GeekBand 极客班

互联网人才 + 油站!

GeekBand 极客班

www.geekband.com

设计模式三

孔祥波

“We become what we behold. We shape our tools and then our tools shape us.”

–Marshall McLuhan 马素·麦克鲁汉

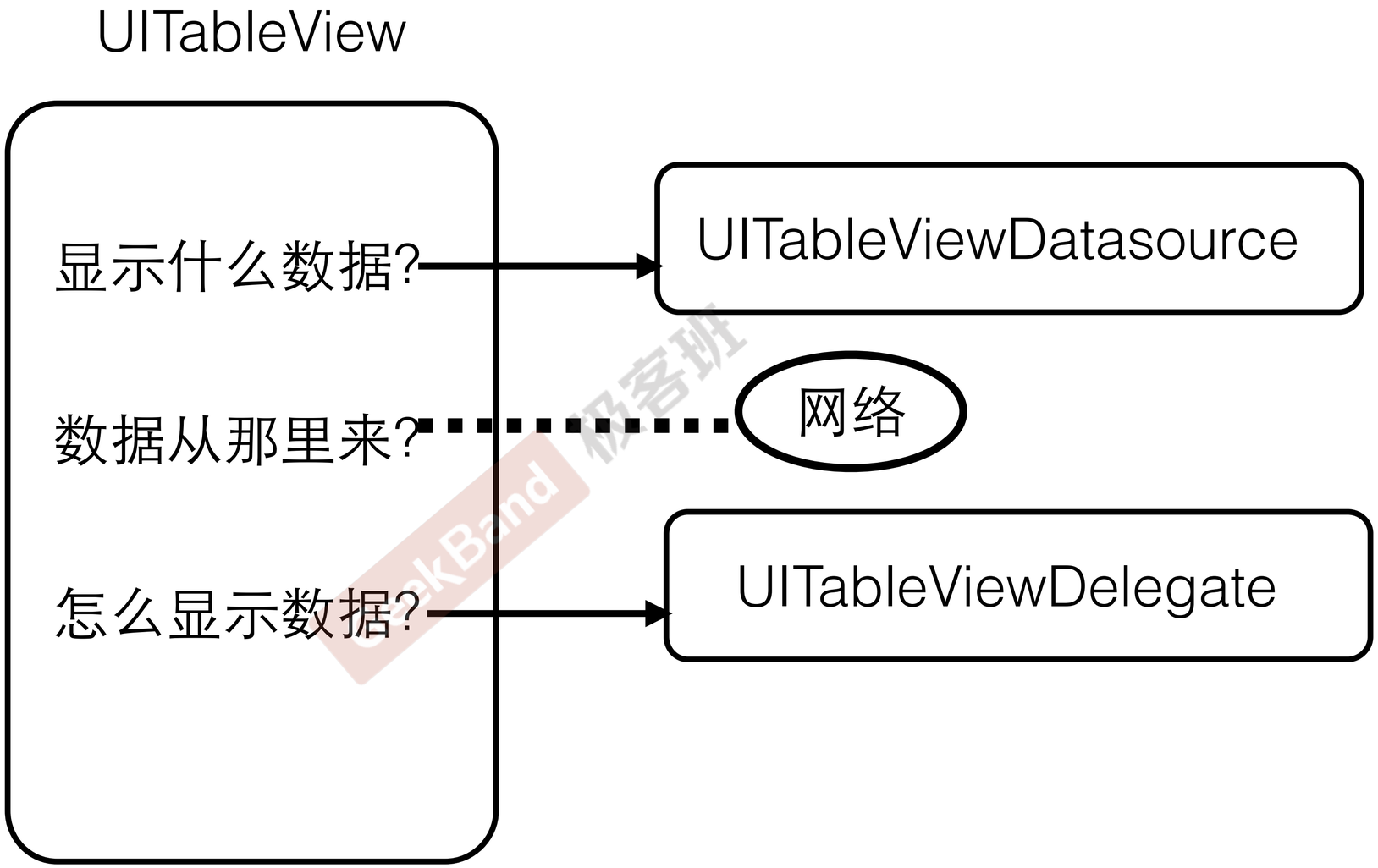
回顾

- 两步创建
- 模版方法
- 单例模式

委托模式delegate

delegate

- 复杂的模型,scrollView,tableView,collectionView
- 单一一个类无法表现复杂的设计
- 设计拆分
- 方便重用
- delegate 独立对象
- 清晰定义功能，变化行为/自定义界面
- 松散耦合，容易扩展


UITableView
显示什么数据?
数据从那里来?
怎么显示数据?
UITableViewDatasource
网络
UITableViewDelegate

Demo

UITableView define

```
NS_CLASS_AVAILABLE_IOS(2_0) @interface UITableView : UIScrollView <NSCoding>

- (instancetype)initWithFrame:(CGRect)frame style:(UITableViewStyle)style
    NS_DESIGNATED_INITIALIZER; // must specify style at creation. -initWithFrame: calls this
    with UITableViewStylePlain
- (instancetype)initWithCoder:(NSCoder *)aDecoder NS_DESIGNATED_INITIALIZER;

@property (nonatomic, readonly) UITableViewStyle style;
@property (nonatomic, weak, nullable) id <UITableViewDataSource> dataSource;
@property (nonatomic, weak, nullable) id <UITableViewDelegate> delegate;
```

```
// this protocol represents the data model object. as such, it supplies no information about
    appearance (including the cells)

@protocol UITableViewDataSource<NSObject>

@required

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

// Row display. Implementers should *always* try to reuse cells by setting each cell's
    reuseIdentifier and querying for available reusable cells with
    dequeueReusableCellWithIdentifier:
// Cell gets various attributes set automatically based on table (separators) and data
    source (accessory views, editing controls)

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *
