Table of Contents

译者的话

iOS开发现在越来越火，也出现了非常多的优秀应用。但是大多数应用都忽略了一点：支持无障碍。

虽然绝大部分用户都可以正常使用APP，但是有一些用户具有视力或者其他障碍，因此无法像我们一样直接操作APP。苹果已经给出了解决办法，那就是使用VoiceOver。

VoiceOver是一个语音辅助软件，具备屏幕阅读器的功能，因此视障者及其他无法正常使用APP的用户都可以通过VoiceOver来进行操作。

但是VoiceOver并不是万能的，并不能兼容开发者自定义的控件和视图，因此作为开发者，需要通过一些额外的工作让APP可以支持无障碍使用。

支持VoiceOver的方法非常简单，快的话一个小时就能看完这篇教程，希望每个开发者都可以让自己的APP支持无障碍使用，用我们的双手给他们创造更好的生活，谢谢。

> 这篇教程翻译自苹果的官方开发教程，原文地址。水平有限，有错误和不足之处还希望大家指出。本项目的GitHub地址。

简介

> 翻译：umcsdon

使用iOS 3.0及以上版本时，VoiceOver 可以帮助有视觉障碍的用户使用他们的iOS设备。在iOS 3.0中引入的UI无障碍编程接口，可以帮助开发人员打造 VoiceOver 用户无障碍应用程序。简单的说，VoiceOver 描述应用程序的用户界面，同时帮助用户使用语音和声音浏览应用程序的视图和控件。熟悉Mac OS X系统提供的 VoiceOver 用户可以利用他们的经验来帮助他们快速上手并使用设备中的 VoiceOver。

运行在iOS 3.0及以上版本的iPhone应用程序应该支持VoiceOver用户访问. iOS和iOS SDK使用以下方式支持这一目标:

- 标准UIKit控件和视图默认支持访问
- 提供UI无障碍编程接口，此接口简化了开发iPhone无障碍应用程序的流程
- 提供工具，从而帮助你在你的代码中实现无障碍使用功能以及对你的应用程序进行无障碍测试

如果你正在开发或升级一个iPhone应用程序，你应该阅读这个文档来学习如何开发 VoiceOver 无障碍应用程序。

文档组成

这个文档包含了以下的章节：

- “理解 iOS 无障碍开发”简要介绍了 VoiceOver 在设备上的工作原理，同时介绍了开发无障碍应用程序的一些编程接口和工具
- “让你的 iOS 应用程序可以被无障碍使用” 深入介绍了如何使你的程序可以被 VoiceOver 用户无障碍使用

参阅

本文档之前包含如何使用 VoiceOver 和无障碍检测器对应用进行无障碍测试

这些内容已经被转移倒了一个叫做 “Verifying App Accessibility on iOS” 的文档中。

理解iOS的无障碍特性

> 翻译：umcsdon

从一开始，基于iOS的设备就包含了很多使得大家都能很容易地使用iOS设备的特性。其中包括可视化的语音留言信箱，邮件大字体，网页、照片、地图缩放。通过引入以下的无障碍特性，有视碍、听碍、肢体残障的人士甚至可以更加方便的使用他们的设备。

- 缩放。放大整个设备屏幕。
- 黑白转换。转换屏幕的颜色。
- 单声道化。把左右声道混合成一个信号，在左右耳播放。
- 自动文本播报。当用户打字时，自动播放文本纠错与候选建议。
- 语音控制。允许用户使用语音命令来打电话和控制iPod回放。

此外，视障用户靠VoiceOver的帮助来使用设备。

无障碍特性与VoiceOver

VoiceOver 是苹果公司创新的读屏技术，通过这个技术，用户无需看屏幕就可以控制设备。VoiceOver在用户界面与用户触控之间充当媒介，描述了应用程序中的元素与行为。当VoiceOver激活时，用户无需担心意外地删除了联系人或者拨打了电话，因为VoiceOver会提醒用户界面位置，他们可以操作什么，结果如何。

一个程序是不是无障碍的，取决于用户可以交互的所有界面元素是否是无障碍的。一个界面元素是否是无障碍的取决于它是否在恰当的时候告知用户它是一个无障碍的元素。

一个无障碍的界面元素必须准确提供关于它的信息才能被使用。这些信息包括它在屏幕的位置、名称、行为、值和类型。这正是VoiceOver播报给用户的信息。iOS SDK包含了编程接口和工具，来帮你确保程序中的用户界面元素是无障碍且好用的（更多信息，请查看“iOS无障碍API与工具”（第7页））。

为什么写没有障碍的应用

你应该让你你的iPhone应用能被VoiceOver的用户们无障碍的使用，因为：

- 这会增加你的用户基数。你已经努力的创造优秀的程序，不要错过让你的程序能被更多用户使用的机会。
- 无障碍应用可以使你的用户无需看屏幕。视障用户可以通过VoiceOver的帮助来使用的你的应用。
- 写无障碍程序使你理解无障碍准则。很多管理部门编写了无障碍准则，你为VoiceOver用户编写无障碍的iPhone应用能帮助你满足这些准则。
- 这是正确的决定。

