UIKit基础6

果核科技
CORE TECH.

这节课我会学到什么？

了解UITabelView的基本结构

如何使用UITableView的各种属性、协议、方法

如何建立分段、索引、分组表格视图

表格视图UITableView

UITableView是 iOS Apps 中最常用的控件。很多应用程序在一定程度上都有使用表视图来显示数据列表。不仅可以用来显示文本数据,也可以呈现图像数据。

UITableView继承自UIScrollView,支持垂直滚动

UITableView的两种style

IOS7之前的两种style:

UITableView的两种style

IOS7以后的两种style(UITableViewStylePlain, UITableViewStyleGrouped):

UITableView的数据源和代理

- 数据源(dataSource)来提供UITableView的数据，向UITableView提供要展示的行数,以及要展示的样式等信息.
- 对象(delegate),以便在UITableView触发某些事件时做出相应的处理，比如选中了某一行。
- 一般会让视图控制器本身充当UITableView的dataSource和delegate,如:

在控制器内部执行:

```
tableView.delegate = self;
tableView.dataSource = self;
```

数据源(DataSouce)

黑核科技 CORE TECH.

1.设置数据源为当前控制器

tableView.dataSource = self;

2.执行UITableViewDataSource协议

```
@interface GHViewController ()<UITableViewDataSource>

@end
```

3.实现UITableViewDataSource的协议方法(只列举有代表性的几个)

// 要显示多少组数据

- (NSInteger)numberOfSectionsInTableView:

// 每组显示多少行数据

-(NSInteger)tableView: numberOfRowsInSection:

// 每行显示什么内容

-(UITableViewCell *)tableView:cellForRowAtIndexPath:

协议Delegate

1.设置代理为当前控制器

tableView.delegate = self;

2.执行UITableViewDelegate协议

```
@interface GHViewController ()<UITableViewDelegate>

@end
```

3.实现UITableViewDelegate的协议方法:(只列举有代表性的几个)

// 每行的高度

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:

// 点击(选中)每行时要执行什么操作

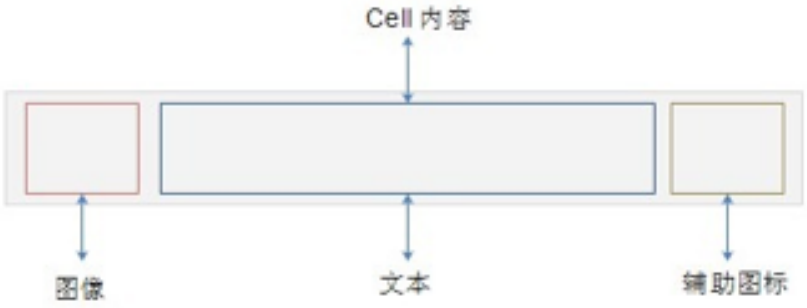
- (void)tableView:didSelectRowAtIndexPath:

