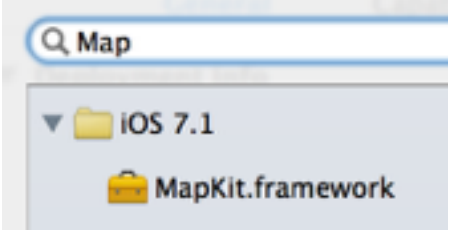
MapKit框架的使用

- MapKit框架使用前提
  - 导入框架



  - 导入主头文件

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

- MapKit框架使用须知
  - MapKit框架中所有数据类型的前缀都是MK
  - MapKit有一个比较重要的UI控件： MKMapView，专门用于地图显示

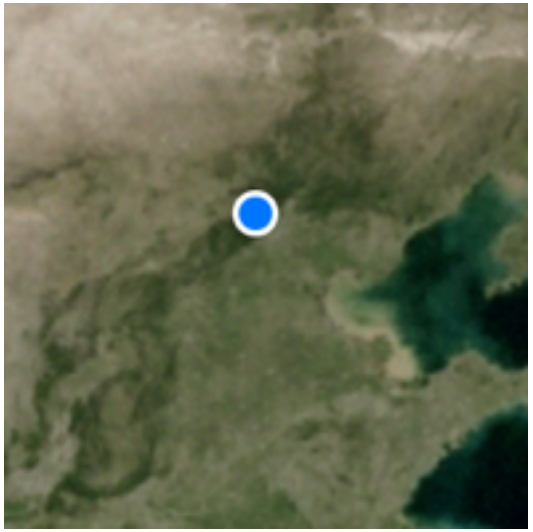
跟踪显示用户的位置

- 设置MKMapView的userTrackingMode属性可以跟踪显示用户的当前位置
  - MKUserTrackingModeNone ： 不跟踪用户的位置
  - MKUserTrackingModeFollow ： 跟踪并在地图上显示用户的当前位置
  - MKUserTrackingModeFollowWithHeading ： 跟踪并在地图上显示用户的当前位置，地图会跟随用户的前进方向进行旋转

- 下图是跟踪效果
  - 蓝色发光圆点就是用户的当前位置
  - 蓝色发光原点，专业术语叫做“大头针”

地图的类型

- 可以通过设置MKMapView的mapViewType设置地图类型
- MKMapTypeStandard： 普通地图（左图）
- MKMapTypeSatellite： 卫星云图 （中图）
- MKMapTypeHybrid： 普通地图覆盖于卫星云图之上（右图）



MKMapView的代理

- MKMapView可以设置一个代理对象，用来监听地图的相关行为

- 常见的代理方法有

p
```
- (void)mapView:(MKMapView *)mapView
didUpdateUserLocation:(MKUserLocation *)userLocation;
```

✓ 一个位置更改默认只会调用一次，不断监测用户的当前位置

✓ 每次调用，都会把用户的最新位置（userLocation参数）传进来

p
```
- (void)mapView:(MKMapView *)mapView
regionWillChangeAnimated:(BOOL)animated;
```

✓ 地图的显示区域即将发生改变的时候调用

p
```
- (void)mapView:(MKMapView *)mapView
regionDidChangeAnimated:(BOOL)animated;
```

✓ 地图的显示区域已经发生改变的时候调用

MKUserLocation

- MKUserLocation其实是个大头针模型，包括以下属性

p `@property (nonatomic, copy) NSString *title;`

✓ 显示在大头针上的标题

p `@property (nonatomic, copy) NSString *subtitle;`

✓ 显示在大头针上的子标题

p `@property (readonly, nonatomic) CLLocation *location;`

✓ 地理位置信息(大头针钉在什么地方?)

CLPlacemark – 地标

CLLocation *location

NSDictionary *addressDictionary

```
typedef struct {
    CLLocationDegrees latitude;
    CLLocationDegrees longitude;
} CLLocationCoordinate2D;
```

经纬度

CLLocation - 位置

CLLocationCoordinate2D coordinate

CLLocationDistance altitude

double

KUserLocation – 大头针模型

CLLocation *location

NSString *title;

NSString *subtitle;

```
typedef struct {
    CLLocationDegrees latitudeDelta;
    CLLocationDegrees longitudeDelta;
} MKCoordinateSpan;
```

跨度

```
typedef struct {
    CLLocationCoordinate2D center;
    MKCoordinateSpan span;
} MKCoordinateRegion;
```

区域

设置地图的显示

- 通过MKMapView的下列方法，可以设置地图显示的位置和区域

p 设置地图的中心点位置

✓
```
@property (nonatomic) CLLocationCoordinate2D centerCoordinate;
```

✓
```
- (void)setCenterCoordinate:(CLLocationCoordinate2D)coordinate
animated:(BOOL)animated;
```

p 设置地图的显示区域

✓
```
@property (nonatomic) MKCoordinateRegion region;
```

✓
```
- (void)setRegion:(MKCoordinateRegion)region
animated:(BOOL)animated;
```

MKCoordinateRegion

- MKCoordinateRegion是一个用来表示区域的结构体，定义如下

```
typedef struct {
    CLLocationCoordinate2D center; // 区域的中心点位置
    MKCoordinateSpan span; // 区域的跨度
} MKCoordinateRegion;
```

