# UITableView

UITableView内置了两种样式：UITableViewStylePlain，UITableViewStyleGrouped

<UITableViewDataSource,UITableViewDelegate>里的方法：

tableView处理步骤

#pragma mark 1.有多少组

- (NSInteger)numberOfSectionsInTableView:(UITableView \*)tableView

#pragma mark 2.第section组头部控件有多高

- (CGFloat)tableView:(UITableView \*)tableView heightForHeaderInSection:(NSInteger)section

#pragma mark 3.第section组有多少行

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

#pragma mark 4.indexPath这行的cell有多高

- (CGFloat)tableView:(UITableView \*)tableView heightForRowAtIndexPath:(NSIndexPath \*)indexPath

#pragma mark 5.indexPath这行的cell长什么样子

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

#pragma mark 6.第section组头部显示什么控件

- (UIView \*)tableView:(UITableView \*)tableView viewForHeaderInSection:(NSInteger)section

- (NSInteger)numberOfSectionsInTableView:(UITableView \*)tableView

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

//每当有一个cell进入视野屏幕就会调用，所以在这个方法内部就需要优化。

- (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath{

if(cell==nil){

//在这里面做创建的工作。循环优化。防止刷新cell进入屏幕的时候重复的创建

}

}

//当调用reloadData的时候，会重新刷新调用数据源内所有方法,其他事情都不会做呀

[self reloadData]

//这个方法只有在一开始有多少条数据才会算多少个高度，这个方法只会调用一次，但是每次reloadData的时候也会调用

//而且会一次性算出所有cell的高度，比如有100条数据，一次性调用100次

- (CGFloat)tableView:(UITableView \*)tableView heightForRowAtIndexPath:(NSIndexPath \*)indexPath

-(NSArray \*)sectionIndexTitlesForTableView:(UITableView \*)tableView //右侧索引

-(void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath //行点击事件

NSIndexPath \*path = [self.tableView indexPathForSelectedRow]; //获得被选中的indexPath可以得到section，row

[self.tableView reloadRowsAtIndexPaths:[self.tableView indexPathsForSelectedRows] withRowAnimation:UITableViewRowAnimationNone]; //刷新table指定行的数据

[self.tableView reloadData]; //刷新table所有行的数据

UITableView常用属性：

UITableView \*tableView = [[UITableView alloc]initWithFrame:CGRectMake(0, 0, 320, 460) style:UITableViewStylePlain]; // 初始化表格

分隔线属性

tableView.separatorStyle = UITableViewCellSeparatorStyleSingleLine; //UITableViewCellSeparatorStyleNone;

[self.tableView setSeparatorStyle:UITableViewCellSeparatorStyleNone]; //取消分隔线

tableView.separatorColor = [UIColor lightGrayColor];

条目多选

tableView.allowsMultipleSelection = YES;

// 设置标题行高

[\_tableView setSectionHeaderHeight:kHeaderHeight];

[\_tableView setSectionFooterHeight:0];

// 设置表格行高

[\_tableView setRowHeight:50];

//设置背景色

self.tableView.backgroundView 优先级高，如果要设置backgroundColor的时候要先把view设置为nil

self.tableView.backgroundColor

//在tableView的头部或者尾部添加view,footerView宽度是不用设置的

xxxView.bounds = CGRectMake(0,0,0,height);

self.tableView.tableFooterView =xxxView;

self.tableView.tableHeaderView =xxxView;

UIButton \*bt = (UIButton\*)[self.contentView viewWithTag:i+100];

增加tableview滚动区域

self.tableView.contentInset = UIEdgeInsetsMake(0, 0, xx, 0);

### UITableViewCell

//创建UITableViewCell

UITableViewCell \*cell = [[UITableViewCell alloc]

initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:nil];

[cell.textLabel setBackgroundColor:[UIColor clearColor]];// 清空标签背景颜色

cell.backgroundView =xx; //设置背景图片

cell.backgroundVColor =xx;

cell.selectedBackgroundView = selectedBgView; //设置选中时的背景颜色

cell.accessoryView = xxxView; //设置右边视图

[cell setAccessoryType:UITableViewCellAccessoryNone]; //设置右侧箭头

[self setSelectionStyle:UITableViewCellSelectionStyleNone]; //选中样式

cell.selectionStyle = UITableViewCellSelectionStyleBlue;

//设置cell的高度

- (CGFloat)tableView:(UITableView \*)tableView heightForRowAtIndexPath:(NSIndexPath \*)indexPath

contentView下默认有3个子视图，其中的2个是UILabel，通过textLabel和detailTextLabel属性访问，第3个是UIImageView，通过imageView属性访问.

UITableViewCellStyleDefault, UITableViewCellStyleValue1, UITableViewCellStyleValue2, UITableViewCellStyleSubtitle

#pragma mark - 重新调整UITalbleViewCell中的控件布局

- (void)layoutSubviews{

[super layoutSubviews];

…

}

cell 里面还有一个contentView

### UITableViewCell表格优化

UITableViewCell对象的重用原理：

重用原理：当滚动列表时，部分UITableViewCell会移出窗口，UITableView会将窗口外的UITableViewCell放入一个对象池中，等待重用。当UITableView要求dataSource返回UITableViewCell时，dataSource会先查看这个对象池，如果池中有未使用的UITableViewCell，dataSource会用新的数据配置这个UITableViewCell，然后返回给UITableView，重新显示到窗口中，从而避免创建新对象