    )indexPath;

@optional

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView                // Default
    is 1 if not implemented

- (nullable NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger
    )section;    // fixed font style. use custom view (UILabel) if you want something
    different
- (nullable NSString *)tableView:(UITableView *)tableView titleForFooterInSection:(NSInteger
    )section;

// Editing
```

```
// this represents the display and behaviour of the cells.

@protocol UITableViewDelegate<NSObject, UIScrollViewDelegate>

@optional

// Display customization

- (void)tableView:(UITableView *)tableView willDisplayCell:(UITableViewCell *)cell
    forRowAtIndexPath:(NSIndexPath *)indexPath;
- (void)tableView:(UITableView *)tableView willDisplayHeaderView:(UIView *)view forSection:
    (NSInteger)section NS_AVAILABLE_IOS(6_0);
- (void)tableView:(UITableView *)tableView willDisplayFooterView:(UIView *)view forSection:
    (NSInteger)section NS_AVAILABLE_IOS(6_0);
- (void)tableView:(UITableView *)tableView didEndDisplayingCell:(UITableViewCell *)cell
    forRowAtIndexPath:(NSIndexPath*)indexPath NS_AVAILABLE_IOS(6_0);
- (void)tableView:(UITableView *)tableView didEndDisplayingHeaderView:(UIView *)view
    forSection:(NSInteger)section NS_AVAILABLE_IOS(6_0);
- (void)tableView:(UITableView *)tableView didEndDisplayingFooterView:(UIView *)view
    forSection:(NSInteger)section NS_AVAILABLE_IOS(6_0);

// Variable height support

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)
    indexPath;
- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section;
- (CGFloat)tableView:(UITableView *)tableView heightForFooterInSection:(NSInteger)section;
```

