HANSER

Erratum

Das Swift-Handbuch

von Thomas Sillmann

Print-ISBN: 978-3-446-45505-4
E-Book-ISBN: 978-3-446-45730-0
E-Pub-ISBN: 978-3-446-46107-9

Errata zu „Das Swift-Handbuch“

Hier finden Sie alle Korrekturen zu den Inhalten des Swift-Handbuchs. Die Errata wurden zum letzten Mal am 19.11.2019 aktualisiert.

Abschnitt 4.1: Integer

Seite 67

Bei einigen Wertebereichen in Tabelle 4.1 sind leider die zugehörigen Vorzeichen untergegangen. In der folgenden Tabelle finden Sie die korrekten Wertebereiche für die Typen `Int8`, `Int16`, `Int32` und `Int64`. Die Angaben für die verschiedenen `UInt`-Varianten sind korrekt.

Zu Tabelle 4.1: Korrektur der Angaben zu folgenden Integer-Typen in Swift

Integer-Typ	Größe und Wertebereich
`Int8`	8 Bit Signed Integer, Wertebereich -128 bis 127.
`Int16`	16 Bit Signed Integer, Wertebereich -32.768 bis 32.767.
`Int32`	32 Bit Signed Integer, Wertebereich -2.147.483.648 bis 2.147.483.647.
`Int64`	64 Bit Signed Integer, Wertebereich -9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807.

Abschnitt 4.4.2: Zusammenfügen von Strings

Seite 70

Die Konsolenausgabe am Ende von Listing 4.7 ist unvollständig. Statt

```
Mein Name ist Thomas
```

ist die Ausgabe

```
Mein Name ist Thomas Sillmann
```

korrekt.

Abschnitt 4.6.9: Sets miteinander vergleichen

Seite 90

In Listing 4.51 muss die zu Beginn erstellte Variable `firstValues` explizit als Set deklariert werden:

Listing 4.51 Vergleichen von Sets mittels isDisjoint(with:) (aktualisiert)

```
var firstValues: Set<Int> = [19, 99]
var secondValues = [9, 11, 19, 88, 99, 111]
var thirdValues = [8, 12, 70]
if firstValues.isDisjoint(with: secondValues) {
    print("firstValues und secondValues besitzen keine gemeinsamen Elemente.")
}
if firstValues.isDisjoint(with: thirdValues) {
    print("firstValues und thirdValues besitzen keine gemeinsamen Elemente.")
}
// firstValues und thirdValues besitzen keine gemeinsamen Elemente.
```

Abschnitt 7.2: Globale und lokale Variablen

Seite 186

Die Konsolenausgabe am Ende von Listing 7.20 ist nicht korrekt. Sie lautet stattdessen wie folgt (die Korrekturen sind fett formatiert):

```
…

Wert der globalen Variablen wird verändert.
Wert der globalen Variablen wurde verändert.
Wert der lokalen Variablen wird verändert.
Wert der lokalen Variablen wurde verändert.
```

Abschnitt 21.6.2: Einblenden neuer View-Controller

Seite 457

Die drei aufgeführten Methoden zum Einblenden neuer View-Controller tragen inzwischen eine andere Bezeichnung, siehe die folgende Tabelle:

Neue Methodennamen zum Einblenden von View-Controllern

Bisheriger Methodenname im Buch	Aktuelle Version
`presentViewControllerAsModalWindow(_:)`	`presentAsModalWindow(_:)`
`presentViewControllerAsSheet(_:)`	`presentAsSheet(_:)`
`presentViewController(_:asPopoverRelativeTo:of:preferredEdge:behavior:`	`present(_:asPopoverRelativeTo:of:preferredEdge:behavior:)`

Seite 463

Auch die Handhabung der auf Seite 463 beschriebenen IDs hat sich minimal geändert. Die beschriebenen Typen `NSStoryboard.Name` sowie `NSStoryboard.SceneIdentifier` sind einfache Typ-Aliase für den Typ `String`. So genügt es, direkt die gewünschte ID zu übergeben, ohne diese in einen der beschriebenen Typen zu packen. Das führt zu den in der folgenden Tabelle aufgeführten Code-Änderungen:

Erstellung von Storyboard-IDs

Bisheriges Verfahren im Buch	Aktuelle Version
`NSStoryboard.Name(rawValue: "Main")`	`"Main"`
`NSStoryboard.SceneIdentifier(rawValue: "Destination")`	`"Destination"`

