A01-QQ聊天-底部的发送框

2014年10月16日 星期四　　0:23

1、演示项目和分析

2、创建项目加载外部文件

3、IB中拖拽控件，搭建底部的44个高度的底部发送框

A02-QQ聊天-加载模型

2014年10月17日 星期五　　16:30

1、定义枚举，标示自己发的消息，还是别人发的消息

```
typedef enum
{
    JSMessageTypeSelf = 0,
    JSMessageTypeOther = 1
} JSMessageType;

@interface JSMessage : NSObject
@property (nonatomic, copy) NSString *text;
@property (nonatomic, copy) NSString *time;
@property (nonatomic, assign) JSMessageType *type;
@end
```

A03-QQ聊天-自定义cell

2014年10月17日 星期五　　16:30

1、设置初始化的方法

```
+ (instancetype)cellWithTableView:(UITableView *)tableView
```

2、创建要显示的子控件

在initWithStyle中

```
//时间
        UILabel *timeLabel = [[UILabel alloc] init];
        [self addSubview:timeLabel];
        self.timeLabel = timeLabel;

        //头像
        UIImageView *iconView = [[UIImageView alloc]
init];
        [self addSubview:iconView];
        self.iconView = iconView;

        //消息
        UIButton *textView = [[UIButton alloc] init];
        [self addSubview:textView];
        self.textView = textView;
```

3、因为显示聊天信息的时候每一行的高度不一样，所以跟上个练习一样，创建frame模型

A04-QQ聊天-创建frame模型

2014年10月17日 星期五　　16:30

1、自定义JSMessageFrame 类

```
#import "JSMessage.h"
@interface JSMessageFrame : NSObject
@property (nonatomic, assign, readonly) CGRect iconF;
@property (nonatomic, assign, readonly) CGRect timeF;
@property (nonatomic, assign, readonly) CGRect textF;
@property (nonatomic, assign, readonly) CGFloat rowHeight;

@property (nonatomic, strong) JSMessage *message;

@end
```

2、修改自定义cell，添加JSMessageFrame 模型属性

```
@interface JSMessageCell : UITableViewCell
@property (nonatomic, strong) JSMessageFrame *messageFrame;
```

3、重写自定义cell中模型属性的setter方法，为子控件设置值

```
- (void)setMessageFrame:(JSMessageFrame *)messageFrame
{
    _messageFrame = messageFrame;

    JSMessage *message = messageFrame.message;

    self.timeView.text = message.time;
    self.timeView.frame = messageFrame.timeF;

    [self.textView setTitle:message.text forState:UIControlStateNormal];
    self.textView.frame = messageFrame.textF;

    NSString *icon = message.type == JSMessageTypeSelf ? @"me" : @"other";
    self.iconView.image = [UIImage imageNamed:icon];
    self.iconView.frame = messageFrame.iconF;
}
```

4、修改controller中的模型

```
@property (nonatomic, weak) NSMutableArray *messageFrames;
#pragma mark - tableView的代理方法
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)
indexPath
{
    JSMessageFrame *frame =  self.messageFrames[indexPath.row];
    return frame.rowHeight;
}
```

5、在messageFrame模型的set方法中计算子控件的frame,计算frame的同时，还要在自定义cell中设置时间居中和字体大小

按钮的文字为黑色，字体大小，不换行。

```
- (void)setMessage:(JSMessage *)message
{
    _message = message;
    //计算frame
    CGFloat margin = 10;
    //时间

    CGFloat timeX = 0;
    CGFloat timeY = 0;
    CGFloat timeW = 320;
    CGFloat timeH = 40;
```

```
    _timeF = CGRectMake(timeX, timeY, timeW, timeH);
    //头像
    CGFloat iconW = 50;
    CGFloat iconH = 50;
    CGFloat iconY = CGRectGetMaxY(self.timeF);
    CGFloat iconX;
    if (self.message.type == JSMessageTypeSelf) {
        iconX = 320 - iconW - margin;
    }else{
        iconX = margin;
    }
    _iconF = CGRectMake(iconX, iconY, iconW, iconH);
    //消息
    CGSize textMaxSize = CGSizeMake(200, MAXFLOAT);
    CGSize textSize = [self sizeWithText:self.message.text font:[UIFont
systemFontOfSize:14] maxSize:textMaxSize];
    CGFloat textX = 0;
    CGFloat textY = iconY;
    if (self.message.type == JSMessageTypeSelf) {
        textX = iconX - textSize.width - margin;
    }else{
        textX = CGRectGetMaxX(self.iconF) + margin;
    }
    _textF = CGRectMake(textX, textY, textSize.width, textSize.height);

    CGFloat textMaxH = CGRectGetMaxY(self.textF);
    CGFloat iconMaxH = CGRectGetMaxY(self.iconF);
    _rowHeight = MAX(textMaxH, iconMaxH) + margin;

}


- (CGSize)sizeWithText:(NSString *)text font:(UIFont *)font maxSize:(CGSize)maxSize
{
    NSDictionary *atts = @{NSFontAttributeName:font};
    return [text boundingRectWithSize:maxSize
options:NSStringDrawingUsesLineFragmentOrigin attributes:atts context:nil].size;
}
```