支持无障碍特性不会影响你创造优异iPhone应用的能力，记得这点非常重要。为用户界面编程接口增加薄薄的一个功能层，不会改变应用的外观，也不会妨碍到应用的主要逻辑。

iOS无障碍API与工具

iOS 3.0及之后的版本包含了用户界面无障碍编程接口，这个轻量级的API可以让应用提供全部所需信息，让VoiceOver来给用户描述界面，以帮助视障用户使用应用。 用户界面无障碍编程接口是UIKit的一部分，默认通过标准UIKit控件和视图来实现。也就是说，当你使用标准的控件和视图时，大部分使应用无障碍的工作都已经完成了。编无障碍程序可以简单到只是给用户接口元素提供准确易懂的描述，简单的程度取决于你程序自定义的程度。

iOS SDK也提供工具来帮你编写无障碍应用：

- 当你设计nib文件时，Interface Builder观察器面板提供了一个简单的方式来添加描述性的无障碍信息。想进一步了解如何添加这些信息，请看”在Interface Builder中添加自定义属性信息“（第19页）。
- Accessibility Inspector可以显示嵌入到应用用户界面中的无障碍信息，你可以在iOS模拟器中运行应用来验证这些信

息。想进一步了解如何检查应用中的无障碍信息，请看“在iOS模拟器中使用Accessibility Inspector来除错”。

此外，你可以使用VoiceOver本身来测试应用的无障碍功能。想学习如何用VoiceOver来测试你的应用，请看“用VoiceOver在你的设备上测试无障碍信息”。

UI无障碍编程接口

UI无障碍编程接口包含两个非正式协议，一个类，一个函数，还有一些常量。

- UIAccessibility非正式协议。实现UIAccessibility协议的对象可以报告无障碍状态（即他们是否是无障碍的）并且提供关于他们的描述性信息。UIAccessibility协议默认由标准UIKit控件和视图实现。
- UIAccessibilityContainer非正式协议。此协议通过子类UIView来使它所包含的部分或者全部对象是无障碍的分离元素，当非UIView子类的对象被UIView视图包含时，这些元素不会自动变成无障碍元素，此时UIAccessibilityContainer协议就非常有用。
- UIAccessibilityElement类。定义了一个对象是否能通过UIAccessibilityContainer协议返回。你可以创建UIAccessibilityElement实例来代表无法自动变为无障碍的元素，例如一个没有继承自UIView的对象，或者一个不存在的对象。
- UIAccessibilityConstants.h头文件。此头文件定义了用来描述某些特征的常量。这些常量即可展示的无障碍元素，还有可被应用发布的通知。

无障碍属性

用来描述无障碍用户界面元素的无障碍属性构成了UI Accessibility API的核心。当用户访问或操作控件和视图时，VoiceOver会告知用户无障碍属性的相关信息。

因为无障碍属性封装了区分不同控件和视图的信息，所以无障碍属性也是你最有可能用到的编程接口的组件。就标准UIKit控件和视图来说，确保应用中默认的属性信息准确可能就够了。就自定义的控件和视图来说，或许要添加大部分的属性信息。

UI Accessibility 编程接口定义了如下属性。

- 标签。一个简短的本地化词或短语，此短语简洁地描述控件或者视图，但是不能识别元素类型。例如“添加”、“播放”。
- 特征。一个或多个独立特征的组合。每个特征描述一个元素状态、行为、使用中的某一方面。例如当元素的行为如同被选中键盘上的按键，这个元素就有了Keyboard Key和Selected的组合特征。
- 提示。一个简短的本地化短语，描述发生在元素上动作的结果。例如“添加标题”或者“打开购物列表”。
- 框架。元素在屏幕上坐标的框架。由CGRect结构体提供，详细说明了元素的屏幕位置和大小。
- 值。不是由标签定义的元素时的当前值。例如，一个幻灯片的标签可以是”速度”，但是它当前的值是“50%”。

无论由你指定还是默认情况，无障碍元素需有无障碍属性值。无障碍元素都要有框架和标签属性。因为无障碍元素必须能够报告它在用户界面上的位置，所以框架属性是强制的。（注意一个由UIView继承的对象默认包含了框架属性值）。因为标签属性包含了VoiceOver播报的无障碍元素的名称和描述，所以它也是强制的。

当提示和特征属性不适用时，就无需这些属性。例如不能触发动作的元素不需提示属性。

仅当元素的内容是可改变并且永不被标签描述时，一个无障碍元素才需要提供给值属性提供内容。例如，一个文本域包含了一个可能有Email标签的Email地址，但是它的内容取决于用户的输入，通常格式为”username@address。“图1-1显示了VoiceOver可以提供了一些信息。

图 **1-1** VoiceOver可以读出由无障碍元素提供的信息。

VoiceOver用户靠听标签和提示来使用应用。因此，确保应用中的每一个无障碍元素提供准确全面的描述是极其重要的。标准UIKit控件和视图的默认值信息经常是准确的，但是你应该检查一下确保无误。对于自定义控件和视图，你可能不得不自己提供部分或者全部信息。关于如何去提供这些信息的指导，请看"打造好用的标签和提示"（第16页）。

