第 5 章　提醒用户的操作

在用户对操作界面进行操作时，有一些地方需要引起用户的注意。这时，就要采用独特的界面才可以让用户注意到它。在 iPhone 中引起用户注意的方法主要有两种，一种是弹出警告视图，一种是动作表单。本章将主要讲解这两种引起用户注意的方法。

5.1　警 告 视 图

警告视图的功能就是将想要让用户引起注意的信息显示给用户。使用 UIAlertView 会向用户显示一个警告视图来提醒用户。本节将主要为大家讲解警告视图的创建、显示、警告视图的 4 种显示形式以及响应警告视图等相关方面的知识。

5.1.1　创建警告视图

在 Objects 窗口中，是没有警告视图 AlerView 的。所以，我们不能采用静态创建方式，必须采用动态创建方式来创建。语法形式如下：

```
UIAlertView *对象名=[[UIAlertView alloc] initWithTitle:字符串 message:字符串 delegate:委托的对象 cancelButtonTitle:字符串 otherButtonTitles:nil];
```

其中，initWithTitle:用来初始化并设置出现在警告视图顶端的标题；message:用来指定将出现在对话框内容区域的字符串；delegate:用来指定将充当提醒委托的对象（所谓委托，顾名思义就是委托别人办事，就是当一件事情发生后，自己不处理，让别人来处理。）；cancelButtonTitle:用来指定警告视图中默认按钮的标题；otherButtonTitles:用来在警告视图中添加额外的按钮。一般警告视图的形式如图 5.1 所示。

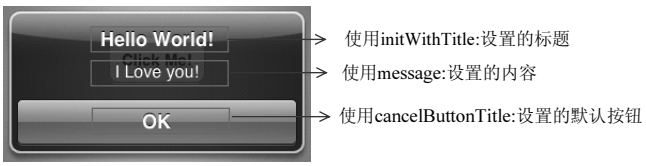

图 5.1　警告视图的形式

5.1.2　警告视图的显示

警告视图创建好以后，是不能直接显示在 iPhone Simulator 模拟器上的，还需要使用 show()方法才可以显示。使用 show()方法的语法形式如下：

```
[UIAlertView 对象名 show];
```

5.1.3　警告视图的 4 种显示形式

在 iPhone 中，警告视图是不会以一种固定形式进行显示的，而是以不同的形式进行显示。接下来我将为大家讲解在 iPhone 中最常用到的 4 种显示形式。

1. 一个按钮的警告视图

一个按钮的警告视图就是在图 5.1 中展示的警告视图，它也是最简单的警告视图。要实现它，方法其实很简单，只要将 otherButtonTitles:中的内容设置为 nil 就可以了。

【示例 5-1】 以下程序显示了只有一个按钮的警告视图。当单击 Click Me!按钮时，只有一个按钮的警告视图就会出现，它的标题为 Hello World!，信息为 I Love you!，默认的按钮为 OK。操作步骤如下：

（1）创建一个项目，命名为 5-1。

（2）单击打开 ViewController.xib 文件，将 Round Rect Button 视图拖放到用户设置界面，并将其标题改为 Click Me!，将该视图和 ViewController.h 文件进行动作 aa:的声明和关联。

（3）单击打开 ViewController.m 文件，编写程序代码，实现显示一个按钮的警告视图。程序代码如下：

```
01  #import "ViewController.h"
02  @interface ViewController ()
03  @end
04  @implementation ViewController
05  - (void)viewDidLoad
06  {
07      [super viewDidLoad];
08      // Do any additional setup after loading the view, typically from a nib.
09  }
10  - (void)didReceiveMemoryWarning
11  {
12      [super didReceiveMemoryWarning];
13      // Dispose of any resources that can be recreated.
14  }
15  - (IBAction)aa:(id)sender {
16      UIAlertView *a=[[UIAlertView alloc]initWithTitle:@"Hello World!"
        message:@"I Love you!"
17  delegate:nil cancelButtonTitle:@"OK" otherButtonTitles: nil];
                                                    //创建警告视图
18      [a show];                                   //显示警告视图
19  }
20  @end
```

运行结果如图 5.2 所示。

2. 多个按钮的警告视图

要实现多个按钮的警告视图 ，只需要在 otherButtonTitles:中添加一些字符串就可以了。

【示例 5-2】 以下程序显示了具有 3 个按钮的警告视图。单击 Click Me!按钮以后，就会弹出 3 个按钮的警告视图，它的标题为 Hello World!，信息为 I Love you!，默认的按钮为 OK。添加的两个按钮分别为 1 和 2。操作步骤如下：