自定义行为而无需创建子类

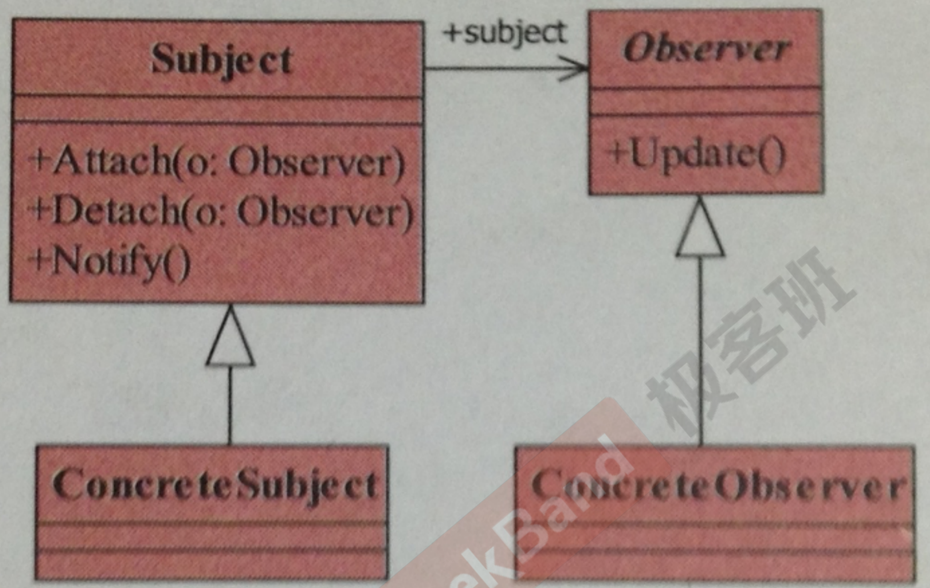
观察者

观察者

- 定义对象间一种一对多的依赖关系，使得每当一个对象改变状态，则所有依赖于他的对象都会得到通知并被自动更新。
- Subject被观察者 定义被观察者必须实现的职责，它必须能够动态的增加、取消观察者。它一般是抽象类或者是实现类，仅仅完成作为被观察者必须实现的职责：管理观察者并通知观察者
- Observer观察者 观察者接收到消息后，即进行update（更新方法）操作，对接收到的信息进行处理。
- 体的被观察者 定义被观察者自己的业务逻辑，同时定义对哪些事件进行通知。
- 具体的观察者 每个观察者在接收到消息后的处理反应应是不同的，各个观察者有自己的处理逻辑。