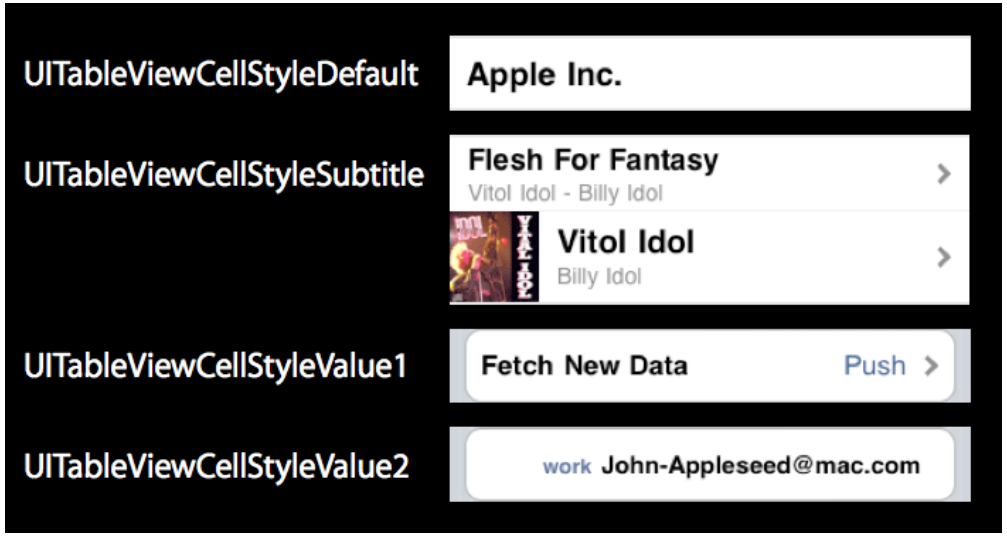
tableView的数据刷新

- ✓ reloadData // 全表刷新
- ✓ reloadRowsAtIndexPaths: withRowAnimation:// 刷新某一行或某几行
- ✓ reloadSections: withRowAnimation:// 刷新某个分组

tableView展示的数据需要进行改变的,在修改了数据以后,要调用tableView刷新数据的方法,这样控制器会重新执行tableView:numberOfRowsInSection:和tableView:cellForRowAtIndexPath:方法,界面显示的数据才会得以更新

单元格UITabelViewCell





UITableViewCellStyleDefault：预设使用这种，若左侧ImageView没图的话，只有一行字(textLabel.text)。

UITableViewCellStyleSubtitle：跟UITableViewCellStyleDefault大致相同，detailTextLable.text出现在textLable.text下方。

UITableViewCellStyleValue1：左侧为textLable.text并且左对齐，右侧为detailTextLabel.text并且右对齐。

UITableViewCellStyleValue2：左侧为detailTextLabel.text，右侧为textLabel.text并且左对齐。

cell.textLabel.text -- 主要的文字
cell.detailTextLabel.text – 补充的文字（灰色的文字或是蓝色的文字）
cell.imageView.image – 显示的图像

单元格UITabelViewCell配件样式

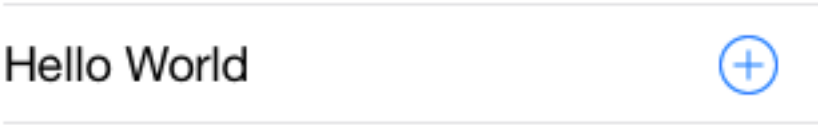
- UITableViewCellAccessoryDisclosureIndicator
- UITableViewCellAccessoryDetailDisclosureButton
- UITableViewCellAccessoryCheckmark
- UITableViewCellAccessoryDetailButton

```
// 直接属性设置
cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;
```

Cell的配件可以支持自定义,如:

```
cell.accessoryView = [UIButton buttonWithType:UIButtonTypeContactAdd];
```

显示在界面上为:



TableView的重用机制

黑核科技 CORE TECH.

```
-(UITableViewCell *)tableView:(UITableView *)atableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath
{
    static NSString *identifier = @"cell";
    UITableViewCell * cell = [tableView dequeueReusableCellWithIdentifier:identifier];
    if (!cell) {
        cell = [[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:identifier];
    }
    return cell;
}
```

当要展示的数据量比较大的时候,TableView并不是一次性创建所有要显示的行.比如你的表格数据有100行，但是屏幕上的空间只够显示10行，那么tableView只会创建10个左右的cell，当你滚动时，有些行会被移出屏幕，这些被移出屏幕的行会被回收放入它的回收空间，而将要出现的行会首先在回收空间查找是否有类似的(标识符相同)可以拿来用的cell，如果找到就直接使用，没找到才创建新的cell,从而避免创建很多不必要的开销。取回来的cell里的内容是旧的，你必需更新它的内容为将要出现的行的内容。

自定义UITableViewCell

果核科技 CORE TECH.