（1）创建一个项目，命名为 5-2。

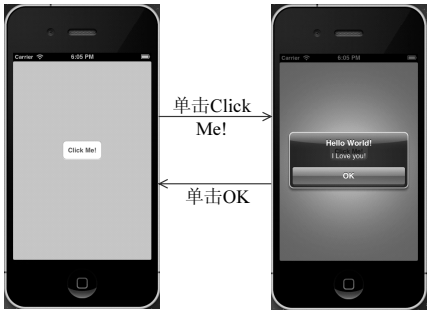

图 5.2　示例 5-1 运行结果

（2）单击打开 ViewController.xib 文件，将 Round Rect Button 视图拖放到用户设置界面，并将其标题改为 Click Me!，将该视图和 ViewController.h 文件进行动作 aa:的声明和关联。

（3）单击打开 ViewController.m 文件，编写程序代码，实现显示具有 3 个按钮的警告视图。程序代码如下：

```
01  #import "ViewController.h"
02  @interface ViewController ()
03  @end
04  @implementation ViewController
05  - (void)viewDidLoad
06  {
07      [super viewDidLoad];
08      // Do any additional setup after loading the view, typically from a nib.
09  }
10  - (void)didReceiveMemoryWarning
11  {
12      [super didReceiveMemoryWarning];
13      // Dispose of any resources that can be recreated.
14  }
15  - (IBAction)aa:(id)sender {
16      UIAlertView *a=[[UIAlertView alloc]initWithTitle:@"Hello World!"
        message:@"I Love you!"
17  delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:@"1",@"2",
    nil];                                                    //创建警告视图
18      [a show];                                            //显示警告视图
19  }
20  @end
```

运行结果如图 5.3 所示。

图 5.3　示例 5-2 运行结果

3. 无按钮

无按钮的警告视图在创建时，将 cancelButtonTitle:和 otherButtonTitles:都设置为 nil 就可以了。

【示例 5-3】 以下程序显示了一个无按钮的警告视图，当单击 Click Me!按钮时，就会弹出一个无按钮的警告视图，其中警告视图的标题为 Please wait。操作步骤如下：

（1）创建一个项目，命名为 5-3。

（2）单击打开 ViewController.xib 文件，将 Round Rect Button 视图拖放到用户设置界面，并将其标题改为 Click Me!，将该视图和 ViewController.h 文件进行动作 aa:的声明和关联。

（3）单击打开 ViewController.m 文件，编写程序代码，实现显示无按钮的警告视图。程序代码如下：

```
01  #import "ViewController.h"
02  @interface ViewController ()
03  @end
04  @implementation ViewController
05  - (void)viewDidLoad
06  {
07      [super viewDidLoad];
08      // Do any additional setup after loading the view, typically from a nib.
09  }
10  - (void)didReceiveMemoryWarning
11  {
12      [super didReceiveMemoryWarning];
13      // Dispose of any resources that can be recreated.
14  }
15  - (IBAction)aa:(id)sender {
16      UIAlertView *b=[[UIAlertView alloc]initWithTitle:@"Please wait"
        message:nil delegate:nil
17  cancelButtonTitle:nil otherButtonTitles:nil];      //创建警告视图
18      [b show];                                      //显示警告视图
19  }
20  @end
```

运行结果如图 5.4 所示。

图 5.4　示例 5-3 运行结果

4. 具有文本框的警告视图

在警告视图中，除了可以添加字符串和按钮以外，还可以添加一个文本框。要在警告

视图中添加文本框其实很简单，首先要创建一个警告视图，使用 Message:添加一行字符串，在这行字符串的位置上添加一个文本框，就将这行字符串覆盖了。再创建一个文本框，将其添加到警告视图中。

【示例 5-4】 以下程序显示了一个具有文本视图的警告视图。单击 Click Me!按钮以后，就会弹出一个具有文本视图的警告视图，其中警告视图的标题为 Please Enter Your Email Address!。操作步骤如下：

（1）创建一个项目，命名为 5-4。

（2）单击打开 ViewController.xib 文件，将 Round Rect Button 视图拖放到用户设置界面，并将其标题改为 Click Me!，将该视图和 ViewController.h 文件进行动作 aa:的声明和关联。

