around line 16) to synthesize the property:

```
@implementation MapViewController
@synthesize myMapView;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
```

DEFINING LOCATION COORDINATES

Now we want to define the coordinates of the Jive Factory to determine where and how much zooming there will be. We need to define three values: **latitude, longitude** and **span** (how far to zoom into the map). We will define these values as constants in our file. **Constants** are similar to variables, except that once the value is set, it won't be changed later.

1. Go to **File > Open.**
2. Navigate to **Desktop > Class Files > yourname-iOS Intro Class > Code Snippets** and open **mapCoordinates.txt.**
3. Press **Cmd–A** to select all the code.
4. Press **Cmd–C** to copy it.
5. Close the file.
6. Find the **@implementation MapViewController** code (around line 15).
7. Paste the code above it, as shown below:

```
@interface MapViewController ()

@end

#define jiveLatitude 40.72004;
#define jiveLongitude -74.003912;
#define jiveSpan 0.05f;

@implementation MapViewController
@synthesize myMapView;
```

This code defines three constants: jiveLatitude, jiveLongitude and jiveSpan. To get the latitude and longitude coordinates of an address, you can use a website such as **itouchmap.com/latlong.html** or **latlong.net**

8. Now we're going to use classes provided for us in the MapKit.framework to specify our location. First we are going to use the MKCoordinateRegion class. An MKCoordinateRegion is made up of a center point and a span.
9. Find the **viewDidLoad** method (around line 31).
10. Add the following bold code at the end of the method to declare a region.

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

    MKCoordinateRegion myRegion;
}
```

This code created a **myRegion** object of the type **MKCoordinateRegion.**

11. We need to create a center point for our region. Add the following bold code:

```
    MKCoordinateRegion myRegion;

    CLLocationCoordinate2D center;
}
```

12. After that, add the following code to set the center's **latitude** and **longitude:**

```
    CLLocationCoordinate2D center;
    center.latitude = jiveLatitude;
    center.longitude = jiveLongitude;
}
```

13. Below that, add the following bold code to create a **span:**

```
    center.longitude = jiveLongitude;

    MKCoordinateSpan span;
}
```

14. After that add the following bold code to set the span's **latitude** and **longitude:**

```
    MKCoordinateSpan span;
    span.latitudeDelta = jiveSpan;
    span.longitudeDelta = jiveSpan;
}
```

15. Next we need to set the center and span values back onto the myRegion object. Add the following bold code:

```
    span.longitudeDelta = jiveSpan;

    myRegion.center = center;
    myRegion.span = span;
}
```

16. Lastly, we need to reference the map view and set the region on it to our region. Add the following bold code:

```
    myRegion.span = span;

    [myMapView setRegion:myRegion animated:YES];
}
```

17. Let's test this out. Click the **Run** button (▶).

NOTE: If you get a warning asking if you want to **Stop "Jive Factory",** click **Stop.**

18. After the iOS Simulator loads, click the **Map** tab. The map is centered over the Jive Factory's location in New York, but it's missing the location indicator/pin. That's because we still need to create an annotation.

19. Switch back to Xcode.

20. After the code we added to set the region to our region (around line 49), add the following bold code to create an **annotation:**

```
    [myMapView setRegion:myRegion animated:YES];

    MKPointAnnotation *myPoint = [[MKPointAnnotation alloc] init];
}
```

21. We need to set three values on this annotation (coordinate, title, subtitle). Add the following bold code:

```
    MKPointAnnotation *myPoint = [[MKPointAnnotation alloc] init];
    myPoint.coordinate = center;
    myPoint.title = @"The Jive Factory";
    myPoint.subtitle = @"580 Lispenard, NY, NY 10013";
}
```

22. Finally, we need to add this annotation to our map. Add the following bold code:

```
    myPoint.subtitle = @"580 Lispenard, NY, NY 10013";

    [myMapView addAnnotation:myPoint];
}
```

23. Click the **Run** button (▶).

    NOTE: If you get a warning asking if you want to **Stop "Jive Factory",** click **Stop.**

24. After the iOS Simulator loads, click the **Map** tab.

25. Check it out! We've got our location pinpointed! Click on the **red pin** and see that it also shows the title and subtitle we added.

CREATING A SEMI-TRANSPARENT STATUS BAR

1. In older versions of iOS, the status bar at the top of the screen was an opaque black. As of iOS 7, the status bar is completely transparent, which makes the top of our map look much too busy. To fix this, go back to Xcode.

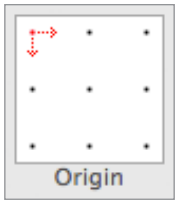
2. In the Project Navigator click on **Main.storyboard.**

3. In the **Object library** ( ) search type in **view**

4. From the **Object library** ( ), drag the **View** onto the **View Controller - Map.**

5. With the View selected, click on the **Size inspector** tab ( ).

6. If it is not already selected, click the **top left** Origin point, as shown to the right.



7. In the **Size inspector** set the following:

   X: **0**
   Y: **0**
   Width: **320**
   Height: **20**

8. Click on the **Attributes inspector** tab ( ).
9. In the menu next to **Alpha** replace any text with **0.9** and hit **Return.**
10. Click the **Run** button ( ).

    NOTE: If you get a warning asking if you want to **Stop "Jive Factory",** click **Stop.**
11. After the iOS Simulator loads, click the **Map** tab.
12. Notice the status bar looks almost (but not completely) white. To see the transparent effect, move the map around a bit. Our map looks less busy now, giving more focus to the content the users are interested in. Much better!
13. Return to Xcode.
14. Click the **Stop** button ( ).
15. Do a **File > Save.**
16. Keep this file open. In the next exercise, we'll link to an external website so that it shows up within our app.