让你的iOS APP支持无障碍使用

> 翻译：umcsdon

要做到无障碍使用，一个iPhone应用必须支持用户使用VoiceOver访问用户界面上的元素。总体来说，这意味着你需要：

- 让用户界面上每个需要进行交互的元素都可以无障碍使用，包括那些只用来显示信息的元素，比如静态文本和控件。
- 所有可以无障碍使用的元素都包含准确并且有用的信息。

除此之外，你还可以通过这些方法来增强VoiceOver用户体验：使用表格视图并确保应用中的动态元素也可以无障碍使用。

让用户界面元素支持无障碍使用

在“无障碍特性与VoiceOver”中我们说过，一个无障碍的界面元素必须准确提供关于它的信息才能被使用。虽然支持无障碍使用并不代表这个元素可以被VoiceOver用户使用，但是这是应用程序无障碍化的第一步。

在“iOS无障碍API与工具”中我们说过，标准的UIKit控件和试图都支持无障碍使用。如果你只使用了标准的UIKit控件，那可能应用本身就已经支持无障碍使用了。详情参见“提供准确和有用的属性信息”。

如果你创建一个自定义视图来展示信息或者和用户进行交互，那你必须确保这些视图支持无障碍使用。无障碍化之后，你需要确认这些视图的确包含用户需要的无障碍信息。（参见“提供准确和有用的属性信息”）。

从无障碍的角度来说，自定义视图要不就是一个独立的视图，要不就是一个容器视图。独立视图并不包含其他需要支持无障碍的视图。举例来说，一个自定义的UIControl子类会展示一个图标并具有按钮的行为，但是除了按钮本身之外并不包含其他可以和用户进行交互的元素。如果要让一个独立视图支持无障碍，请阅读“让自定义独立视图支持无障碍使用”。

容器视图恰好相反，包含其他可以和用户进行交互的元素。举例来说，一个自定义的UIView子类会绘制一个几何图形，这个图形就是一个不同于容器视图的可以和用户进行交互的元素。容器视图内的独立元素并不会自动支持无障碍使用（因为它们不是UIView的子类），也不会自动提供任何无障碍信息。如果要让一个容器视图支持无障碍，请阅读“让自定义容器视图的内容支持无障碍使用”。

让自定义独立视图支持无障碍使用

如果你的应用包含一个可以和用户进行交互的自定义独立视图，那你必须让这个试图支持无障碍使用。（还记得吧，独立视图就是不包含其他需要支持无障碍的视图。）

除了使用Interface Builder，还可以通过两种编程方法让自定义独立视图支持无障碍使用。第一种方法是在初始化视图的时候设置它的无障碍状态。如下面的代码片段所示：

```
@implementation MyCustomViewController
- (id)init
{
  _view = [[[MyCustomView alloc] initWithFrame:CGRectZero] autorelease];
  [_view setIsAccessibilityElement:YES];

  /* Set attributes here. */
}
```

另一种方法是实现UIAccessibility协议的isAccessibilityElement方法。如下面的代码片段所示：

```
@implementation MyCustomView
  /* Implement attribute methods here. */

- (BOOL)isAccessibilityElement
{
  return YES;
}
```

> 注意：在这两个代码片段中，都是用注释来代替真实的代码。如果想查看完整的代码片段可以阅读“通过编程来定义自定义属性信息”。

让自定义容器视图的内容支持无障碍使用

如果你的应用中展示了一个自定义视图，其中包含其他可以进行用户交互的元素，那你需要实现每个元素的无障碍使用。与此同时，你必须让容器视图本身不支持无障碍使用，因为用户是和容器的内容交互，不是和容器本身交互。

为了实现这一点，你自定义的容器视图需要实现UIAccessibilityContainer协议。这个协议会定义一些方法并把可以访问的元素放入一个数组中。

下面的代码片段展示了一个自定义容器视图的部分实现。可以看到这个容器视图只会在调用UIAccessibilityContainer协议的方法时才会创建无障碍元素数组。所以，如果iPhone的无障碍并没有被激活，那就不会创建数组。

```
@implementation MultiFacetedView

- (NSArray *)accessibleElements

{

  if ( _accessibleElements != nil )

  {

    return _accessibleElements;

  }

  _accessibleElements = [[NSMutableArray alloc] init];


  /* Create an accessibility element to represent the first contained element and initialize it as a component of MultiFacetedView. */

  UIAccessibilityElement *element1 = [[[UIAccessibilityElement alloc] initWithAccessibilityContainer:self] autorelease];


  /* Set attributes of the first contained element here. */

  [_accessibleElements addObject:element1];


  /* Perform similar steps for the second contained element. */

  UIAccessibilityElement *element2 = [[[UIAccessibilityElement alloc] initWithAccessibilityContainer:self] autorelease];


  /* Set attributes of the second contained element here. */

  [_accessibleElements addObject:element2];


  return _accessibleElements;

}


/* The container itself is not accessible, so MultiFacetedView should return NO in isAccessiblityElement. */

- (BOOL)isAccessibilityElement

{

  return NO;

}


/* The following methods are implementations of UIAccessibilityContainer protocol methods. */
```

```
- (NSInteger)accessibilityElementCount

{

  return [[self accessibleElements] count];

}


- (id)accessibilityElementAtIndex:(NSInteger)index

{

  return [[self accessibleElements] objectAtIndex:index];

}


- (NSInteger)indexOfAccessibilityElement:(id)element

{

  return [[self accessibleElements] indexOfObject:element];

}

@end
```