（3）单击打开 ViewController.h 文件，声明一个 UITextField 视图的变量，程序代码如下：

```
#import <UIKit/UIKit.h>
@interface ViewController : UIViewController{
    UITextField *b;      //声明一个 UITextField 视图的变量
}
- (IBAction)aa:(id)sender;
@end
```

（4）单击打开 ViewController.m 文件，编写程序代码，实现显示具有文本框的警告视图。程序代码如下：

```
#import "ViewController.h"
@interface ViewController ()
@end
@implementation ViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (IBAction)aa:(id)sender {
    UIAlertView *a=[[UIAlertView alloc]initWithTitle:@"Please Enter
    Your Email Address!"
message:@"aaa" delegate:nil cancelButtonTitle:@"OK" otherButton
Titles: nil];
    b=[[UITextField alloc]initWithFrame:CGRectMake
    (12.0,70.0,260.0,25.0)];
    [b setBackgroundColor:[UIColor whiteColor]]; //设置文本框视图的颜色
    [a addSubview:b];                          //将文本视图添加到警告视图中
    [a show];
}
@end
```

运行结果如图 5.5 所示。

在图 5.5 所示的运行结果中，文本框的位置和大小需要经过多次的测试来确定。

图 5.5　示例 5-4 运行结果

5.1.4　响应警告视图

在图 5.3 所示的运行结果中，虽然警告视图具有多个按钮，这几个按钮的功能就是执行一个操作——关闭警告视图。但是关闭警告视图的按钮就是使用 cancelButtonTitle:进行定义的，也就是 OK 按钮。对于 1 和 2 这两个按钮，它们是没有任何功能的。要想使警告视图得到充分的体现，我们要实现用户按钮的响应，要实现这一功能，必须要调用 ClickedButtonAtIndex()方法。

【示例 5-5】 以下程序显示了一个具有 3 个按钮的警告视图，当单击 Click Me!按钮，就会弹出具有 3 个按钮的警告视图，这 3 个按钮分别为 OK、1 和 2。当单击 OK 按钮时，警告视图就会退出；当单击按钮 1 时，就会出现另一个警告视图，这时此警告视图的标题为“你确定你的按键为“1”吗”；当单击按钮 2 时，就会出现另一个警告视图，这时此警告视图的标题为“你确定你的按键为“2”吗”。操作步骤如下：

（1）创建一个项目，命名为 5-5。

（2）单击打开 ViewController.xib 文件，将 Round Rect Button 视图拖放到用户设置界面，并将其标题改为 Click Me!，将该视图和 ViewController.h 文件进行动作 aa:的声明和关联。

（3）单击打开 ViewController.m 文件，编写程序代码，实现显示具有 3 个按钮的警告视图以及单击这些警告视图所作出的响应。程序代码如下：

```
01  #import "ViewController.h"
02  @interface ViewController ()
03  @end
04  @implementation ViewController
05  - (void)viewDidLoad
06  {
07      [super viewDidLoad];
08      // Do any additional setup after loading the view, typically from a nib.
09  }
10  - (void)didReceiveMemoryWarning
11  {
12      [super didReceiveMemoryWarning];
13      // Dispose of any resources that can be recreated.
14  }
15  - (IBAction)aa:(id)sender {
16      UIAlertView *a=[[UIAlertView alloc]initWithTitle:@"Hello World!"
```

```
        message:@"I Love China"
17  delegate:self cancelButtonTitle:@"OK" otherButtonTitles:@"1",
    @"2",nil];                                          //创建警告视图
18      [a show];
19  }
20  -(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
    (NSInteger)buttonIndex{
21      NSString *b=[alertView buttonTitleAtIndex:buttonIndex];
                                                        //设置被按下键的标题
22      //判断按下的按钮是哪一个
23      if([b isEqualToString:@"1"])
24      {
25          UIAlertView *c=[[UIAlertView alloc]initWithTitle:@"你确定你的
            按键为"1"吗" message:nil
26  delegate:nil cancelButtonTitle:@"YES" otherButtonTitles:nil];
27          [c show];
28      }
29     if([b isEqualToString:@"2"])
30      {
31          UIAlertView *d=[[UIAlertView alloc]initWithTitle:@"你确定你的
            按键为"2"吗" message:nil
32  delegate:nil cancelButtonTitle:@"YES" otherButtonTitles:nil];
33          [d show];
34      }
35  }
36  @end
```

