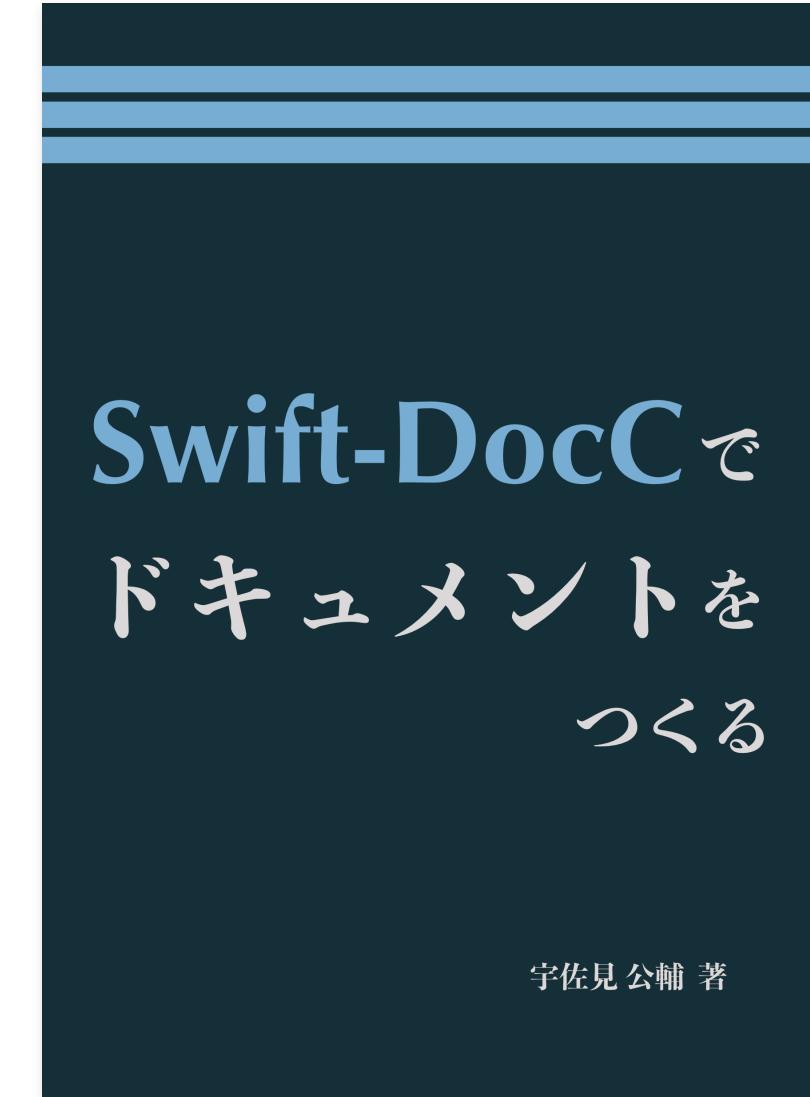


Swift-DocCやDokkaで
アプリのドキュメントを
書いてみる

自己紹介

- 宇佐見 公輔（うさみこうすけ） / @usamik26
- 株式会社ゆめみ / iOSテックリード
- 最近、本を書いたりiOSの記事を寄稿したりしています。



