

iOS 平台多媒体短信系统开发的设计模式研究

徐 明, 陈广宇

(扬州大学 信息工程学院, 江苏 扬州 225127)

摘 要: 本文论述了在 Easytalk 移动应用项目“iOS 多媒体短信”系统的开发过程中, 对于几种常用设计模式的分析、研究, 以及这些模式在 iOS 平台的独特实现及其所具备的优势, 从“iOS 多媒体短信”系统的整个开发过程, 记录了通过引入设计模式思想合理解决具体问题的过程. 从面向接口编程角度引入设计模式思想, 从而降低代码的耦合性, 提高代码的可扩展性与健壮性. 文中进一步论述了该多媒体系统的实现过程. 目前该系统已通过测试, 并在 AppStore 上线.

关键词: 设计模式; iOS; 多媒体短信

中图分类号: TP399

文献标识码: A

文章编号: 1000-7180(2012)11-0112-04

On Design Patterns in iOS Multimedia Message System

XU Ming, CHEN Guang-yu

(College of Information Engineering, Yangzhou University, Yangzhou 225127, China)

Abstract: This paper discusses developing the iOS multimedia messaging module in Easytalk mobile system. Analyzes and researches some often used design patterns and the realized with unique advantages in iOS platform. In the entire development process of iOS multimedia messaging system this paper records reasonable solution which solves specific problems by importing the idea of design patterns. From the oriented interface programming importing the idea of design patterns will reduce the code coupling and increase the code scalability and robustness. This paper also takes a global view of realized the multimedia messaging system. This system has passed the test at present and Easytalk can be downloaded in Appstore also.

Key words: design patterns; iOS; multimedia message

1 引言

OS 是目前最流行的手机操作系统之一, 随着移动互联网的不断发展, 移动应用的开发将会迎来高峰; 设计模式是一套被反复使用、多数人知晓、经过分类编目的代码设计经验的总结, 在移动应用不断复杂化、全面化的今天, 规范的开发流程与程序的健壮性、可扩展性将是移动应用开发者需要关注的焦点之一. 将设计模式的思想引入到移动开发领域并加以调整与发展, 增强移动应用开发的规范性, 提升移动开发代码质量.

本文建立在掌握常用设计模式思想的基础上, 基于 Easytalk 系统“iOS 多媒体短信”子系统的开

发, 研究了常用设计模式在 iOS 平台的独特实现及其具备的优势, 包含了“iOS 多媒体短信”模块的整个开发过程, 记录了通过引入设计模式思想合理解决具体问题的过程^[1-2].

2 设计模式在 iOS 中的应用

设计模式 (Design pattern) 是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结. 使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性. 设计模式最早出现在建筑领域, 随后引入面向对象设计, 为进一步达到降低耦合的目的. 设计模式不仅仅指《设计模式》书中出现的 23 种模式, 而是一种在面向对象领域的设计

思想^[3-4]. 本多媒体短信系统实现主要采用如下一些设计模式.

2.1 MVC 模式

MVC 即 Model-View-Controller, 即把一个应用的输入、处理、输出流程按照 Model、View、Controller 的方式进行分离, 这样一个应用被分成三个层——模型层、视图层、控制层. 在 iOS 开发中, MVC 有其独特的实现.

2.2 单例模式

单例模式的实现方法是构造函数私有, 整个程序只有一个实例, 常为 static 型. 图 1 是单例模式的类图.