在此程序中，我们将 delegate:设置为 self，方法 clickedButtonAtIndex()发生后，就让警告视图自己处理。运行结果如图 5.6 所示。

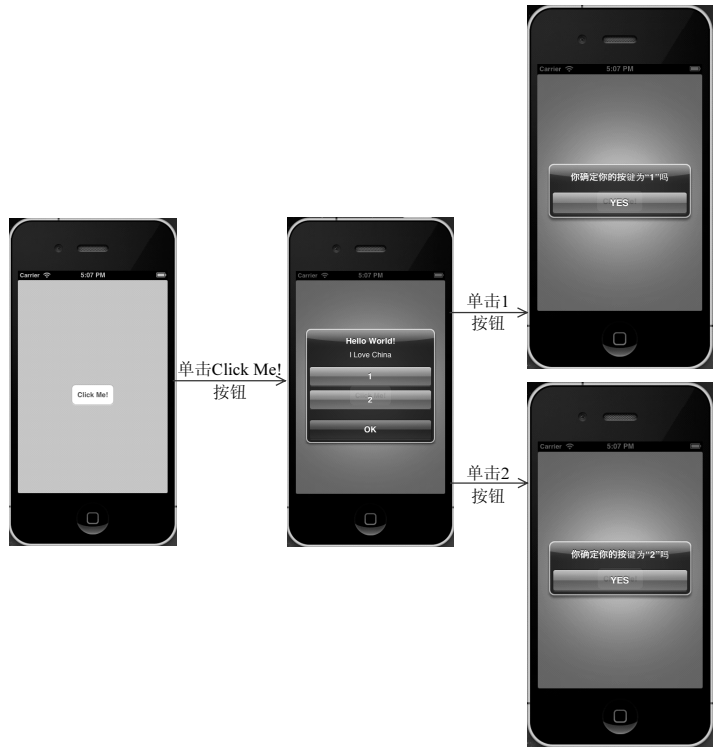

图 5.6　示例 5-5 运行结果

5.2　动 作 表 单

虽然警告视图可以用来显示多个按钮，但它最主要的功能还是引起用户的注意。如果想要在显示消息的时候，为用户提供多种选择，那么就要使用到动作表单。

5.2.1　动作表单的创建

因为在 Objects 窗口中是没有动作表单的，所以要使用代码创建。创建动作表单的语法形式如下：

```
UIActionSheet *对象名=[[UIActionSheet alloc]initWithTitle:字符串 delegate:
委托对象 cancelButtonTitle:字符串 destructiveButtonTitle:字符串
otherButtonTitles: nil];
```

其中，initWithTitle:用来初始化并设置出现在动作表单顶端的标题；delegate:用来指定将作为动作表单委托的对象；cancelButtonTitle:用来指定动作表单中默认按钮的标题；destructiveButtonTitle:用来指定将导致信息丢失的按钮标题，其中颜色为红色显示；otherButtonTitles:用来在动作表单中添加额外的按钮。一般动作表单的形式如图 5.7 所示。

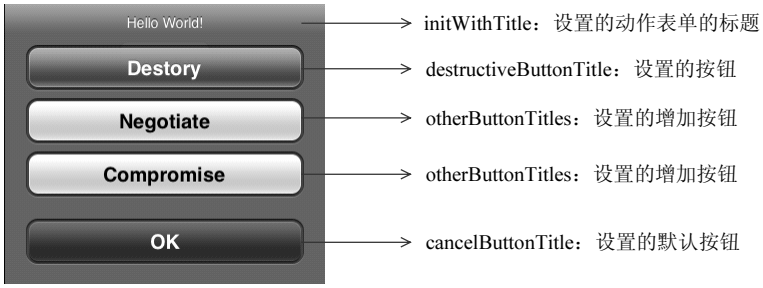

图 5.7　动作表单的形式

5.2.2　动作表单的显示

当创建好动作表单以后，动作表单和警告视图一样是不能显示的，需要使用一个实现显示的方法，即使用 showInView()方法。showInView()使用的语法形式如下：

```
[动作表单对象名 showInView:视图];
```

这里的视图是动作表单来源的视图。

【示例 5-6】 以下程序显示了一个动作表单。当单击 Click Me!按钮后，就会出现一个动作表单。其中动作表单的标题为 Hello World，默认按钮为 OK，导致信息丢失的按钮为 Destory，为动作表单新增的两个按钮分别为 Negotiate 和 Compromise。操作步骤如下：