还有一个非常重要的问题：有时候需要自定义UITableViewCell（用一个子类继承UITableViewCell），而且每一行用的不一定是同一种UITableViewCell（如短信聊天布局），所以一个UITableView可能拥有不同类型的UITableViewCell，对象池中也会有很多不同类型的UITableViewCell，时可能会得到错误类型的UITableViewCell那么UITableView在重用UITableViewCell。解决方案：UITableViewCell有个NSString \*reuseIdentifier属性，可以在初始化UITableViewCell的时候传入一个特定的字符串标识来设置reuseIdentifier（一般用UITableViewCell的类名）。当UITableView要求dataSource返回UITableViewCell时，先通过一个字符串标识到对象池中查找对应类型的UITableViewCell对象，如果有，就重用，如果没有，就传入这个字符串标识来初始化一个UITableViewCell对象

/\*\*

单元格优化

1. 标示符统一，使用static的目的可以保证表格标示符永远只有一个

2. 首先在缓冲池中找名为"myCell"的单元格对象

3. 如果没有找到，实例化一个新的cell

\*\*/

- (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath{

static NSString \*cellIdentifier = @"myCell";

UITableViewCell \*cell = [tableView dequeueReusableCellWithIdentifier:cellIdentifier];

//使用这种方法不用判断下面的cell

UITableViewCell \*cell = [tableView dequeueReusableCellWithIdentifier:cellIdentifier forIndexPath:indexPath];

if (cell == nil) {

cell = [[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault reuseIdentifier:cellIdentifier];

}

return cell;

}

### 表格的编辑模式

删除、插入

- (void)setEditing:(BOOL)editing animated:(BOOL)animated; 开启表格编辑状态

- (UITableViewCellEditingStyle)tableView:(UITableView \*)tableView editingStyleForRowAtIndexPath:(NSIndexPath \*)indexPath{

返回表格编辑编辑样式。不实现默认都是删除

return editingStyle : UITableViewCellEditingStyleDelete, UITableViewCellEditingStyleInsert

}

- (void)tableView:(UITableView \*)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath \*)indexPath{

//根据editingStyle处理是删除还是添加操作

完成删除、插入操作刷新表格

- (void)insertRowsAtIndexPaths:(NSArray \*)indexPaths withRowAnimation:(UITableViewRowAnimation)animation;

-(void)deleteRowsAtIndexPaths:(NSArray \*)indexPaths withRowAnimation:(UITableViewRowAnimation)animation;

}

移动

- (void)tableView:(UITableView \*)tableView moveRowAtIndexPath:(NSIndexPath \*)sourceIndexPath toIndexPath:(NSIndexPath \*)destinationIndexPath

sourceIndexPath 移动的行

destinationIndexPath 目标的行

### 自定义表格行UITableViewCell

**storyboard方式创建：**

直接拖到UITableView里面设置UITableViewCell

注意：

1.通过XIB或者Storyboard自定义单元格时，在xib和Storyboard里面需要指定单元格的可重用标示符Identifier

2.注意表格的优化中的差别

在Storyboard中两者等效

xxCell \*cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

xxCell \*cell1 = [tableView dequeueReusableCellWithIdentifier:CellIdentifier forIndexPath:indexPath];

在xib文件中有差别：

第一种情况，只能在iOS 6以上使用，如果在viewDidLoad注册了nib文件，并且指定了“单元格”的可重用标示符，那么

dequeueReusableCellWithIdentifier：

dequeueReusableCellWithIdentifier:forIndexPath：

方法是等效的。如果在viewDidLoad中注册了nib文件，表格缓冲池中的管理，有系统接管！

第二种情况，是在iOS 4以上均可以使用，如果没有在viewDidLoad注册nib文件，那么，只能使用

dequeueReusableCellWithIdentifier:并且需要判断cell没有被实例化，并做相应的处理

在代码创建中差别：

用代码创建cell中的处理和nib一样，注册了cell就有系统接管并且可以用带forIndexPath的方法，没有注册就要自己去实例化cell，不能用带forIndexPath的方法

[tableView registerClass:XxxCell class] forCellReuseIdentifier:@"xxCell"];

**xib方式创建：**

//注册Identifier

- (void)viewDidLoad{

[super viewDidLoad];

/\*\*

注意：以下几句注册XIB的代码，一定要在viewDidLoad中！

注册XIB文件,获得根视图，并且转换成TableView,为tableView注册xib

Identifier名要在xib文件中定义，并且保持一致

\*\*/

UINib \*nib = [UINib nibWithNibName:@"BookCell" bundle:[NSBundle mainBundle]];

UITableView \*tableView = (UITableView \*)self.view;

[tableView registerNib:nib forCellReuseIdentifier:@"bookCell"];

}

//没有注册Identifier只能使用下面方法

static NSString \*CellIdentifier = @"bookCell";

BookCell \*cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

if (cell == nil) {

cell = [[BookCell alloc]initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier];

NSBundle \*bundle = [NSBundle mainBundle];

NSArray \*array = [bundle loadNibNamed:@"BookCell" owner:nil options:nil];

cell = [array lastObject];

}

**代码方式创建：**

1. 建立UITableViewCell的类，继承UITableViewCell
2. 往cell里面加入view的时候注意点：

//新建的组件放入contentView中

[self.contentView addSubview:xxView];

//设置图片拉伸属性stretch

UIImage \*normalImage = [UIImage imageNamed:@"xx.png"];

normalImage = [normalImage stretchableImageWithLeftCapWidth:

normalImage.size.width / 2 topCapHeight:normalImage.size.height / 2];

//在tableView里面viewDiDLoad里面要注册cell类

[tableView registerClass:XxxCell class] forCellReuseIdentifier:@"xxCell"];

### 自定义表格中Header

//自定义表格在这个方法中定义

-(UIView \*)tableView:(UITableView \*)tableView viewForHeaderInSection:(NSInteger)section

-(NSString \*)tableView:(UITableView \*)tableView titleForHeaderInSection:(NSInteger)section