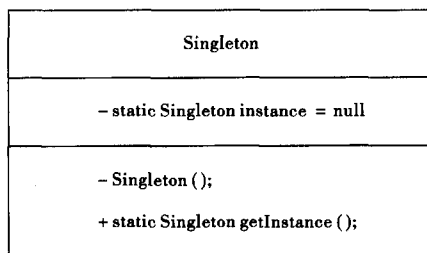


图 1 单例模式类图

在实际项目开发中, 单例模式可以解决很多问题, 例如在“iOS 多媒体短信”模块的开发中, composeNewSmsController 类是一个 Controller, 其中包含了整个短信列表以及短信发送方法, 多媒体短信的发送封装在专门的类中, 由于多媒体短信最终依旧会以普通短信的形式发送, 因此多媒体短信封装类也会调用 composeNewSmsController 类中的方法, 考虑到 composeNewSmsController 作为一个包含了整个短信列表的 Controller, 其对象可以在整个系统中唯一, 最终决定引入单例模式完成:

```

static composeNewSmsController * composeNewSmsControllerInstance; // singleton
+ (composeNewSmsController *) getComposeNewSmsControllerInstance {
    if (! composeNewSmsControllerInstance) { composeNewSmsControllerInstance =
        [[composeNewSmsController alloc] initWithNibName:
            @"composeNewSms" bundle:nil]; } return composeNewSmsControllerInstance; } + (void) releaseComposeNewSmsControllerInstance { [composeNewSmsControllerInstance release];
    composeNewSmsControllerInstance = nil; }
  
```

引入单例模式不仅完成了功能, 而且控制了 composeNewSmsController 对象的数量, 是合理而有效的. 另外, 单例模式不一定只允许单个实例, 其

思想是使类对象的数目可控, 构建对象池, 由对象池管理有限个对象依然属于单例模式范畴(多例模式).

2.3 代理模式

代理模式是最常用的设计模式, 功能极其强大, 利用组合可以简单地实现.

“iOS 多媒体短信”模块的开发大量使用代理模式, 以 UITableViewDelegate 为例:

```

tableView: heightForFooterInSection:
- (CGFloat) tableView:(UITableView *)tableView
    heightForFooterInSection:(NSInteger)section
  
```

此代理方法允许设置指定 Tableview 的指定 Section 的 Footer 高度. 这样就能根据内容动态地设置短信气泡的高度, 如图 2 所示.

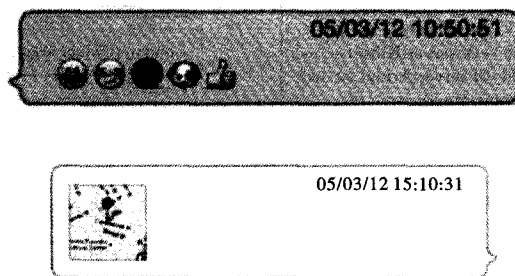


图 2 短信气泡动态设置

```

tableView: accessoryButtonTappedForRowWithIndexPath:
- (void) tableView:(UITableView *)tableView
    accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath
  
```

此代理方法响应用户对于指定行的触摸动作.

2.4 桥梁模式

桥梁模式主要的作用是“抽象与实现相分离”, 当对象引用情况比较复杂时, 选择引用抽象, 从而降低耦合. 图 3 是桥梁模式的类图.

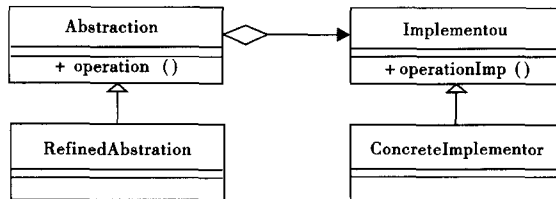


图 3 桥梁模式类图

在“iOS 多媒体短信”模块中, 不管是图片、视频、音频还是 GPS 信息, 略过服务器交互, 其本质上都是普通短信的收发, 如果不采用桥梁模式, 那么势必会在图片、视频、音频、GPS 信息和普通短信这些类之间存在大量的方法冗余, 这里采用桥梁模式, 普通短信类是 Abstraction, 其中包含所有短信收发方

法,图片、视频、音频、GPS 短信类是 Implementor,他们实现自己的逻辑,当发送图片、视频、音频、GPS 短信时,只需将他们的实例赋给普通短信类(Abstraction)的发送方法即可,符合迪米特法则、依赖倒置原则和开闭原则。