提供准确和有用的属性信息

给无障碍元素添加属性信息包含两部分：

- 创建准确并且有用的信息
- 确保应用中的无障碍元素可以正确地提供这些信息

如果你使用的是自定义视图，那必须包含所有的属性信息，参见“如何创建有用的标签和提示”以及“选择合适的特性”。

即使你使用的是标准UIKit控件和试图，也可以修改一些默认的属性信息，让它们更适合你的应用，参见“修改默认的属性信息”。

如果你需要提供或者修改标准和自定义UI元素的无障碍属性，那你可以使用Interface Builder（参见“在Interface Builder中定义自定义属性信息”）或者通过编程来实现。（参见“通过编程定义自定义属性信息”）。

修改默认的属性信息

作为标准UIKit控件和视图内置无障碍性的一部分，iOS已经提供了一些VoiceOver可以使用的默认的属性信息。大多数情况下这些信息已经足够了。然而，有时候你可以修改这些信息从而增强VoiceOver的用户体验：

- 如果你使用一个标准UIKit控件或者视图来展示一个系统提供的图标或者标题，首先确认你的用法是否符合它们的含义（参见“iOS用户界面教程”），然后确认默认的标签属性是否能让用户理解这个控件或者视图的效果。如果不能的话，最好提供一个提示属性。

举例来说，如果你想在导航栏中放置一个添加按钮并在UIBarButtonItem对象中使用系统提供的加（+）图标，那它会自动包含一个默认的标签属性：添加。如果用户可以明确地知道每次点击这个按钮会添加哪个项目，那就不需要提供提示属性。但是如果可能产生误解的话，你应该提供一个自定义的提示并描述使用这个按钮的效果，比如“添加一个账户”或者“添加一个评论”。

- 如果你在标准的UIKit视图（比如UIButton对象）中展示一个自定义的图标或者图片，那你需要提供一个自定义标签属性来进行描述。

如何创建有用的标签和提示

当VoiceOver用户使用你的应用时，他们会通过VoiceOver的描述来判断应用的作用和用法，因此这些描述非常重要，必须非常准确并且有用。本节会告诉你如何创建有用的标签和提示。

如何创建标签

标签属性可以描述用户界面中的元素，因此每个无障碍的用户界面元素都需要提供标签属性的内容。

> 注意：表格中的一行也可以包含一个标签属性。然而，创建表格行标签的方法和创建其他类型控件和视图的方法不一样。具体的创建方法参见“让表格视图支持无障碍使用”。

有一个判断标签内容的好办法，那就是思考一下正常的用户只通过观察可以了解什么信息。如果你的用户界面设计得很好的话，正常的用户应该可以通过阅读标题和图标来了解控件和视图的作用。同理，VoiceOver用户需要从标签属性中获取的就是这些信息。

如果你使用的是自定义控件或者视图，或者在标准控件和视图中使用了自定义图标，那你需要提供包含下面内容的标签：

- 可以简洁地描述元素。如果标签只包含一个词那就最好了，比如“添加”、“播放”、“删除”、“搜索”、“喜欢”和“音量”。

好好设计你的应用，让每个元素的含义和用法在当前上下文中可以通过一个词来表示。然而，有时候可能必须要使用一句话才能准确描述一个元素。这时选择一个尽量短的句子，比如“播放音乐”、“添加姓名”或者“添加到事件中”。

- 不要包含控件或者视图的类型。类型信息包含在元素的特性属性中，因不需要在标签中重复。

举例来说，如果你的添加按钮包含控件类型，那VoiceOver用户听到的是“添加按钮按钮”。用户体验非常不好。

- 用大写字母开头。这可以帮助VoiceOver用恰当的语气来阅读标签。
- 不要用句号结束。标签并不是一句话因此并不需要用句号结束。
- 支持本地化。确保你的应用已经本地化了所有的字符串，包括无障碍属性中的字符串。通常来说，VoiceOver会使用用户在国际属性中设置的语言来进行朗读。

如何创建提示

提示属性描述了控件和视图的执行结果。只有无法通过标签来看出元素结果的时候才使用提示。

举例来说，如果你的应用中有一个播放按钮，那用户可以根据上下文很清楚地判断出按下这个按钮有什么结果。然而，如果你允许用户通过点击列表中的歌名来播放歌曲，那你就需要创建一个提示来进行描述。原因是列表项目的标签描述的是项目本身（在本例中是指歌名），并不是用户点击它的结果。

