**iOS开发UI篇—UITableview控件基本使用**

**一、一个简单的英雄展示程序**

NJHero**.**h文件代码（字典转模型）

[](javascript:void(0);)

1 #import <Foundation/Foundation.h>

2

3 @interface NJHero : NSObject

4 /\*\*

5 \* 头像

6 \*/

7 @property (nonatomic, copy) NSString \*icon;

8 /\*\*

9 \* 名称

10 \*/

11 @property (nonatomic, copy) NSString \*name;

12 /\*\*

13 \* 描述

14 \*/

15 @property (nonatomic, copy) NSString \*intro;

16

17 - (instancetype)initWithDict:(NSDictionary \*)dict;

18 + (instancetype)heroWithDict:(NSDictionary \*)dict;

19 @end

[](javascript:void(0);)

NJViewController.m文件代码

[](javascript:void(0);)

1 #import "NJViewController.h"

2 #import "NJHero.h"

3

4 @interface NJViewController ()<UITableViewDataSource, UITableViewDelegate>

5 /\*\*

6 \* 保存所有的英雄数据

7 \*/

8 @property (nonatomic, strong) NSArray \*heros;

9 @property (weak, nonatomic) IBOutlet UITableView \*tableView;

10

11 @end

12

13 @implementation NJViewController

14

15 #pragma mark - 懒加载

16 - (NSArray \*)heros

17 {

18 if (\_heros == nil) {

19 // 1.获得全路径

20 NSString \*fullPath = [[NSBundle mainBundle] pathForResource:@"heros" ofType:@"plist"];

21 // 2.更具全路径加载数据

22 NSArray \*dictArray = [NSArray arrayWithContentsOfFile:fullPath];

23 // 3.字典转模型

24 NSMutableArray \*models = [NSMutableArray arrayWithCapacity:dictArray.count];

25 for (NSDictionary \*dict in dictArray) {

26 NJHero \*hero = [NJHero heroWithDict:dict];

27 [models addObject:hero];

28 }

29 // 4.赋值数据

30 \_heros = [models copy];

31 }

32 // 4.返回数据

33 return \_heros;

34 }

35

36 - (void)viewDidLoad

37 {

38 [super viewDidLoad];

39 // 设置Cell的高度

40 // 当每一行的cell高度一致的时候使用属性设置cell的高度

41 self.tableView.rowHeight = 60;

42 self.tableView.delegate = self;

43 }

44

45 #pragma mark - UITableViewDataSource

46 // 返回多少组

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

48 {

49 return 1;

50 }

51 // 返回每一组有多少行

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

53 {

54 return self.heros.count;

55 }

56 // 返回哪一组的哪一行显示什么内容

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

58 {

59 // 1.创建CELL

60 UITableViewCell \*cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:nil];

61 // 2.设置数据

62 // 2.1取出对应行的模型

63 NJHero \*hero = self.heros[indexPath.row];

64 // 2.2赋值对应的数据

65 cell.textLabel.text = hero.name;

66 cell.detailTextLabel.text = hero.intro;

67 cell.imageView.image = [UIImage imageNamed:hero.icon];

68 // 3.返回cell

69 return cell;

70 }

71 #pragma mark - UITableViewDelegate

72 /\*

73 // 当每一行的cell的高度不一致的时候就使用代理方法设置cell的高度

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

75 {

76 if (1 == indexPath.row) {

77 return 180;

78 }

79 return 44;

80 }

81 \*/

82

83 #pragma mark - 控制状态栏是否显示

84 /\*\*

85 \* 返回YES代表隐藏状态栏, NO相反

86 \*/

87 - (BOOL)prefersStatusBarHidden

88 {

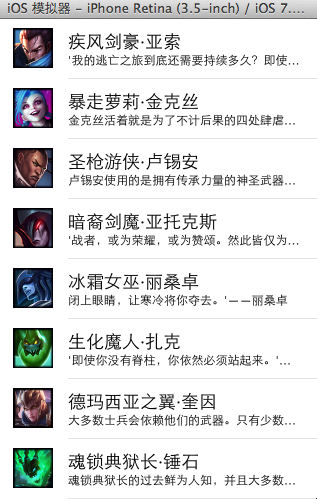
89 return YES;