- MKCoordinateSpan的定义

```
typedef struct {
    CLLocationDegrees latitudeDelta; // 纬度跨度
    CLLocationDegrees longitudeDelta; // 经度跨度
} MKCoordinateSpan;
```

大头针

- 什么是大头针
- 现实生活中的大头针（左图）

- 地图上的大头针（右图）

- 钉在某个具体位置，用来标识这个位置上有特定的事物（比如这个位置上有家餐馆）

大头针的基本操作

- 添加一个大头针

```
- (void)addAnnotation:(id <MKAnnotation>)annotation;
```

- 添加多个大头针

```
- (void)addAnnotations:(NSArray *)annotations;
```

- 移除一个大头针

```
- (void)removeAnnotation:(id <MKAnnotation>)annotation;
```

- 移除多个大头针

```
- (void)removeAnnotations:(NSArray *)annotations;
```

- (id <MKAnnotation>)annotation参数是什么东西?

大头针模型对象：用来封装大头针的数据，比如大头针的位置、标题、子标题等数据

大头针模型

- 新建一个大头针模型类

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

@interface MJTuangouAnnotation : NSObject <MKAnnotation>
/** 坐标位置 */
@property (nonatomic, assign) CLLocationCoordinate2D coordinate;
/** 标题 */
@property (nonatomic, copy) NSString *title;
/** 子标题 */
@property (nonatomic, copy) NSString *subtitle;
@end
```

添加大头针

```
MJTuangouAnnotation *anno = [[MJTuangouAnnotation alloc] init];
anno.title = @“AA";
anno.subtitle = @“BB";
anno.coordinate = CLLocationCoordinate2DMake(40, 116);
[self.mapView addAnnotation:anno];
```

自定义大头针

- 很多情况下，需要自定义大头针的显示样式，比如显示一张图片

自定义大头针

- 如何自定义大头针
- 设置MKMapView的代理
- 实现下面的代理方法，返回大头针控件

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView
viewForAnnotation:(id <MKAnnotation>)annotation;
```

- 根据传进来的(id <MKAnnotation>)annotation参数创建并返回对应的大头针控件

- 代理方法的使用注意
- 如果返回nil，显示出来的大头针就采取系统的默认样式
- 标识用户位置的蓝色发光圆点，它也是一个大头针，当显示这个大头针时，也会调用代理方法
- 因此，需要在代理方法中分清楚(id <MKAnnotation>)annotation参数代表自定义的大头针还是蓝色发光圆点

自定义大头针

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView
viewForAnnotation:(id<MKAnnotation>)annotation
{
    // 判断annotation的类型
    if (![annotation isKindOfClass:[MJTuangouAnnotation class]]) return nil;

    // 创建MKAnnotationView
    static NSString *ID = @"tuangou";
    MKAnnotationView *annoView = [mapView
dequeueReusableAnnotationViewWithIdentifier:ID];
    if (annoView == nil) {
        annoView = [[MKAnnotationView alloc] initWithAnnotation:annotation
reuseIdentifier:ID];
        annoView.canShowCallout = YES;
    }
```

自定义大头针

```
// 传递模型数据
annoView.annotation = annotation;
// 设置图片
MJTuangouAnnotation *tuangouAnnotation = annotation;
annoView.image = [UIImage imageNamed:tuangouAnnotation.icon];
return annoView;
}
```

MKAnnotationView

- 地图上的大头针控件是MKAnnotationView

- MKAnnotationView的属性

p `@property (nonatomic, strong) id <MKAnnotation> annotation;`

✓ 大头针模型

p `@property (nonatomic, strong) UIImage *image;`

✓ 显示的图片

MKAnnotationView

- p `@property (nonatomic) BOOL canShowCallout;`
- ✓ 是否显示标注

- p `@property (nonatomic) CGPoint calloutOffset;`
- ✓ 标注的偏移量

- p `@property (strong, nonatomic) UIView *rightCalloutAccessoryView;`
- ✓ 标注右边显示什么控件

- p `@property (strong, nonatomic) UIView *leftCalloutAccessoryView;`
- ✓ 标注左边显示什么控件

MKPinAnnotationView

- MKPinAnnotationView是MKAnnotationView的子类

- MKPinAnnotationView比MKAnnotationView多了2个属性

p `@property (nonatomic) MKPinAnnotationColor pinColor;`

✓ 大头针颜色

p `@property (nonatomic) BOOL animatesDrop;`

✓ 大头针第一次显示时是否从天而降