> 注意：VoiceOver用户可以设置是否听提示，默认是开启的。

如果控件或者视图的作用无法通过标签了解，那就创建一个这样的提示：

- 可以简洁地描述结果。即使需要提示的控件和视图很少，还是要尽量让描述更简洁。这样可以减少用户在使用元素前需要花费在听上的时间。

然而，这并不是说需要牺牲语法的简洁性。比如说，把“添加一个城市”改成“添加城市”并不能显著减少提示的长度，但是会让听者更加迷惑。

- 用动词开头并不使用祈使句。确保使用第三人称单数的动词形式，比如“Plays”，不要用祈使句“Play”，因为后者听起来像一条命令。比如说，你肯定不希望用户听到的是“播放这个歌曲”，而是“可以播放歌曲”。

有一种方法可以帮你找到正确的单词，假设你正在给一个朋友描述控件的用法。你可能会说“点击控件可以播放歌曲”。通常来说你可以把句子的第二部分当做提示（本例中是“可以播放歌曲”）。

- 用大写字母开头并使用句号。即使提示只是一条短语，不是一句话，也要用句号结束。这可以帮助VoiceOver用合适的语气来朗读它。

- 不要包含动作或者手势的名称。提示并不会告诉用户如何触发动作，它只告诉用户动作的结果。因此，不要创建类似“点击来播放音乐”、“点击来购买项目”或者“滑动来删除项目”这样的提示。

这非诚重要，因为VoiceOver会使用VoiceOver特定的手势来和元素进行交互。如果你在提示中使用了另一个手势的话会令用户非常困惑。

- 不要包含控件或者视图的名字。用户可以从标签属性中获取这个信息，所以不需要在提示中重复。因此，不要创建类似“保存按钮可以保存你的编辑”或者“返回按钮会返回到之前的屏幕”这样的提示。

- 不要包含控件或者视图的类型。用户可以通过元素的特性属性来获取这个信息，因此不要创建类似“用来添加姓名的按钮”或者“用来控制缩放的滑块”这样的提示。

- 支持本地化。和无障碍标签一样，提示也需要进行本地化。

选择合适的特性

特性属性包含一个或者多个特性，用来描述一个无障碍用户界面元素的行为。因为一些单独的特性可以组合起来一起描述一个元素，因此可以准确地描述一个元素的行为。

> 注意：单独的特性通过OR操作符组合。在代码以外的情况下都使用“组合”来描述这个方法。

一个标准的UIKit控件，比如一个按钮或者文本框，都会在特性属性中提供默认的内容。如果你创建了一个自定义控件或者视图，那就必须提供元素特性属性的内容。

UI无障碍编程接口定义了12个独立的特性，有些是可以组合的。有些特性描述的是控件类型（比如按钮）或者对象类型（比如图片）。有些特性描述的是控件的行为，比如可以播放引用。

你可以在应用中使用下面的特性来描述元素：

- 按钮
- 链接
- 搜索框
- 键盘按键
- 图片
- 播放音乐
- 选择
- 总觉元素
- 频繁更新
- 不可用
- 空

通常来说，控件类型特性可以和描述行为特性组合。举例来说，你可以组合“按钮”特性和“播放音乐”特性来描述一个自定义控件，它是一个按钮，点击的时候可以播放音乐。

大多数情况下你可以认为控件类型特性是互斥的。也就是说，你不应该使用多余一个类型特性来描述应用中的元素，应该选择最合适的那个特性。然后，如果你的元素有额外的行为，可以用类型特性组合一个行为特性。

举例来说，假设用户在iPhone的Safari中点击一个链接会显示一个图片，那你应该组合“图片”和“链接”特性来描述这个元素。另一个例子是点击键盘按键的时候会修改另一个键盘按键。你可以组合“键盘按键”和“选择”特性来描述这个元素。