2.5 观察者模式

观察者模式应用十分广泛,功能十分强大.图4是观察者模式的类图。

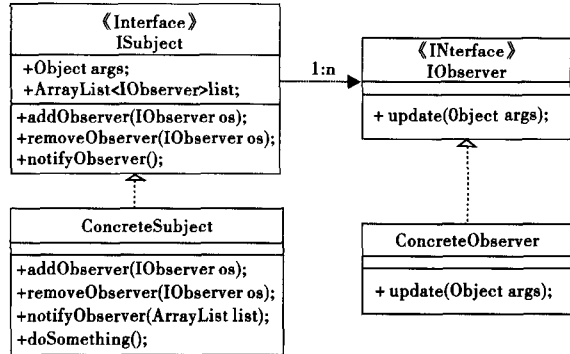


图4 观察者模式类图

在“iOS 多媒体短信”系统开发中,播放视频文件的同时需要对其进行简单控制,包括开始、暂停、上一个、下一个、自动关闭等,以自动关闭为例,为了能使视频播放器在播放完视频之后自动关闭,运用如下方法:

```
[[NSNotificationCenter defaultCenter]
addObserver:self
selector:@selector(movieFinishedCallback:)
name:MPMoviePlayerPlaybackDidFinishNotification
object:[moviePlayerViewController moviePlayer]];
-(void) movieFinishedCallback: (NSNotification *)
aNotification { [ saveVideoButton removeFromSuperview];
[moviePlayerViewController.view removeFromSuperview]; }
```

NSNotificationCenter 对象是一个单例,其中存放着所有观察者,关注同一 notification 的观察者分为一组,这里 self (即自身类对象,ComposeNewSmsCell 对象)作为被观察者,而播放器 moviePlayerViewController 则作为观察者,当 ComposeNewSmsCell 接收到 MPMoviePlayerPlaybackDidFinishNotification 消息时,通知观察者 moviePlayerViewController,用 removeFromSuperview 关闭播放器,完成通知。

在“iOS 多媒体短信”模块中,会设置很多信号量,这些信号量一旦发生变化,就会在界面产生相应

的反馈,为了达到这一目的,使用 KVO 机制:首先指定作为观察者和被观察的属性,用 @property 标记,然后 addObserver: forKeyPath: options: context: 语句将观察者注册到被观察者中,再执行观察者注册 observeValueForKeyPath: ofObject: change: context: 方法,该方法即观察者在接到通知后执行的具体逻辑.此时,当被观察者的值发生改变时,KVO 就能再次自动执行观察者方法 observeValueForKeyPath: ofObject: change: context:.

3 “iOS 多媒体短信”系统的开发

3.1 Easytalk 功能

Easytalk 是一款 iOS/Android 平台的通讯软件,包括实时通话、多媒体短信、视频会议等功能,“iOS 多媒体短信”子系统是 Easytalk 的重要功能之一,通过该系统,用户之间可以任意地发送图片、视频、声音以及地理位置信息,从而实现交流互动^[5-7].

3.2 “iOS 多媒体短信”系统功能

“iOS 多媒体短信”系统开发要考虑以下几个两个方面:功能实现和用户体验.实现的功能主要包括:客户端界面设计、视频采集与播放、图像采集与播放、音频采集与播放、地理位置获取与现实、文件编解码、播放控制、智能中断和多任务并发;用户体验要求软件能突出隐喻、方便用户使用.图5展示了该系统的功能。

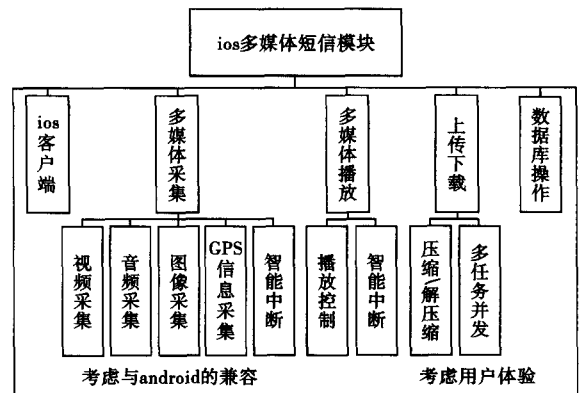


图5 “iOS 多媒体短信”系统功能

3.3 详细设计

3.3.1 多媒体短信发送流程

图6展示了多媒体短信的发送流程。

3.3.2 实现多任务并发

为了实现并发,使不同的 Cell 中都可同时接受或发送文件,作如下设计:

① 封装文件上传发送任务为 NSTask.