模式名称	观察者模式 Observer Pattern	类型	行为类

10



描述：定义对象间一种一对多的依赖关系，使得每当一个对象改变状态，则所有依赖于它的对象都会得到通知并被自动更新。

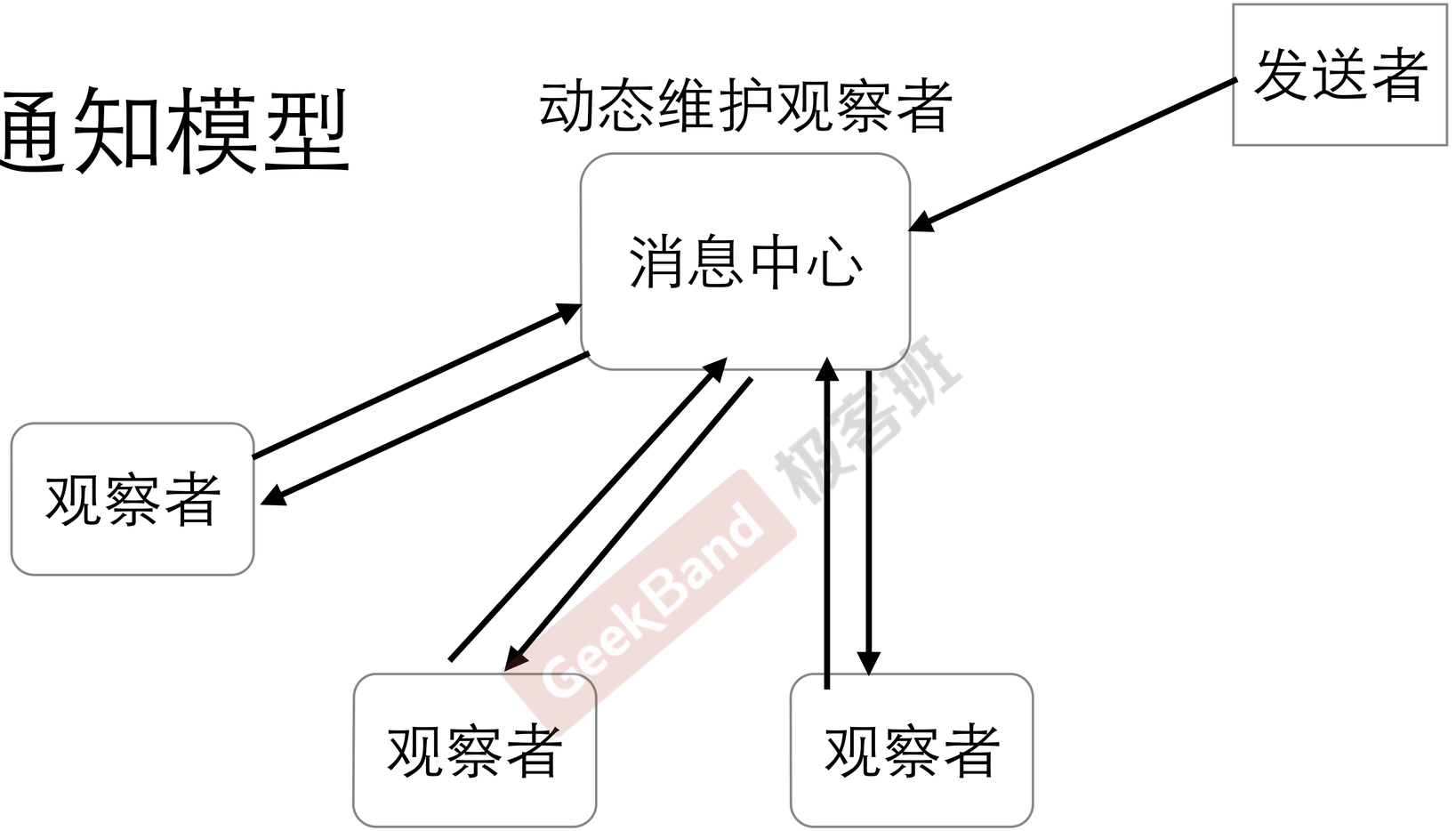
优缺点

- 优点：观察者和被观察者之间是抽象耦合，建立一套触法机制
- 缺点：观察者模式需要考虑开发效率和运行效率的问题

Cocoa的观察者模型

- 消息中心(NSNotificationCenter)
- Key-Value-Coding and Key-Value-Observing

消息通知模型



```
/*****************   Notifications   *****************/

@interface NSNotification : NSObject <NSCopying, NSCoding>

@property (readonly, copy) NSString *name;
@property (nullable, readonly, retain) id object;
@property (nullable, readonly, copy) NSDictionary *userInfo;

- (instancetype)initWithName:(NSString *)name object:(nullable id)object userInfo:(nullable
    NSDictionary *)userInfo NS_AVAILABLE(10_6, 4_0) NS_DESIGNATED_INITIALIZER;
- (nullable instancetype)initWithCoder:(NSCoder *)aDecoder NS_DESIGNATED_INITIALIZER;

@end

@interface NSNotification (NSNotificationCreation)

+ (instancetype)notificationWithName:(NSString *)aName object:(nullable id)anObject;
+ (instancetype)notificationWithName:(NSString *)aName object:(nullable id)anObject
    userInfo:(nullable NSDictionary *)aUserInfo;

- (instancetype)init /*NS_UNAVAILABLE*/;    /* do not invoke; not a valid initializer for
    this class */

@end
```

```
/*****************   Notification Center ****************/

@interface NSNotificationCenter : NSObject {
    @package
    void * __strong _impl;
    void * __strong _callback;
    void *_pad[11];
}

+ (NSNotificationCenter *)defaultCenter;

- (void)addObserver:(id)observer selector:(SEL)aSelector name:(nullable NSString *)aName
    object:(nullable id)anObject;

- (void)postNotification:(NSNotification *)notification;
- (void)postNotificationName:(NSString *)aName object:(nullable id)anObject;
- (void)postNotificationName:(NSString *)aName object:(nullable id)anObject userInfo:
    (nullable NSDictionary *)aUserInfo;

- (void)removeObserver:(id)observer;
- (void)removeObserver:(id)observer name:(nullable NSString *)aName object:(nullable id)