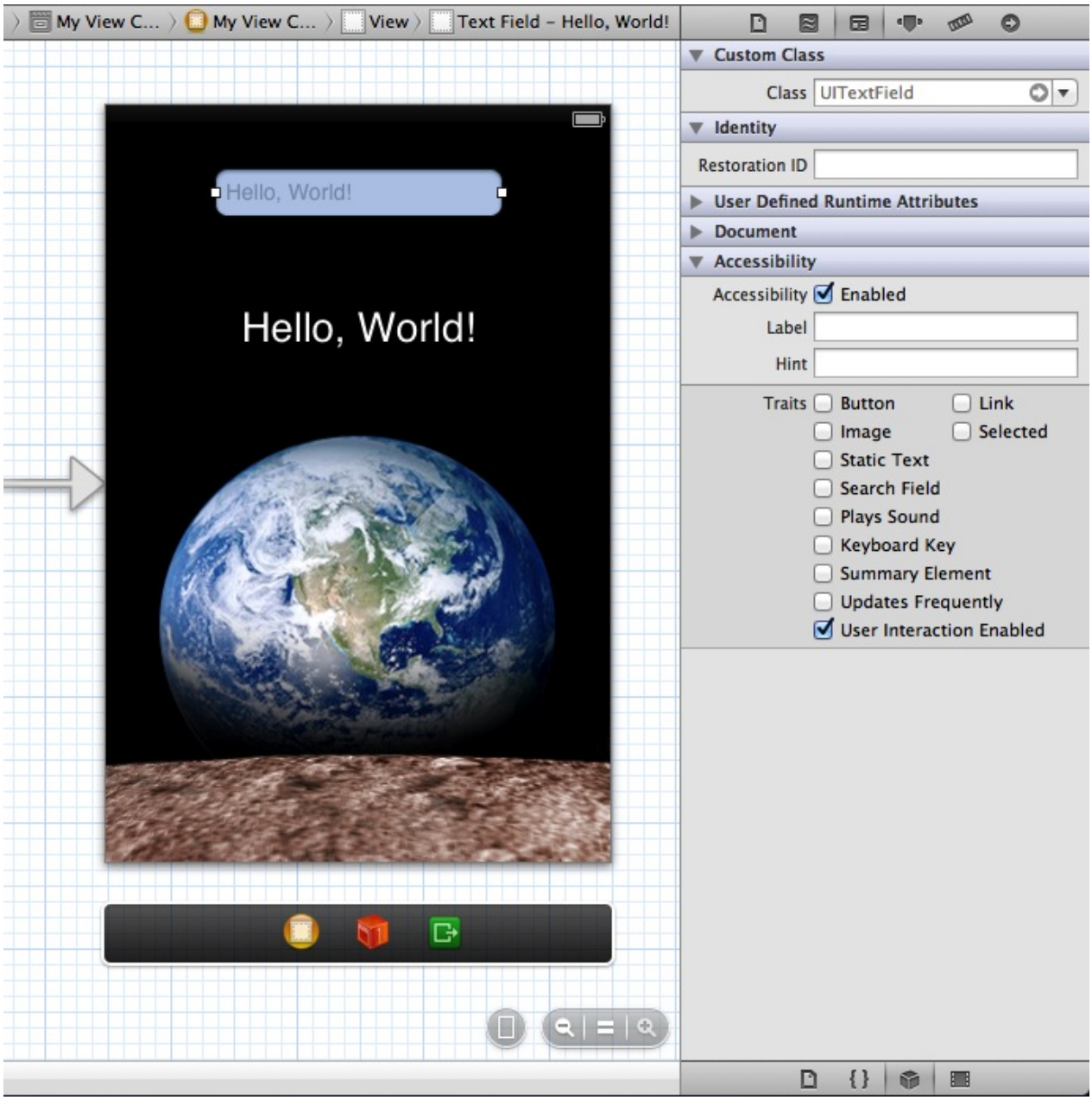
如果要查看更多关于特性描述控件的例子，可以使用Accessibility Inspector来查看标准控件的默认特性。Accessibility Inspector的用法参见“在iOS模拟器中使用Accessibility Inspector调试无障碍性”。

在Interface Builder中定义自定义属性信息

iOS SDK 3.0自带一个Interface Builder，其中包含应用无障碍性相关的功能。如果你的应用包含标准UIKit控件和视图，那你可以在Interface Builder中完成所有的无障碍工作。

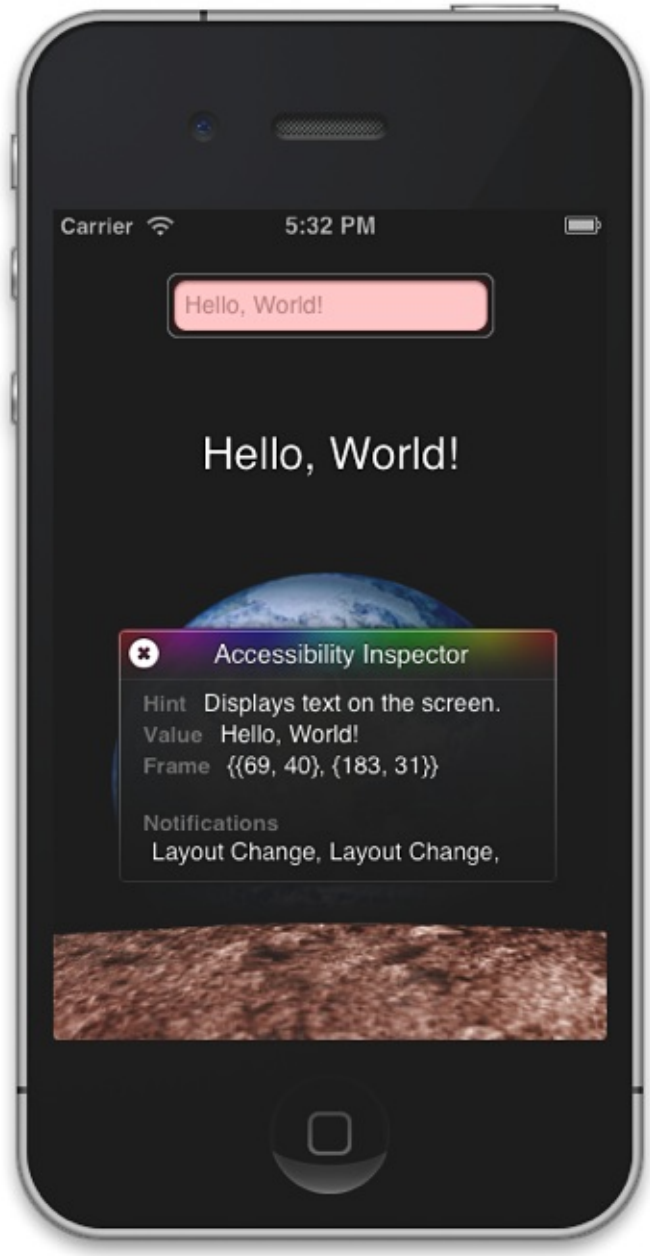
可以使用Interface Builder设置元素的无障碍状态并提供自定义的标签、提示和特性属性的内容。具体的设置方法是，在你的nib文件中选择用户界面元素并打开Idendidy标签，可以在Accessibility部分看到下图的内容：