（1）创建一个项目，命名为 5-6。

（2）单击打开 ViewController.xib 文件，将 Round Rect Button 视图拖放到用户设置界面，并将其标题改为 Click Me!，将该视图和 ViewController.h 文件进行动作 aa:的声明和关联。

（3）单击打开 ViewController.m 文件，编写程序代码，实现显示一个动作表单。程序

代码如下：

```
#import "ViewController.h"
@interface ViewController ()
@end
@implementation ViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (IBAction)aa:(id)sender {
    //创建动作表单
    UIActionSheet *a=[[UIActionSheet alloc]initWithTitle:@"Hello
    World!"
delegate:nil cancelButtonTitle:@"OK" destructiveButtonTitle:
@" Destroy " otherButtonTitles:@"Negotiate",@"Compromise", nil];
    [a showInView:self.view];           //将动作表单添加到当前的视图中
}
@end
```

运行结果如图 5.8 所示。

图 5.8　示例 5-6 运行结果

5.2.3　响应动作表单

在图 5.8 所示的运行结果中，OK、Destory、Negotiate 和 Compromise 这 4 个按钮的功能都是退出动作表单，但是使用 cancelButtonTitle:设置的按钮 OK 才是退出动作表单。剩下的 3 个按钮，它们是没有任何作用的，为了使动作表单提供选择操作，使用 ClickedButtonAtIndex()方法来实现响应动作表单。

【示例 5-7】 以下程序显示了一个动作表单。当单击 Click Me!按钮，就会弹出具有 4 个按钮的动作表单，这 4 个按钮分别为 OK、 Destory、Negotiate 和 Compromise。单击 OK 按钮，动作表单就会退出。当单击剩下的 3 个按钮时，就会将对应按钮的标题显示在 TextField 视图中。操作步骤如下：

（1）创建一个项目，命名为 5-7。

（2）单击打开 ViewController.xib 文件，将 Round Rect Button 视图拖放到用户设置界面，并将其标题改为 Click Me!，将该视图和 ViewController.h 文件进行动作 aa:的声明和关联。

（3）拖动一个 Label 视图到用户设置界面，双击将标题改为“你选择的按钮为：”。再拖动一个 TextField 视图到用户设置界面的 Label 视图的右侧，这时用户设置界面的效果如图 5.9 所示。

图 5.9　用户设置界面的效果

（4）单击打开 ViewController.h 文件，进行插座变量的声明，程序代码如下：

```
01  #import <UIKit/UIKit.h>
02  @interface ViewController : UIViewController{
03      IBOutlet UITextField *te;            //插座变量的声明
04  }
05  - (IBAction)aa:(id)sender;
06  @end
```

（5）将 ViewController.h 文件中声明的插座变量和 ViewController.xib 文件中的 TextField 视图进行关联。

（6）单击打开 ViewController.m 文件，编写程序代码，实现显示具有 4 个按钮的动作表单以及单击这些按钮所对的响应。程序代码如下：

```
01  #import "ViewController.h"
02  @interface ViewController ()
03  @end
04  @implementation ViewController
05  - (void)viewDidLoad
06  {
07      [super viewDidLoad];
08       // Do any additional setup after loading the view, typically from a nib.
09  }
10  - (void)didReceiveMemoryWarning
11  {
12      [super didReceiveMemoryWarning];
13      // Dispose of any resources that can be recreated.
14  }
15
16  - (IBAction)aa:(id)sender {
17       UIActionSheet *a=[[UIActionSheet alloc]initWithTitle:@"Hello
         World!" delegate:self
18   cancelButtonTitle:@"OK" destructiveButtonTitle:@"Destroy"
     otherButtonTitles:
19  @"Negotiate",@"Compromise", nil];          //创建动作表单
20      [a showInView:self.view];              //将动作表单添加到当前的视图中
21  }
22  -(void)actionSheet:(UIActionSheet *)actionSheet clickedButtonAtIndex:
    (NSInteger)buttonIndex{
23      NSString *t=[actionSheet buttonTitleAtIndex:buttonIndex];
                                               //设置被按下键的标题
24  //判断用户单击的按钮
25      if([t isEqualToString:@"Destroy"]){
26          te.text=@"Destroy";
```