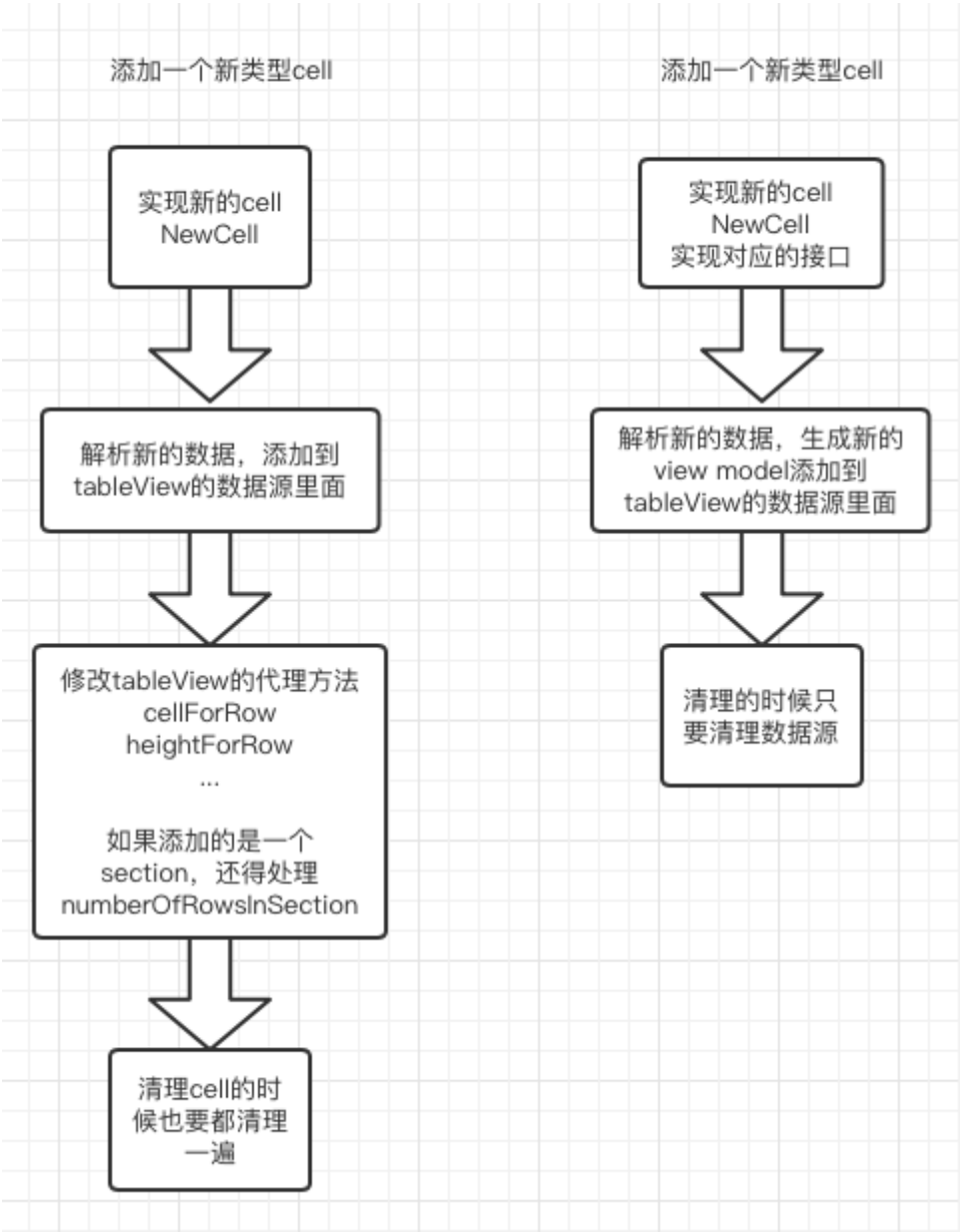

好处
* 适应经常变化的首页UI
* 用数据驱动UI
* 可重用性提升
* cell跟view controller解耦

几个点
* cell注册
    - 在vc中引入cell头文件注册。
    - 在cellForRow方法中注册，存在反复注册的情况，每次调cellForRow方法都会执行注册代码。
* cell高度返回
* 现有cell支持，cell添加category实现接口或直接cell实现接口，给cell添加一个view model。
* section header 的解决方式
    - 做成cell的形式
    - 在header的代理方法里面处理
* 分隔线的解决方法

协议说明

* YTCellProtocol
table view的cell都要实现这个接口，接口提供配置数据的方法和注册cell的方法。
```
/**
 cell 的接口
 提供一个配置cell的ViewModel的方法
 */
@protocol YTCellProtocol <NSObject>

@required


/**
 通过 view model 来配置cell, table view 的数据源里面装的都会是 view model

 @param viewModel: cell 对应的 view model
 */
- (void)configCellWithViewModel:(id)viewModel;


/**
 给table view 注册cell 方便复用，不用知道cell是用xib的形式，还是代码的形式实现的

 @param table 要注册的table view
 */
+ (void)registerFor:(UITableView *)table;


@end
```

* YTTableCellViewModelProtocol
table view cell 对应的view model 都要实现这个接口，实现返回cell高度的方法和cell的复用id
```
/**
 UITableViewCell 对应的ViewModel 的协议
 */
@protocol YTTableCellViewModelProtocol <NSObject>

@required;
/**
 返回cell的高度, 这个方法是在view model中实现，view model中有cell的全部数据，所以这里可以通过数据计算高度，或者直接返回固定高度

 @return cell的高度
 */
- (CGFloat)cellHeight;  // 返回cell的高度


/**
 返回cell的复用id

 @return cell的复用id
 */
+ (NSString *)identifier;

@end
```

Section类说明
```
@interface TableViewSection : NSObject
```

```
/**
 用来标识section的类型，要保证每个section的都不同
 代理方法里面如果要做特殊处理会用到，所以不能相同
 */
@property (nonatomic, copy) NSString *sectionKey;


/**
 section 里面 row 的 view model 集合
 */
@property (nonatomic, copy) NSArray *viewModels;


/**
 返回这个section有多少row，这个不用设置，在设置viewModels时，会自动设置
 */
@property (nonatomic, readonly) NSInteger numberOfRows;

@end
```

```
@implementation TableViewSection

- (instancetype)initWithSectionKey:(NSString *)key viewModels:(NSArray *)viewModels; {
    self = [super init];
    if (self) {
        _sectionKey = key;
        _viewModels = viewModels;
        _numberOfRows = viewModels.count;
    }

    return self;
}

@end
```