自定义UITableViewCell


```
    NSString *cellID = @"cell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:cellID];
    if(!cell){
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:cellID];
        // 需要添加在cell上的控件可以添加在这里，但苹果更推荐的方式是自定义Cell
        UILabel *label = [[UILabel alloc]initWithFrame:CGRectMake(20, 20, 100, 20)];
        label.tag = 100;
        [cell.contentView addSubview:label];    }

return cell;
```

自定义UITableViewCell

更好的方法是:新建一个UitableViewCell的子类,
在自定义方法里面添加一些需要的控件

```
@implementation TableViewCell

- (id)initWithStyle:(UITableViewCellStyle)style reuseIdentifier:(NSString
*)reuseIdentifier
{
    self = [super initWithStyle:style reuseIdentifier:reuseIdentifier];
    if (self) {
        _nameLabel = [[UILabel alloc]initWithFrame:CGRectMake(kXpos, 0, 100, 30)];
        _nameLabel.font = [UIFont systemFontOfSize:22];
        [self.contentView addSubview:_nameLabel];

        _ageLabel = [[UILabel alloc]initWithFrame:CGRectMake(kXpos, 60, 100, 20)];
        [self.contentView addSubview:_ageLabel];
        }
    return self;
}
```

表格控制器UITableViewController

- UITableViewController类继承自UIViewController类，极大地简化了创建UITableView的过程。
- UITableViewController负责处理表格布局，并使用一个UITableView实例对其进行填充。
- 可设置此控制器的窗体以支持任意导航栏或工具栏。可以通过tableView实例变量访问表格视图(.tableView)。

TableView的编辑模式

- TableView的editing(isEditing)属性可以获得tableView当前是否处于编辑模式

```
BOOL isEditModeStatus = self.tableView.editing;
```

- 也可以通过这个属性设置tableView的编辑状态:

```
[self.tableView setEditing:!isEditModeStatus animated:YES];
```

```
// 这个协议方法可以设置那一行可以被编辑,如果不实现此方法,则所有行都可以编辑
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath
*)indexPath
{
    if (indexPath.row == 0) {
        return NO;
    }
    else{
        return YES;
    }
}
```

删除单元格

很多用户已经知道滑动单元格，就会出现一个红色的删除按钮。

```
- (void)tableView:(UITableView *)tableView commitEditingStyle:
(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath
{
  //当不进行设置时候,默认的编辑状态就是删除状态
   if (editingStyle == UITableViewCellEditingStyleDelete) {
      // 1.先要修改展示的数据源
      [dataArray removeObjectAtIndex:indexPath.row];

      // 2.然后再执行界面刷新
      [tableView deleteRowsAtIndexPaths:@[indexPath]
                       withRowAnimation:UITableViewRowAnimationFade];
   }
}
```

添加单元格


```
// 可以通过这个协议方法返回需要的编辑状态
- (UITableViewCellEditingStyle)tableView:(UITableView *)tableView editingStyleForRowAtIndexPath:(NSIndexPath
*)indexPath
{
   return  UITableViewCellEditingStyleInsert;
}

// 同样是在这个方法里面处理编辑事件
-(void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
forRowAtIndexPath:(NSIndexPath *)indexPath
{
   if (editingStyle==UITableViewCellEditingStyleDelete) {
      ……
   }
   else if(editingStyle==UITableViewCellEditingStyleInsert)
   {

      i=i+1;
      NSInteger row = [indexPath row];
      NSArray *insertIndexPath = [NSArray arrayWithObjects:indexPath, nil];
      NSString *mes = [NSString stringWithFormat:@"添加的第%d行",i];
      //      添加单元行的设置的标题
      [self.listData insertObject:mes atIndex:row];
      [tableView insertRowsAtIndexPaths:insertIndexPath withRowAnimation:UITableViewRowAnimationRight];
   }
}
```

移动单元格


```
// 可设定哪些行支持单元格的移动,如不实现此方法,则所有单元格都可以点击
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (indexPath.row == 4) {
        return NO;
    }
    else{
        return YES;
    }
}
// 移动行的处理
-(void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)sourceIndexPath toIndexPath:
(NSIndexPath *)destinationIndexPath
{
    //   需要的移动行
        NSInteger fromRow = [sourceIndexPath row];
    //   获取移动某处的位置
        NSInteger toRow = [destinationIndexPath row];
    //   从数组中读取需要移动行的数据
    id object = [self.listData objectAtIndex:fromRow];
    //   在数组中移动需要移动的行的数据
    [self.listData removeObjectAtIndex:fromRow];
    //   把需要移动的单元格数据在数组中，移动到想要移动的数据前面
    [self.listData insertObject:object atIndex:toRow];
   //   刷新展示数据
     [tableView insertObject:object atIndex:toRow];
}
```