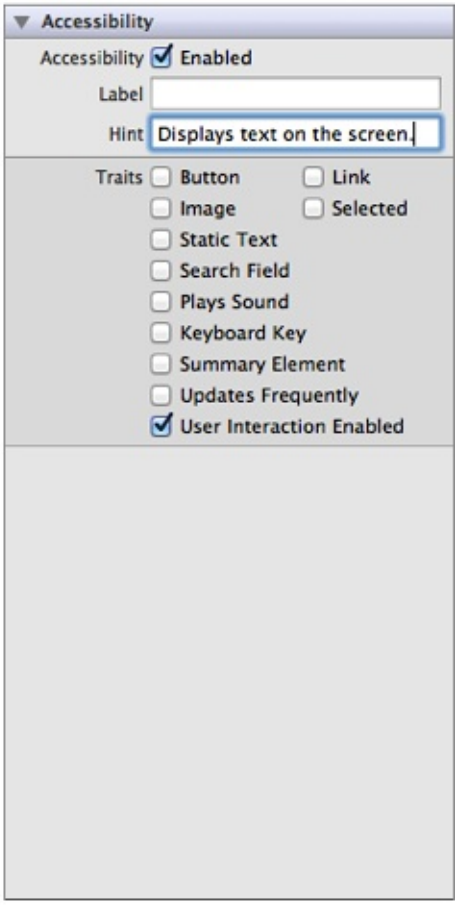
图 **2-1** Interface Builder中所显示的标准文本框的默认无障碍信息。



如图所示，nib文件中使用的标准的文本框默认是无障碍的，并且包含默认的标签、提示和特性属性的信息。（注意，对于包含占位符文字的文本框来说，默认标签是占位符文本。）你可以任意修改这些默认值，如下图所示（下图同样展示了Accessibility标签如何展示文本框的无障碍信息。具体的使用方法参见“在iOS模拟器中使用Accessibility Inspector调试无障碍性”）：

图 **2-2** 在Interface Builder中修改无障碍信息。





通过编程定义自定义属性信息

如果你愿意，可以通过编程来提供自定义属性的信息。如果你没有使用Interface Builder的话可能会选择这么做。

就像“让自定义独立视图支持无障碍使用”中描述的一样，你可以在实例化视图的视图子类中设置无障碍信息。两种方法都可以用，但是如果你的视图会动态展示数据或者经常改变内容的话，那在子类中实现属性方法比在实例中实现更好。举例来说，如果你想显示时间的话，那应该在子类刷新数据的方法中实现无障碍性，因为如果只在实例化子类的时候设置属性，那更新数据之后就无法支持无障碍使用了。

下面的代码片段是基于“让自定义独立视图支持无障碍使用”的方法写成的，包含一些属性特有的方法。如果你想在子类中实现无障碍方法，那可以参考下面的代码：

```
@implementation MyCustomView

- (BOOL)isAccessibilityElement

{

  return YES;

}


- (NSString *)accessibilityLabel

{

  return NSLocalizedString(@"MyCustomView.label", nil);

}


/* This custom view behaves like a button. */

- (UIAccessibilityTraits)accessibilityTraits

{

  return UIAccessibilityTraitButton;

}


- (NSString *)accessibilityHint

{

  return NSLocalizedString(@"MyCustomView.hint", nil);

}

@end
```

如果你想在实例化视图的代码中使用UIAccessbility协议中设置属性的方法的话，那可以参考下面的代码：

```
@implementation MyCustomViewController

- (id)init

{

 _view = [[MyCustomView alloc] initWithFrame:CGRectZero];


 [_view setIsAccessibilityElement:YES];

 [_view setAccessibilityTraits:UIAccessibilityTraitButton];

 [_view setAccessibilityLabel:NSLocalizedString(@"view.label", nil)];

 [_view setAccessibilityHind:NSLocalizedString(@"view.hint", nil)];
}
```

让表格视图支持无障碍使用

如果你的应用中展示了一个表格，它的每个单元格中包含非文字（或者不只是文字）的内容，那你可以通过一些方法让它更好地支持无障碍使用。同理，如果你的表格视图中每行展示的信息许多，那可以把这些信息聚合到标签中。