90 }

91 @end

[](javascript:void(0);)

实现效果：



代码注意点：

（1）在字典转模型的代码处用下面的代码，为可变数组分配dictArray.count个存储空间，可以提高程序的性能

NSMutableArray \*models = [NSMutableArrayarrayWithCapacity:dictArray.count];

（2）设置cell的高度

有三种办法可以设置cell的高度

1） 可以在初始加载方法中设置，self.tableView.rowHeight = 60;这适用于当每一行的cell高度一致的时候，使用属性设置cell的高度。

2）在storyboard中设置，适用于高度一致

3）当每一行的cell的高度不一致的时候就使用代理方法设置cell的高度

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

{

    if (1 == indexPath.row) {

        return 180;

    }

    return 44;

}

二、cell的一些属性

代码示例：

[](javascript:void(0);)

1 #import "NJViewController.h"

2 #import "NJHero.h"

3

4 @interface NJViewController ()<UITableViewDataSource, UITableViewDelegate>

5 /\*\*

6 \* 保存所有的英雄数据

7 \*/

8 @property (nonatomic, strong) NSArray \*heros;

9 @property (weak, nonatomic) IBOutlet UITableView \*tableView;

10

11 @end

12

13 @implementation NJViewController

14

15 #pragma mark - 懒加载

16 - (NSArray \*)heros

17 {

18 if (\_heros == nil) {

19 // 1.获得全路径

20 NSString \*fullPath = [[NSBundle mainBundle] pathForResource:@"heros" ofType:@"plist"];

21 // 2.更具全路径加载数据

22 NSArray \*dictArray = [NSArray arrayWithContentsOfFile:fullPath];

23 // 3.字典转模型

24 NSMutableArray \*models = [NSMutableArray arrayWithCapacity:dictArray.count];

25 for (NSDictionary \*dict in dictArray) {

26 NJHero \*hero = [NJHero heroWithDict:dict];

27 [models addObject:hero];

28 }

29 // 4.赋值数据

30 \_heros = [models copy];

31 }

32 // 4.返回数据

33 return \_heros;

34 }

35

36 - (void)viewDidLoad

37 {

38 [super viewDidLoad];

39 // 设置Cell的高度

40 // 当每一行的cell高度一致的时候使用属性设置cell的高度

41 self.tableView.rowHeight = 60;

42 self.tableView.delegate = self;

43

44 }

45

46 #pragma mark - UITableViewDataSource

47 // 返回多少组

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

49 {

50 return 1;

51 }

52 // 返回每一组有多少行

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

54 {

55 return self.heros.count;

56 }

57 // 返回哪一组的哪一行显示什么内容

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

59 {

60 // 1.创建CELL

61 UITableViewCell \*cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:nil];

62 // 2.设置数据

63 // 2.1取出对应行的模型

64 NJHero \*hero = self.heros[indexPath.row];

65 // 2.2赋值对应的数据

66 cell.textLabel.text = hero.name;

67 cell.detailTextLabel.text = hero.intro;

68 cell.imageView.image = [UIImage imageNamed:hero.icon];

69

70 // 2.3设置cell的辅助视图

71 // cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;

72 if (0 == indexPath.row) {

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

74 }else

75 {

76 cell.accessoryView = [[UISwitch alloc] init];

77 }

78 // UIButton \*btn = [[UIButton alloc] init];

79 // btn.backgroundColor = [UIColor redColor];

80 // cell.accessoryView = btn;

81

82

83 // 2.4设置cell的背景颜色

84 cell.backgroundColor = [UIColor blueColor];

85

86 // 设置默认状态的背景

87 // UIView \*view = [[UIView alloc] init];

88 // view.backgroundColor = [UIColor blueColor];

89 // cell.backgroundView = view;

90

91 UIImageView \*iv = [[UIImageView alloc] initWithImage:[UIImage imageNamed:@"buttondelete"]];

92 cell.backgroundView = iv;

93

94 // 设置选中状态的背景

95 UIView \*view2 = [[UIView alloc] init];

96 view2.backgroundColor = [UIColor purpleColor];

97 cell.selectedBackgroundView = view2;

98 // 3.返回cell

99 return cell;