小结（1）

1) 初始化 UITableView对象

– initWithFrame:style: // 代码生成方式，如果你在nib里加的tableview不需要使用这个方法

2)配置TableView

– dequeueReusableCellWithIdentifier: // 必须要实现的方法，与TableView同生同死

style property // 有两种 UITableViewStylePlain, UITableViewStyleGrouped，经常用

– numberOfRowsInSection: //一个section有多少行，经常用

– numberOfSections //一个TableView有多少个section，经常用

rowHeight property // 行高，适用于所以cell高度一样的情况

separatorStyle property // cell之间的分割线样式

separatorColor property // cell分割线颜色

backgroundView property // tableview的背景view, 这个背景view在所有cell, header views, footer views之后

tableHeaderView property // tableview上方的一个headerView, 和delete里的section header不是一个概念

tableFooterView property // tableview下方的一个footerview

sectionHeaderHeight property // section Header的高度，

sectionFooterHeight property // sectjion Footer的高度

小结（2）

3） 访问Cells和Sections

– cellForRowAtIndexPath: //根据IndexPath返回cell

– indexPathForCell: //根据cell返回它的indexPath,和上面的方法互补

– indexPathForRowAtPoint://根据一个几何点返回indexPath,如果超过边界返回nil

– indexPathsForRowsInRect: //根据一个几何的矩形返回矩形所覆盖的行,返回是一个indexPath数组

– indexPathsForVisibleRows //同上

4) 滚动TableView

– scrollToRowAtIndexPath:atScrollPosition:animated: // 滚动到指定位置

– scrollToNearestSelectedRowAtScrollPosition:animated: // 同上

5) 管理sections

– indexPathForSelectedRow //返回选定行的indexPath,单行

– indexPathsForSelectedRows //返回选定行的indexPath数组，多行

– selectRowAtIndexPath:animated:scrollPosition: //根据indexPath选择一行

– deselectRowAtIndexPath:animated: //反选一行

allowsSelection property //是否允许用户选取一行

allowsMultipleSelection property // 是否选取多行，缺省为NO. 可以试试YES后的效果，哈哈

allowsSelectionDuringEditing property // 编辑模式时是否可选取一行

allowsMultipleSelectionDuringEditing property // 编辑模式时可否选取多行

小结（3）

6) 插入、删除、移动行和sections

– beginUpdates // 和endUpdates一起用，让插入、删除、选择操作同时动画，没用过
– endUpdates //
– insertRowsAtIndexPaths:withRowAnimation: //根据indexPath数组插入行
– deleteRowsAtIndexPaths:withRowAnimation: //根据indexPath数组删除行
– moveRowAtIndexPath:toIndexPath: //移动一行到另一行
– insertSections:withRowAnimation: //插入sections
– deleteSections:withRowAnimation: //删除sections
– moveSection:toSection: //移动section

7) 管理和编辑cell

editing property // YES进入编辑模式，tableview cell会出现插入、删除、重排序的控件
– setEditing:animated: //设置进入退出编辑模式

8) 重新加载TableView

– reloadData // 重建整个表，包括cells、header、footer，indexs
– reloadRowsAtIndexPaths:withRowAnimation: // 改进，不用reload整个表
– reloadSections:withRowAnimation: // 同上
– reloadSectionIndexTitles // 同上

小结（4）

黑核科技 CORE TECH.

9) 访问TableView的画图区

– rectForSection: // 返回指定section的矩形

– rectForRowAtIndexPath: //返回indexPath指定行的矩形

– rectForFooterInSection: // 返回section的footer矩形

– rectForHeaderInSection: // 返回section的header矩形

10) Registering Nib Objects for Cell Reuse

– registerNib:forCellReuseIdentifier: //

11) 管理委托和数据源 (重要)

dataSource property // 通常会这么用： myTableView.delegate = self; self 为viewController

delegate property // 通常会这么用： myTableView.dataSource = self; self 为viewController