A05-QQ聊天-显示聊天列表

2014年10月17日 星期五　　16:30

1、设置数据源

```
#pragma mark - tableView的数据源方法
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return self.messageFrames.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    JSMessageCell *cell = [JSMessageCell cellWithTableView:tableView];
    //设置模型
    cell.messageFrame = self.messageFrames[indexPath.row];
    return cell;
}
```

2、去除tableView的分割线，设置背景颜色

```
self.tableView.separatorStyle = UITableViewCellSeparatorStyleNone;
self.tableView.backgroundColor = [UIColor lightGrayColor];
```

3、设置tableView中的cell不允许选中

```
self.tableView.allowsSelection = NO;
```

4、设置cell的背景颜色

```
self.backgroundColor = [UIColor lightGrayColor];
```

5、隐藏状态栏

```
//隐藏状态栏
- (BOOL)prefersStatusBarHidden
{
    return YES;
}
```

A06-QQ聊天-去掉相同的时间

2014年10月17日 星期五　　16:30

1、在JSMessage中添加一个属性

```
//是否显示时间
@property (nonatomic, assign, getter = isHideTime) BOOL
hideTime;
```

2、在初始化message对象的时候判断时间时候和上一个消息的时间相等

```
JSMessage *lastMessage;
    for (NSDictionary *dic in dicArray) {
        JSMessage *message = [[JSMessage alloc]
initWithDic:dic];

        if ([message.time
isEqualToString:lastMessage.time]) {
            message.hideTime = YES;
        }

        [messages addObject:message];

        lastMessage = message;
    }
```

3、在设置frame的时候，判断是否要显示时间

```
if (!message.isHideTime) {
        _timeF = CGRectMake(timeX, timeY, timeW, timeH);
    }
```

A07-QQ聊天-设置聊天信息的背景

2014年10月17日 星期五　　16:30

1、因为背景图片的颜色部分和边缘有一定的间距，而按钮的文字大小和按钮大小是一样的。所以先让按钮的文字和边缘产生20的边距。在initWithStyle方法中

```
#define JSTextPadding 20
textView.contentEdgeInsets = UIEdgeInsetsMake(JSTextPadding, JSTextPadding, JSTextPadding, JSTextPadding);
```

分别设置按钮和按钮文字的背景颜色，可以看到这个效果

2、因为按钮的边距各增加了20，所以让改变按钮的大小，增加40

```
CGSize textMaxSize = CGSizeMake(150, MAXFLOAT);
    CGSize textSize = [self sizeWithText:self.message.text font:[UIFont systemFontOfSize:14] maxSize:textMaxSize];

    CGSize textButtonSize = CGSizeMake(textSize.width + JSTextPadding * 2, textSize.height + JSTextPadding * 2);
    CGFloat textX = 0;
    CGFloat textY = iconY;
    if (self.message.type == JSMessageTypeSelf) {
        textX = iconX - textButtonSize.width - margin;
    }else{
        textX = CGRectGetMaxX(self.iconF) + margin;
    }
    _textF = CGRectMake(textX, textY, textButtonSize.width, textButtonSize.height);
```

3、对按钮的大小设置完毕，设置按钮的图片，拉伸图片

3.1 通过文档分析拉伸图片的三个方法

```
- (UIImage *)resizableImageWithCapInsets:(UIEdgeInsets)capInsets NS_AVAILABLE_IOS(5_0); // create a resizable version of this image. the interior is tiled when drawn.
- (UIImage *)resizableImageWithCapInsets:(UIEdgeInsets)capInsets resizingMode:(UIImageResizingMode)resizingMode NS_AVAILABLE_IOS(6_0); // the interior is resized according to the resizingMode

// use resizableImageWithCapInsets: and capInsets.

- (UIImage *)stretchableImageWithLeftCapWidth:(NSInteger)leftCapWidth topCapHeight:(NSInteger)topCapHeight;
```

3.2 实现

```
- (UIImage *)resizeImage:(NSString *)imgName
{
    UIImage *img = [UIImage imageNamed:imgName];
    img = [img stretchableImageWithLeftCapWidth:img.size.width * 0.5 topCapHeight:img.size.height * 0.5];
    return img;
}

//拉伸图片

    if (message.type == JSMessageTypeSelf) {
        [self.textView setBackgroundImage:[self resizeImage:@"chat_send_nor"] forState:UIControlStateNormal];
        [self.textView setBackgroundImage:[self resizeImage:@"chat_send_press_pic"] forState:UIControlStateHighlighted];
    }else{
        [self.textView setBackgroundImage:[self resizeImage:@"chat_recive_nor"] forState:UIControlStateNormal];
        [self.textView setBackgroundImage:[self resizeImage:@"chat_recive_press_pic"] forState:UIControlStateHighlighted];
    }
```