> 注意：如果你的单元格包含任何标准的表格视图元素，比如信息展示指示、细节展示指示或者删除控件，那你不需要额外的工作。然而如果你的单元格中包含其他类型的控件比如选择器或者滑块，那你需要让这些元素都支持无障碍使

> 用。

如果应用中的单元格包含多种不同元素，首先要判断用户是直接和单元格进行交互还是和单个元素进行交互。如果用户需要访问单元格中的独立元素，那你应该：

- 让每个独立元素都支持无障碍使用
- 让单元格本身不支持无障碍使用
- 在单元格的标签属性中简单描述单元格的所有内容。注意，在这种情况下标签也被看作是单元格中一个支持无障碍访问的元素。

你可能已经发现了，包含多个项目（比如文字和控件）的单元格和容器视图很像。但是，你不需要把单元格看作是一个容器视图并实现任何UIAccessibilityContainer协议的方法，因为单元格本身就是一个容器。

如果单元格包含许多块信息，那应该把这些信息都组合到一个标签属性中，这样VoiceOver用户可以用一个手势获取所有信息。

一个很好的例子就是内置的股票应用。它并没有用单独的字符串来提供公司名、当前股票价格和价格波动，而是组合到一个标签中，听起来类似“苹果公司，432.39美元，涨幅1.3%”。注意标签中的逗号，当你结合多个片段的时候，可以使用逗号，这样VoiceOver就会在每个逗号短暂停留，让用户更容易理解信息。

下面这段代码把两个元素的信息组合到一个标签中：

```
@implementation CurrentWeather

/* This is a view that provides weather information. It contains a city subview and a temperature subview, each of which provides a se

- (NSString *)accessibilityLabel

{

    NSString *weatherCityLabel = [self.weatherCity accessibilityLabel];

    NSString *weatherTempLabel = [self.weatherTemp accessibilityLabel];



    /* Combine the city and temperature information so that VoiceOver users can get the weather information with one gesture. */

    return [NSString stringWithFormat:@"%@, %@", weatherCityLabel, weatherTempLabel];
}
@end
```

让动态元素支持无障碍使用

如果用户界面元素会动态改变，那你需要确保无障碍信息也会更新。如果应用屏幕的布局发生改变，那你需要发送通知，这样VoiceOver才能帮助用户切换到新布局。UI Accessibility编程接口提供了两种通知类型，你可以使用它们。（具体的教程参见“UIAccessbility协议参考”中的“通知”。）

如果用户界面元素可以有不同的状态，那你需要在代码中返回所有状态对应的无障碍信息。由于这些改变在用户的动作发生后发生，所以最好把实现添加到子类中而不是实例化子类的代码中。

下面的代码展示了如何处理动态状态的改变以及如何在屏幕布局发生改变的时候发送通知。代码中的UIView子类的行为类似一个自定义键盘按键。按键的无障碍标签会根据实例的内容进行改变，并且会根据shift键是否按下进行改变。按键也会根据实例内容和是否被选中返回不同的无障碍特性。注意，下面的代码假设会有许多方法查询键盘的状态：

@implementation BigKey

- (NSString *)accessibilityLabel

{

NSString *keyLabel = [_keyLabel accessibilityLabel];

if ([self isLetterKey])

{

```
    if ( [self isShifted] )
    {
      return [keyLabel uppercaseString];
    }
    else
    {
      return [keyLabel lowercaseString];
    }
```

}

else

{

```
    return keyLabel;
```

}

}

- (UIAccessibilityTraits)accessibilityTraits

{

UIAccessibilityTraits traits = [super accessibilityTraits] | UIAccessibilityTraitKeyboardKey;

/ If this is the shift key and it's selected, users need to know that shift is currently in effect. /

if ([self isShiftKey] && [self isSelected])

{

```
    traits |= UIAccessibilityTraitSelected;
```

}

return traits;

}

- (void)keyboardChangedToNumbers

{

/ Code to perform the change to a number keyboard here. /

/ *Send a notification of this change to the screen layout.* /

UIAccessibilityPostNotification(UIAccessibilityLayoutChangedNotification, nil);

}

@end

```
## 让非文字数据支持无障碍阅读

有时你的应用会显示一些不能自动支持无障碍方法的数据，比如一张图片，这时你需要在无障碍标签中提供信息，这样VoiceOver用户就可以理解图片传达的内容

下面的代码展示的是一个自定义视图，它会用星星的数量来表示项目的评分。代码展示了如何根据星星数量返回一个合适的无障碍标签：
```

@implementation RatingView

/ *Other subclass implementation code here.* /

- (NSString *)accessibilityLabel

{

/ *_starCount is an instance variable that contains how many stars to draw.* /

NSInteger starCount = _starCount;

if (starCount == 1)

{

```
ratingString = NSLocalizedString(@"rating.singular.label", nil); // here, ratingString is "star"
```

}

else

{

```
ratingString = NSLocalizedString(@"rating.plural.label", nil); // here, ratingString is "stars"
```

}

return [NSString stringWithFormat:@"%d %@", starCount, ratingString];

}

@end ```