Directing Input to a Specific Responder

Touch events are automatically dispatched to the view in which they occurred, but most other events are directed to the object designated as the first responder. You can also set the first responder programmatically to any of your app's responder objects. User interactions may also cause a view to become the first responder automatically, but only if it wants to be the first responder. For example, tapping a text field causes the text field to become the first responder, but tapping a button triggers the button's action and does not cause it to become the first responder.

터치 이벤트는 이벤트가 발생한 뷰에 자동으로 전달되지만, 대부분의 다른 이벤트 들은 first responder로 지정된 객체에 전달되기도 합니다. 또한 여러분의 앱의 리스폰더 객체를 코드로 first responder으로 설정할 수 있습니다. 특정 객체는 사용자의 상호 작용으로 first responder가 되려는 경우엔 유저의 상호작용이 자동으로 뷰가 first responder가 되게 합니다. 예를 들어, 텍스트 필드를 탭하는 것은 텍스트 필드가 first responder가 되게 하지만 버튼을 탭하면 버튼의 액션이 발생하고 이것은 first responder가 되게 하지는 않습니다.

When an object becomes the first responder, UIKit displays the *input view* associated with that responder. An input view is a custom interface that you can use to gather information. The system keyboard is an example of an input view.

객체가 first responder가 되고 싶어할 때, UIKit은 리스폰더와 관련된 *input view*를 표시합니다. *input view*란 정보를 수집하는 데에 사용할 커스텀 인터페이스를 말합니다. 시스템 키보드는 *input view*의 한 예입니다.

Programmatically Changing the First Responder

To designate an object as the first responder, call its `becomeFirstResponder` method. UIKit makes the object the first responder only if the following conditions are met:

객체를 first responder가 되게하고 싶으면 `becomeFirstResponder`메소드를 호출하세요. UIKit은 아래와 같은 상황일 때 그 객체를 first responder로 만들어 줍니다.

- The current first responder's `canResignFirstResponder` property returns `YES`.

  현재의 first reponder의 `canResignFirstResponder`프로퍼티를 YES로 설정하세요.

- The new responder's `canBecomeFirstResponder` property returns `YES`. (The default implementation of this property returns `NO`, so you must override it to allow your object to become the first responder.)

  first responder가 되고 싶은 새 리스폰더의 `canBecomeFirstResponder` 프로퍼티를 YES로 설정하세요. (이 프로퍼티의 기본 값은 NO입니다. 그래서 first responder로 만들려면 이 프로퍼티를 재정의해야 합니다.)

Most responder objects resign the first responder status readily, but you can return `NO` as needed from the `canResignFirstResponder` property. For example, you might return `NO` to prevent your custom control from resigning as first responder while it contains invalid data.

대부분의 리스폰더 객체는 즉시 first responder 상태를 해제하지만 필요에 따라 `canResignFirstResponder`프로퍼티의 값을 NO로 설정할 수 있습니다. 예를 들어, 커스텀 컨트롤 객체가 유효하지 않은 데이터를 가지고 있을 때 first responder가 해제되는 것을 `canResignFirstResponder`프로퍼티의 값을 NO로 설정함으로 막을 수 있습니다.

Assigning an Input View to a Responder

When an object becomes the first responder, UIKit displays its associated input view. An input view handles user interactions and generates data for the underlying responder object. Input views are used mostly to manage custom data entry for views. For example, the default input view for text fields and text views is the keyboard, shown in Figure 8-1. The keyboard generates character strings and delivers them to the view's delegate.

객체가 first responder가 될 때, UIKit은 그 객체와 관련된 input view를 표시합니다. input view는 유저의 상호작용을 처리하고 데이터를 생성합니다. Input view는 주로 뷰의 커스텀 데이터 항목을 관리하는데 사용됩니다. 예를 들어, 아래 그림 8-1처럼 텍스트 필드와 텍스트 뷰의 기본 input view는 키보드 입니다. 키보드는 문자열을 생성하고 뷰의 델리게이트에 전달합니다.