Aufgrund dieser Änderungen folgt hier noch die aktualisierte Version von Listing 21.6 und Listing 21.7:

Listing 21.6 Erzeugen der destinationViewController-Property (aktualisiert)

```
private let destinationViewController: NSViewController = {
    let mainStoryboard = NSStoryboard(name: "Main", bundle: nil)
    let destinationViewController = mainStoryboard.instantiateController(withIdentifi
er: "Destination") as! NSViewController
    return destinationViewController
}()
```

Listing 21.7 Vollständige Implementierung der ViewController-Klasse zum Einblenden eines neuen View-Controllers (aktualisiert)

```
class ViewController: NSViewController {

    @IBOutlet weak var popoverButton: NSButton!

    private let destinationViewController: NSViewController = {
        let mainStoryboard = NSStoryboard(name: "Main", bundle: nil)
        let destinationViewController = mainStoryboard.instantiateController(with
Identifier: "Destination") as! NSViewController
        return destinationViewController
    }()

    @IBAction func presentModalWindow(_ sender: Any) {
        presentAsModalWindow(destinationViewController)
    }

    @IBAction func presentSheet(_ sender: Any) {
        presentAsSheet(destinationViewController)
    }

    @IBAction func presentPopover(_ sender: Any) {
        let popoverSize = NSRect(x: 0, y: 0, width: 300, height: 300)
        present(destinationViewController, asPopoverRelativeTo: popoverSize, of:
popoverButton, preferredEdge: .maxX, behavior: .semitransient)
    }

}
```

Abschnitt 21.6.3: Ausblenden eines View-Controllers

Seite 467

Die Methode zum Ausblenden eines View-Controllers heißt `dismiss(_:)`, nicht `dismissViewController(_:)`. Hier finden Sie zusätzlich die aktualisierte Version von Listing 21.8:

Listing 21.8 Ausblenden eines View-Controllers bei Betätigen eines Buttons (aktualisiert)

```
class DestinationViewController: NSViewController {

    @IBAction func dismiss(_ sender: Any) {
        dismiss(self)
    }

}
```

Abschnitt 24.4.3: Auf Ein- und Ausblenden des Keyboards reagieren

Seite 812

Die Namen der beschriebenen Notifications, die über das Ein- und Ausblenden der Bildschirmtastatur informieren, haben sich wie folgt geändert:

- `UIKeyboardWillShow` entspricht jetzt `UIResponder.keyboardWillShowNotification`
- `UIKeyboardDidShow` entspricht etzt `UIResponder.keyboardDidShowNotification`
- `UIKeyboardWillHide` entspricht jetzt `UIResponder.keyboardWillHideNotification`
- `UIKeyboardDidHide` entspricht jetzt `UIResponder.keyboardDidHideNotification`

Darüber hinaus lautet der Schlüssel zum Zugriff auf den Keyboard-Frame nicht länger `UIKeyboardFrameEndUserInfoKey`, sondern `UIResponder.keyboardFrameEndUserInfoKey`.

Hier die entsprechend aktualisierte Version von Listing 24.36:

Listing 24.36 Reaktion auf Ein- und Ausblenden der virtuellen Bildschirmtastatur (aktualisiert)

```
class ViewController: UIViewController {

    @IBOutlet weak var textView: UITextView!

    override func viewDidLoad() {
        super.viewDidLoad()
        NotificationCenter.default.addObserver(self, selector:#selector(keyboardDidSh
ow(_:)), name: UIResponder.keyboardDidShowNotification, object: nil)
        NotificationCenter.default.addObserver(self, selector:
#selector(keyboardDidHide(_:)), name: UIResponder.keyboardDidHideNotification, object:
nil)
    }

    @IBAction func dismissKeyboard() {
        textView.resignFirstResponder()

    }

    @objc private func keyboardDidShow(_ notification: Notification) {
        let endFrame = notification.userInfo![UIResponder.keyboardFrameEndUserInfoKey]
as! CGRect
        textView.contentInset.bottom = endFrame.size.height
        textView.scrollIndicatorInsets = textView.contentInset

    }
```

```
    @objc private func keyboardDidHide(_ notification: Notification) {
        textView.contentInset.bottom = 0
        textView.scrollIndicatorInsets = textView.contentInset

    }

    deinit {
        NotificationCenter.default.removeObserver(self, name: UIResponder.
keyboardDidShowNotification, object: nil)
        NotificationCenter.default.removeObserver(self, name: UIResponder.
keyboardDidHideNotification, object: nil)
    }

}
```