100 }

101

102

103 #pragma mark - 控制状态栏是否显示

104 /\*\*

105 \* 返回YES代表隐藏状态栏, NO相反

106 \*/

107 - (BOOL)prefersStatusBarHidden

108 {

109 return YES;

110 }

111 @end

[](javascript:void(0);)

实现效果：



cell的一些属性：

（1）设置cell的辅助视图，设置cell.accessoryView（系统提供了枚举型，也可以自定义@父类指针指向子类对象）；

（2）设置cell的背景颜色，有两种方式可以设置cell的背景颜色：

通过backgroundColor 和 backgroundView都可以设置cell的背景。但是backgroundView 的优先级比 backgroundColor的高，所以如果同时设置了backgroundColor和backgroundView, 那么backgroundView会盖住backgroundColor

    示例：cell.backgroundColor = [UIColorblueColor];

（3）设置cell默认状态的背景

  示例1：

　　    UIView \*view = [[UIView alloc] init];

   　　 view.backgroundColor = [UIColor blueColor];

    　　cell.backgroundView = view;

  示例2：

    UIImageView \*iv = [[UIImageViewalloc] initWithImage:[UIImageimageNamed:@"buttondelete"]];

    cell.backgroundView = iv;（父类指针指向子类对象，可以使用图片用简单的操作设置绚丽的效果）

（4）设置cell选中状态的背景

示例：

　　UIView \*view2 = [[UIView alloc] init];

    view2.backgroundColor = [UIColorpurpleColor];

    cell.selectedBackgroundView = view2;

三、tableview的一些属性

代码示例：

[](javascript:void(0);)

1 #import "NJViewController.h"

2

3 @interface NJViewController ()<UITableViewDataSource>

4

5 @end

6

7 @implementation NJViewController

8

9 - (void)viewDidLoad

10 {

11 [super viewDidLoad];

12

13 // 1.创建tableview

14 UITableView \*tableview = [[UITableView alloc] init];

15 tableview.frame = self.view.bounds;

16

17 // 2.设置数据源

18 tableview.dataSource =self;

19

20 // 3.添加tableview到view

21 [self.view addSubview:tableview];

22

23 // 4.设置分割线样式

24 // tableview.separatorStyle = UITableViewCellSeparatorStyleNone;

25

26 // 5.设置分割线颜色

27 接收的参数是颜色的比例值

28 tableview.separatorColor = [UIColor colorWithRed:0/255.0 green:255/255.0 blue:0/255.0 alpha:255/255.0];

29

30 // 设置tableview的头部视图

31 tableview.tableHeaderView = [UIButton buttonWithType:UIButtonTypeContactAdd];

32 tableview.tableFooterView = [[UISwitch alloc] init];

33 }

34

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

36 {

37 return 1;

38 }

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

40 {

41 return 10;

42 }

43

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

45 {

46 // 1.创建cell

47 UITableViewCell \*cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:nil];

48

49 // 2.设置cell的数据

50 cell.textLabel.text = [NSString stringWithFormat:@"%d", indexPath.row ];

51

52 // 3.返回cell

53 return cell;

54 }

55

56 - (BOOL)prefersStatusBarHidden

57 {

58 return YES;

59 }

60 @end

[](javascript:void(0);)

实现效果：



tableview的一些属性：

（1）设置分割样式（tableview.separatorStyle），这是个枚举类型

（2）设置分割线的颜色，可以直接使用系统给出的颜色，如果系统给定的颜色不能满足需求时，也可以自定义。

　　补充：颜色分为24位和32位的，如下

  24bit颜色

     R 8bit 0 ~ 255

     G 8bit 0 ~ 255

     B 8bit 0 ~ 255

     32bit颜色

     A 8bit 0 ~ 255（tou）

     R 8bit

     G 8bit

     B 8bit

     #ff ff ff 白色

     #00 00 00 黑色

     #ff 00 00 红色

     #255 00 00

设置为自定义颜色的实例:tableview.separatorColor = [UIColorcolorWithRed:0/255.0green:255/255.0blue:0/255.0alpha:255/255.0];

 //接收的参数是颜色的比例值

 (3)设置顶部和底部视图

tableview.tableHeaderView   //顶部

tableview.tableFooterView    //底部