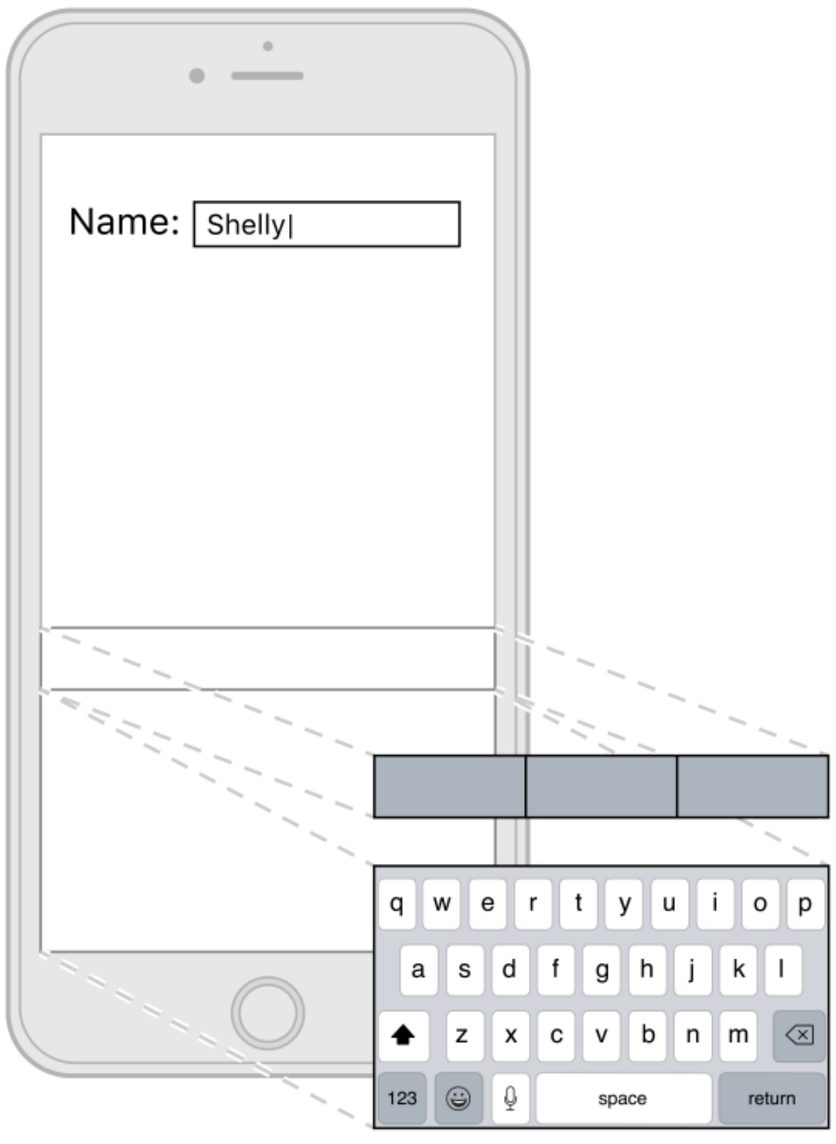

Figure 8-1 The input view for a text field

Every responder can have a custom input view, which you assign to the `inputView` property. (If you prefer to manage your input view using a view controller, assign the view controller to the `inputViewController` property instead.) When the responder becomes the first responder, UIKit animates your input view into position. Here are some tips for setting up your input view:

모든 리스폰더는 `inputView`프로퍼티에 값을 할당하여 커스텀 input view를 가질 수 있습니다. (만약 input view를 뷰 컨트롤러를 이용해 관리하고 싶다면, 대신 뷰 컨트롤러의 `inputViewController` 프로퍼티에 값을 할당하세요.) 리스폰더가 first responder가 되면, UIKit은 input view를 설정한 위치에 애니메이션 효과와 함께 위치시킵니다. 여기 아래에 input view를 설정하는 데에 필요한 몇 가지 팁이 있습니다.

- **Configure your input view to span the width of the screen.** UIKit sets the size of your input view to the screen width and the height you specify for your view.

  input view를 스크린의 너비에 맞게 구성하세요. UIKit은 input view의 크기를 지정한 스크린의 너비와 높이에 맞게 설정합니다.

- **Use a delegate to report data back to your responder.** It is your responsibility to manage communications between your responder and the input view. One way to do this is for your input view to deliver updates to an assigned delegate object.

  델리게이트를 사용하여 데이터를 리스폰더에게 전달하세요. 리스폰더와 input view사이의 통신을 관리해야합니다. 관리하는 방법 중 한가지는 할당 된 델리게이트 객체에 업데이트 내용을 전달하는 것입니다.

- **Use the keyboard notifications to detect when your input view is displayed.** UIKit posts the `UIKeyboardWillChangeFrameNotification`, `UIKeyboardWillShowNotification`, `UIKeyboardDidChangeFrameNotification`, and `UIKeyboardDidShowNotification` notifications (in that order) for your custom input views as well as for the system keyboard. Use these notifications to update your view or animate other changes in your interface. For example, you might adjust the frame of the underlying view to ensure that controls are not hidden by your input view.

  키보드 노티피케이션을 이용하여 input view가 언제 표시되는 지 감지하세요. UIKit은 커스텀 input view 뿐만 아니라 시스템 키보드에 대한 `UIKeyboardWillChangeFrameNotification`, `UIKeyboardWillShowNotification`, `UIKeyboardDidChangeFrameNotification`, `UIKeyboardDidShowNotification` 노티피케이션을 발송합니다. 이 노티피케이션을 이용하여 view를 업데이트 하거나 다른 변화에 대한 애니메이션을 적용하세요. 예를 들어, 컨트롤 객체가 input view에 의해 가려지지 않도록 뷰의 프레임을 조정합니다.

In addition to an input view, your responder can also have an *accessory view*, which you provide using the `inputAccessoryView` or `inputAccessoryViewController` property. Accessory views supplement your main input view. For example, the default accessory view for the keyboard lists potential completions for partially typed words. It is your responsibility to manage any communication between your input view and your accessory view.

input view 외에도, 리스폰더는 `inputAccessoryView` 또는 `inputAccessoryViewController`프로퍼티를 이용하여 *accessory view*를 가질 수 있습니다. 악세서리 뷰는 메인 input view를 보완해줍니다. 예를 들어, 키보드의 기본 악세서리 뷰는 부분적으로 입력된 단어의 자동완성 단어를 나열합니다. input view와 악세서리 뷰 간의 통신을 관리하는 것은 여러분의 책임입니다.