    anObject;

- (id <NSObject>)addObserverForName:(nullable NSString *)name object:(nullable id)obj queue:
    (nullable NSOperationQueue *)queue usingBlock:(void (^)(NSNotification *note))block
    NS_AVAILABLE(10_6, 4_0);
    // The return value is retained by the system, and should be held onto by the caller in
    // order to remove the observer with removeObserver: later, to stop observation.

@end
```

应用场景

- 窗口变化通知
- 系统键盘的出现和消失/位置大小变化
- UITextField 字符变化通知(可以用来限制输入长度)
- MPMoviePlayerController 播放器的行为变化(开始结束等事件)
- 自定义Class使用

Key-Value Coding (KVC)

- Access object values
  - `NSString *name = person.name;`
  - `NSString *name = [person name];`
  - `NSString *name = [person valueForKey:@“name”];`

- Set object values:
  - `[person setName:@“Pee-Wee Herman”];`
  - `person.name = @“Pee-Wee Herman”;`
  - `[person setValue:@“Pee-Wee Herman” forKey:@“name”];`

Key-Value Coding (KVC)

- Get/set a value on an object by key (a string)
- First attempts to access via KVC-Compliant getters/setters
- If that fails, attempts to get to value directly

Key Paths

- Traverse objects using dot-separated keys
- Ex: @”person.address.street”
- Must use “keyPath” methods, instead of “key” methods to automatically parse the string

```
- (id)valueForKeyPath:(NSString *)keyPath;
- (void)setValue:(id)value forKeyPath:(NSString *)keyPath;
```

Key-Value Observing (KVO)

- Listen for changes to an object's KVC-compliant values
- NSObject automatically broadcasts changes to observers
- No changes required to object being listened to

Key-Value Observing (KVO)

- To listen for changes:

- To stop listening for changes:

```
-(void)addObserver:(NSObject *)anObserver
        forKeyPath:(NSString *)keyPath
           options:(NSKeyValueObservingOptions)options
           context:(void *)context;

-(void)removeObserver:(NSObject *)anObserver
           forKeyPath:(NSString *)keyPath;
```

