Sample Code

PDF Widgets

Demonstrates adding widgets—interactive form elements—to a PDF document.

Download

iOS 11.0+ | iPadOS 11.0+ | Xcode 11.3+

Overview

The PDF file format allows for annotations to be associated with a page and a location on that page. One powerful annotation type is the widget, which represents interactive form elements. PDFKit supports three widget types: text, button, and choice widgets. Text widgets are basic text fields. Button widgets consist of three subtypes: radio buttons, checkboxes, and push buttons. Choice widgets consist of two subtypes: list (a table view) and combo box (a drop-down). This sample includes text and button widgets.

Create Widget Annotations

All widget annotations are instantiated using the `PDFAnnotation` constructor with subtype `widget`. To inform PDFKit which type of interactive element to add to the document, the `widgetFieldType` property is set for each widget:

```
// Create a text widget
widgetAnnotation.widgetFieldType = PDFAnnotationWidgetSubtype.text.rawValue
```

```
// Create a button widget
widgetAnnotation.widgetFieldType = PDFAnnotationWidgetSubtype.button.rawValue
```

```
// Create a choice widget
widgetAnnotation.widgetFieldType = PDFAnnotationWidgetSubtype.choice.rawValue
```

Set Widget Control Types

Similarly, when creating a button widget annotation, the `widgetControlType` property is set to inform PDFKit which type of button to add:

```
// Create a radio button
widgetAnnotation.widgetControlType = .radioButtonControl
```

```
// Create a checkbox
widgetAnnotation.widgetControlType = .checkBoxControl
```

```
// Create a push button
widgetAnnotation.widgetControlType = .pushButtonControl
```

Choose Widget Control Properties

Toggles for the two kinds of choice widgets are set using the `isListChoice` property (list is the default):

```
// Create a list choice widget (default)
widgetAnnotation.isListChoice = true
```

```
// Create a combo box choice widget
widgetAnnotation.isListChoice = false
```

Add Widgets to the PDF Page

`ViewController` first loads a URL to the `MyForm.pdf` file through the app's main bundle. This URL is then used to instantiate a `PDFDocument`. On success, the document is assigned to the `PDFView` that was set up in Interface Builder. Once the document has been successfully loaded, The first page can be extracted so that widget annotations can be added.

`ViewController` adds the following widgets to the extracted `PDFPage`: three text fields, two radio buttons, three checkboxes, and one push button.

In addition to basic widget creation, `ViewController` includes a few extra widget-specific properties:

Text widget properties. The `textFieldDate` text widget annotation sets two extra properties: `maximumLength` and `hasComb`. The `maximumLength` property sets the maximum number of characters a user can enter into the text field. The `hasComb` property, if set, divides the text field into as many equally spaced positions, or combs, as the value of the maximum length of the field. The `hasComb` property only works in conjunction with the `maximumLength` property.

The `textFieldMultiline` text widget annotation sets one extra property: `isMultiline`. By default, all text widget annotations are single line, meaning they don't include word wrapping. If you want to create a large, multiline text widget with word wrapping, set this property to `true`.

Radio widget properties. The `radioButtonYes` and `radioButtonNo` button widget annotations set two extra properties: `fieldName` and `buttonWidgetStateString`. All widgets have their own unique identifier, which is determined by the `fieldName` property. To group widgets—in this case the radio buttons—the buttons must have the same field name, which is set in `ViewController`. To give radio buttons of the same field name their own unique subidentifier, the `buttonWidgetStateString` property must be set on each, which is also done in `ViewController`. When the radio buttons are grouped by field name, and each has its own unique `buttonWidgetStateString`, they behave like normal radio buttons (where selecting one deselects others in the group).

> **Note**
>
> The strings used for both `fieldName` and `buttonWidgetStateString` are arbitrary; what matters is that `fieldName` is the same for each button, and the `buttonWidgetStateString` strings for each button are unique to that grouping.

Button widget properties. The `resetButton` button widget annotation sets one extra property: `PDFActionResetForm`. With this property set, the given action is performed when the user clicks (or taps) this button. The `PDFActionResetForm` action is special in that it clears all widgets associated with it. `PDFActionResetForm` takes an array of strings that represent widget field names. The default behavior is to clear all fields included in that array. This behavior can be changed to clear all widgets not included in the array, by setting the `fieldsIncludedAreCleared` property to false. Because the behavior to clear all widgets isn't included in the list, and no field names were set, this action clears all widgets in the document.

See Also

Annotations

Adding Widgets to a PDF Document

Add text, button, and choice widgets to a PDF document.

Adding Custom Graphics to a PDF

Create and add custom annotation and page graphics to your PDF document.

{} Custom Graphics

Demonstrates adding a watermark to a PDF page.

`class` `PDFAnnotation`

An annotation in a PDF document.