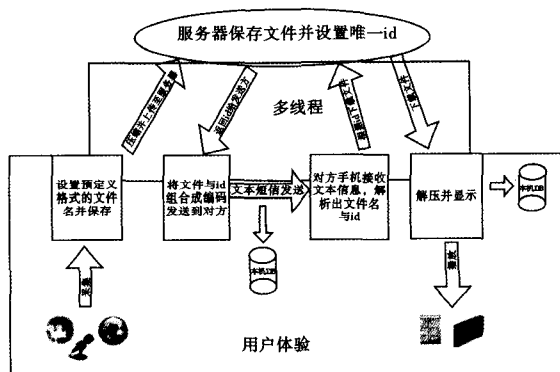


图6 多媒体短信发送流程

② 设置 `NSConnectionDelegate`, 实现具体的代理方法并指定。

③ 创建 `NSDictionary` 用于存放不同 `NSTask`。

④ 每当新建连接进行接受或发送时, 以 `connection` 为 `key` 将 `NSTask` 放入 `NSDictionary` 中。

⑤ 当主线程接收到 `NSConnectionDelegate` 的 `callback` 时, 根据 `callback` 的参数 `connection` 在 `NSDictionary` 中找到对应 `NSTask`, 执行重命名、解压、存储等操作后将此 `task` 移出 `dictionary`。

3.3.3 子线程刷新 VIEW

在子线程对界面的修改不会实时反馈, 需要等到主线程刷新界面, 子线程的修改才会显示出来, 这样严重影响了用户体验, 为了实现子线程实时刷新 VIEW, 使用方法 `performSelectorOnMainThread:` `withObject:` `waitUntilDone:`;

短信发送线程需要不断地将进度传递给主线程修改进度条, 子线程的代理中实时记录发送的数据包占总数据量的比例, 并通过此方法传给主线程。

3.3.4 多媒体短信文件名格式约定

系统通过唯一文件名指定到具体文件, 文本短信发送的内容也是文件名, 文件名中包含一系列有意义的信息需要组合与拆分, 因此需要为文件名的格式进行合理设计。

图片保存格式: `TYPE FROM TO NAME. JPEG`

图片发送格式: `filename = TYPE FROM TO NAME. JPEG &objected=objid`

视频保存格式: `TYPE FROM TO NAME. MOV`

视频发送格式: `filename = TYPE FROM TO NAME. MOV &objected=objid`

音频保存格式: `TYPE FROM TO NAME. ILBC`

音频发送格式: `filename = TYPE FROM TO NAME.`

`ILBC &objected=objid`

GPS 发送格式: `latitude=num & longitude=num`

4 结束语

经过在线实时测试, 采用本系统能够实现 iOS 设备间正常发送文本、视频、音频、图片及 GPS 信息, iOS 客户端的操作体验基本满足客户要求, 通过对用户实际使用调查发现大多数 iPhone 用户能够很快上手该软件, 多媒体短信模块的加入不影响原有 VoIP 电话的功能, 但 iOS 与 Android 之间的视频通讯由于编码问题暂不兼容。

由于系统中引入了设计模式思想, 从而很好地降低了代码的耦合, 程序的健壮性、可读性与可扩展性得到了很大提高。后一步研究工作将进一步引入设计模式: 装饰模式、状态模式, 尤其是状态模式的引入将提高多状态情况下气泡的效率。

参考文献:

- [1] Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides. Design patterns: elements of reusable object-oriented software[M]. Holland: Pearson Education, 2000.
- [2] Erica Sadun. The iPhone developers cookbook[M]. 2nd ed. USA: Addison-Wesley, 2010.
- [3] 乔纳森. AppStore 掘金: iPhone SDK 应用程序开发[M]. 北京: 机械工业出版社, 2010.
- [4] 王宇新, 田佳. 应用模糊方法的设计模式挖掘策略研究[J]. 计算机工程与应用, 2010(2): 150-154.
- [5] 刘浩, 董秋鸿. 基于观察者模式的屏幕布局控件设计[J]. 电子科技, 2010(8): 110-112.
- [6] 徐山峰. 基于 SIP 协议的 VoIP 系统的 QoS 机制的研究[J]. 现代电子技术, 2010(9): 59-62.
- [7] 戴祖诚, 林海云, 王宇娇等. SIP 协议在移动学习中的应用研究[J]. 微处理机, 2010(2): 133-136.
- [8] Barbosa A, Goncalves J, Ribeiro A N, et al. Integration of SIP protocol in android media framework[C]// EUROCON-International Conference on Computer as a Tool (EUROCON), 2011 IEEE. Sweden, Stockholm, 2011.

作者简介:

徐明男, (1961—), 讲师。研究方向为分布式网络数据库系统、移动数据库应用技术。