A08-QQ聊天-封装常用代码

2014年10月17日 星期五　　16:30

1、把计算文字的大小的方法封装到一个分类中，以后可以重用

2、封装缩放图片的方法

A09-通知机制

2014年10月17日 星期五　　16:30

1、通知中心，通知中心有两种方法，一种是发布通知，一种是订阅通知，必须先订阅通知，再发布通知

```
NSNotificationCenter *center = [NSNotificationCenter
defaultCenter];
```

2、订阅通知

```
JSPerson *mj = [[JSPerson alloc] init];
        mj.name = @"lmj";

        JSPerson *nj = [[JSPerson alloc] init];
        nj.name = @"lnj";
        //必须现有订阅者
        //observer 通知的订阅者
        //aSelector 接收到通知后做的事情
        //aName  要接收的通知的名称，如果为nil 则接收所有
通知
        //anObject 发布者，接收谁发布的通知，如果为nil接收
所有人发布的通知
        [center addObserver:mj selector:
@selector(niuNaiComing:) name:@"mainiumaile" object:nil];
```

3、发布通知

```
//现有订阅者，再发布通知
        //发布通知  sanyuan 通知的发布者
        [center postNotificationName:@"mainiumaile"
object:sanyuan userInfo:@{@"tx":@"好牛奶在三元"}];
        [center postNotificationName:@"mainiumaile"
object:sanlu userInfo:@{@"tx":@"毒牛奶在三鹿"}];
```

4、在通知的接收者中处理通知的消息

```
    //noti.object  通知的发布者
    //noti.userInfo  发送者给接受者发送的信息
    //noti.name  通知的名称

- (void)niuNaiComing:(NSNotification *)noti
{
    //noti.name;  //收到的通知的名称
    JSCompany *com =  noti.object;  //通知的发布者
    NSLog(@"%@",com.name);
    //noti.userInfo;  //发布通知的时候发送的额外信息
userInfo:
    //NSLog(@"%@",noti);
    NSLog(@"%@",noti.userInfo[@"tx"]);
```

```
}
```

5、当订阅者销毁，也要取消订阅通知，否则可能会出现野指针错误；

```
- (void)dealloc
{
    //在arc中不能，也不用调用[super dealloc];
    //取消订阅
    /在监听者的dealloc方法中，
    //必须取消监听，否则，当通知再次出现，通知中心任然回向该监听者发送消息
    //因为对象已经释放，所以可能会导致崩溃

    [[NSNotificationCenter defaultCenter] removeObserver:self];

}
```

6、通知和代理的区别

1、相同点

代理和通知都能完成对象之间的通信（A对象告诉B对象发生了什么，A对象传递数据给B对象）

2、不同点

代理：1对1（1个对象，只能告诉另一个对象发生了什么）

通知：多对多（1个对象可以通知多个对象，1个对象可以订阅多个对象发布的通知）

A10-键盘通知

2014年10月17日 星期五　　16:30

```
    //键盘的通知
    //UIKeyboardWillShowNotification(键盘即将显示)
    //UIKeyboardDidShowNotification(键盘已经显示)
    //UIKeyboardWillHideNotification(键盘即将隐藏)
    //UIKeyboardDidHideNotification(键盘已经隐藏)
    //UIKeyboardWillChangeFrameNotification(键盘的位置尺寸即将发生改变)
    //UIKeyboardDidChangeFrameNotification(键盘的位置尺寸已经发生改变)
```

A11-QQ聊天-键盘位置改变View跟着变化

2014年10月17日 星期五　　16:30

1、订阅键盘的frame即将改变的通知

```
//监听键盘frame变化
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillChangeFrame:)
name:UIKeyboardWillChangeFrameNotification object:nil];
```

2、通知到达后执行的方法，通知到达改变view的位置

```
- (void)keyboardWillChangeFrame:(NSNotification *)noti
{
    //NSLog(@"%@",noti.userInfo);

    //键盘弹出/消失的时候。获取最终位置
    CGRect keyBoardFrame =
[noti.userInfo[@"UIKeyboardFrameEndUserInfoKey"] CGRectValue];
    //用键盘的最终的y值  - view自身的高度，就是要移动的距离
    CGFloat ct = keyBoardFrame.origin.y -
self.view.frame.size.height;
    //获取键盘的动画时间
    CGFloat duration =
[noti.userInfo[@"UIKeyboardAnimationDurationUserInfoKey"]
floatValue];
    [UIView animateWithDuration:duration animations:^{
        //平移view
        self.view.transform =
CGAffineTransformMakeTranslation(0, ct);
    }];
}
```