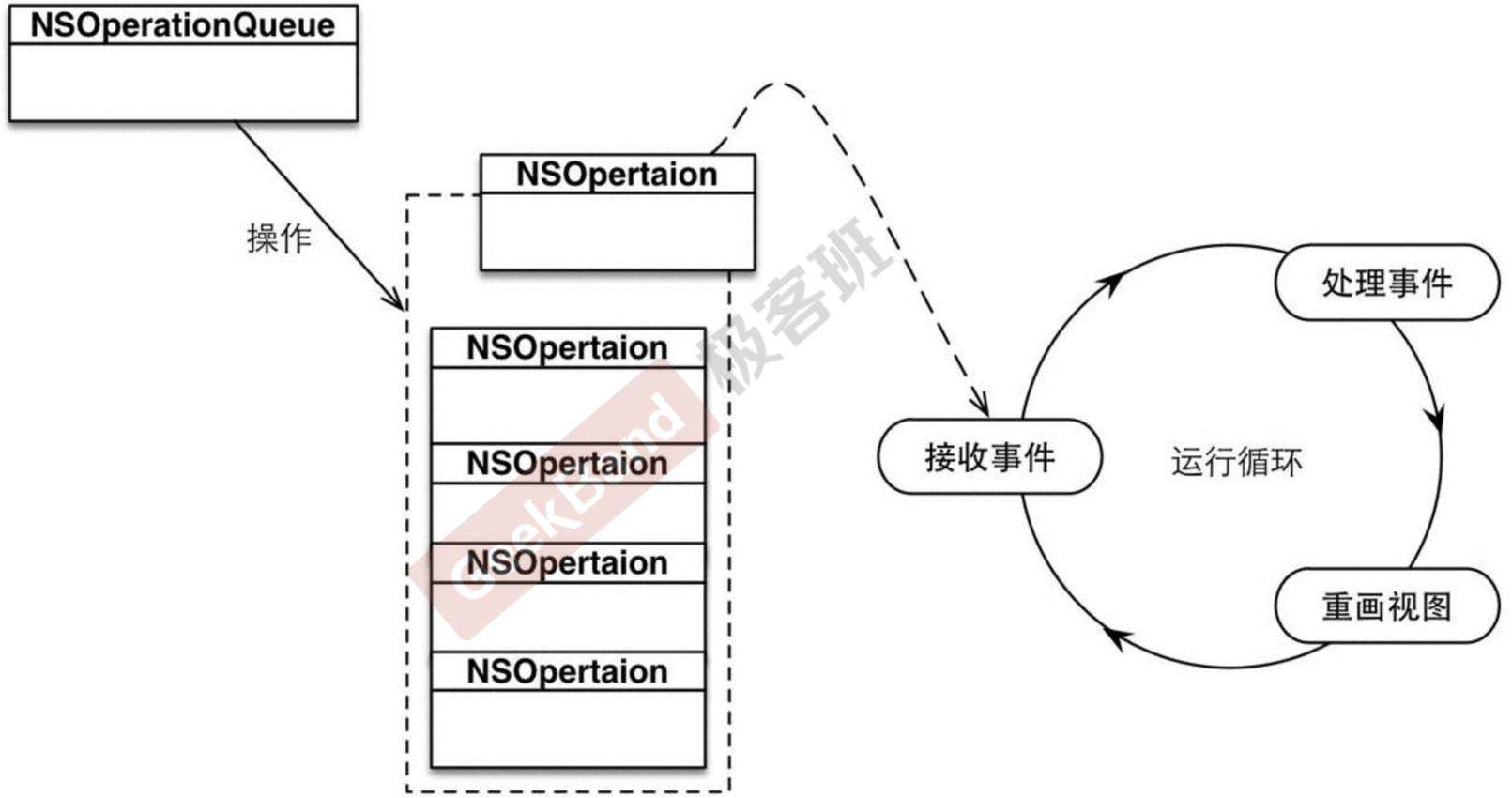
典型例子NSOperation and NSOperationQueue

```
@interface NSOperation : NSObject {
@private
    id _private;
    int32_t _private1;
#if __LP64__
    int32_t _private1b;
#endif
}

- (void)start;
- (void)main;

@property (readonly, getter=isCancelled) BOOL cancelled;
- (void)cancel;

@property (readonly, getter=isExecuting) BOOL executing;
@property (readonly, getter=isFinished) BOOL finished;
@property (readonly, getter=isConcurrent) BOOL concurrent; // To be deprecated; use and
    override 'asynchronous' below
@property (readonly, getter=isAsynchronous) BOOL asynchronous NS_AVAILABLE(10_8, 7_0);
@property (readonly, getter=isReady) BOOL ready;
```



引用来源 http://www.jianshu.com/p/cf7f7affb8b4

小结

- 委托模式delegate
- 观察者
- 消息通知
- KVC/KVO