3、在销毁方法中取消通知

```
- (void)dealloc
{
    //取消订阅通知
    [[NSNotificationCenter defaultCenter]
removeObserver:self];
}
```

4、拖动tableView取消键盘

```
#pragma mark - scrollView的代理方法 -- UITableViewDelegate -》
UIScrollViewDelegate
- (void)scrollViewWillBeginDragging:(UIScrollView *)scrollView
{
    //退出键盘
    [self.view endEditing:YES];
}
```

5、点击cell 取消键盘

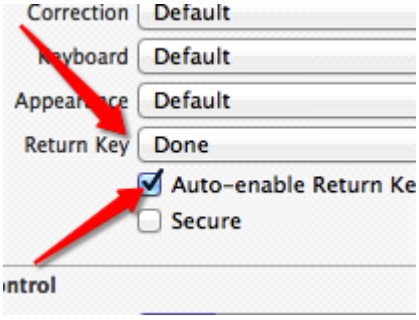
```
//点击cell，取消键盘
```

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)
event
{
    [self.superview.superview.superview endEditing:YES];
}
```

A12-QQ聊天-设置消息文本框

2014年10月17日 星期五　　16:30

1、设置键盘的确定按钮，设置文本框的如下属性



A13-QQ聊天-发送消息

2014年10月17日 星期五　　16:30

1、点键盘的确认按钮发送消息
2、设置UITextField的代理
　　2.1、代码设置

```
//设置文本框的代理
 self.messageView = self;
```

　　2.2、连线设置
3、写代理的方法

```
#pragma mark - 文本框的代理　　--点击键盘的确定按钮，执行此方法
- (BOOL)textFieldShouldReturn:(UITextField *)textField
```

4、发送消息就是往模型数组中放入一个消息对象
5、设置时间

```
//获取当前时间
    NSDate *date = [NSDate date];
    //格式化日期对象
    NSDateFormatter *formatter = [[NSDateFormatter alloc]
init];
    formatter.dateFormat = @"HH:mm";

    msg.time = [formatter stringFromDate:date];
```

6、判断当前消息和上一次消息的时间是否一样

```
//判断当前消息的时间和上一次消息的时间是否一样
    JSMessageFrame *lastMsgFrame = [self.messageFrames
lastObject];
    JSMessage *lastMsg = lastMsgFrame.message;
    if ([msg.time isEqualToString:lastMsg.time]) {
        msg.hideTime = YES;
    }
```

7、刷新表格
8、发完消息后。滚动到最后一行（新发的消息那一行）
9、封装成方法

```
- (void)sendMsg:(NSString *)text messageType:(JSMessageType)
type
{
    JSMessageFrame *frame = [[JSMessageFrame alloc] init];

    JSMessage *msg = [[JSMessage alloc] init];
    msg.type = type;
    msg.text = text;

    //获取当前时间
    NSDate *date = [NSDate date];
    //格式化日期对象
    NSDateFormatter *formatter = [[NSDateFormatter alloc]
init];
```

```
    formatter.dateFormat = @"HH:mm";

    msg.time = [formatter stringFromDate:date];


    //判断当前消息的时间和上一次消息的时间是否一样
    JSMessageFrame *lastMsgFrame = [self.messageFrames
lastObject];
    JSMessage *lastMsg = lastMsgFrame.message;
    if ([msg.time isEqualToString:lastMsg.time]) {
        msg.hideTime = YES;
    }


    frame.message = msg;

    [self.messageFrames addObject:frame];

    //刷新表格
    [self.tableView reloadData];

    //发完消息后滚动到最后一行
    NSIndexPath *path = [NSIndexPath
indexPathForRow:self.messageFrames.count - 1 inSection:0];
    [self.tableView scrollToRowAtIndexPath:path
atScrollPosition:UITableViewScrollPositionBottom
animated:YES];

}
```

X01-掌握

2014年10月17日 星期五　　16:29

X02-上课笔记

2014年11月1日 星期六　　15:48

1、画界面
2、设置controller中的步骤
    1、frame模型数组 属性
    2、懒加载 frame模型数组
    3、tableView的数据源方法 返回数据
    4、tableView的代理方法返回行高
3、写模型类。。。
4、写自定义cell类
    //思路
    1、frame模型属性
    2、类方法 返回cell对象 cell的可重用
    3、构造方法 初始化自定义cell的内部控件
    4、重写frame模型属性的setter方法，设置cell内部控件的内容和frame
5、写frame模型类，只定义属性
6、回到cell中完成跟frame相关的代码
7、回到controller中完成controller中的代码
8、继续frame模型类，重写setter方法，计算frame和行高