ドキュメント生成ツール

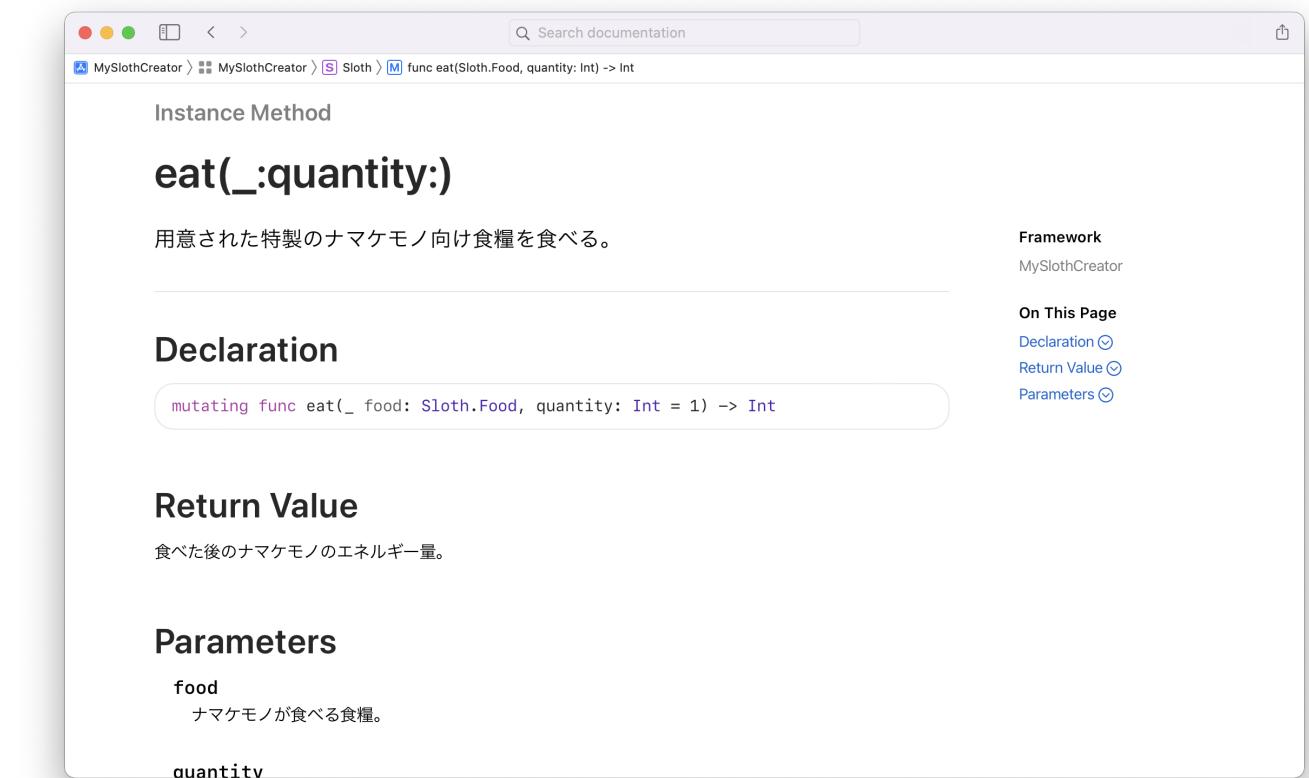
ドキュメント生成ツール

- iOS
 - Swift-DocC
 - <https://www.swift.org/documentation/docc/>
- Android
 - Dokka (+ KDoc)
 - <https://kotlinlang.org/docs/kotlin-doc.html>

Swift-DocC

- Swift-DocC : Swiftのドキュメント生成ツール、Xcodeに標準で組み込まれている
- ドキュメントコメントのシンタックスはMarkdownベース
- ドキュメントコメントの例：

```
/// 用意された特製のナマケモノ向け食糧を食べる。
///
/// - Parameters:
///   - food: ナマケモノが食べる食糧。
///   - quantity: ナマケモノが食べる食糧の数。
/// - Returns: 食べた後のナマケモノのエネルギー量。
mutating public func eat(_ food: Food, quantity: Int = 1) -> Int {
    energyLevel += food.energy * quantity
    return energyLevel
}
```



Dokka

- Dokka : Kotlinのドキュメント生成ツール、Gradleプラグインで使える
- KDoc : Kotlinのドキュメントコメントのシンタックス
- ドキュメントコメントの例：

```
/**  
 * A group of *members*.  
 *  
 * @param T the type of a member in this group.  
 * @property name the name of this group.  
 * @constructor Creates an empty group.  
 */  
class Group<T>(val name: String) {  
    /**  
     * Adds a [member] to this group.  
     * @return the new size of the group.  
     */  
    fun add(member: T): Int { ... }  
}
```

Swiftのガイドライン

- API Design Guidelines <https://www.swift.org/documentation/api-design-guidelines/>
- 最初にサマリーを書く
- たいていは宣言とサマリーだけ見れば理解できる

```
struct List {  
    /// Returns a `List` containing `head` followed by the elements  
    /// of `self`.  
    func prepending(_ head: Element) -> List  
}
```

Kotlinのガイドライン

- Coding conventions <https://kotlinlang.org/docs/coding-conventions.html>
- 一般に `@param` や `@return` タグの使用は避ける
- そのかわりに、パラメータと返り値がわかるようにドキュメントコメントを書く

```
/**  
 * Returns the absolute value of the given number.  
 * @param number The number to return the absolute value for.  
 * @return The absolute value.  
 */  
fun abs(number: Int): Int { /*...*/ }
```



```
/**  
 * Returns the absolute value of the given [number].  
 */  
fun abs(number: Int): Int { /*...*/ }
```

- 長い説明が必要な場合のみ `@param` や `@return` タグを使う

アプリのドキュメントに
ツールを活用する

ドキュメント生成ツールを使う場面

- ドキュメント生成ツールが活躍するのは、主にライブラリが多い
- 公開クラス、メソッドのドキュメントを生成する
- ライブラリを使う人のために、APIリファレンスが必要



- アプリではどうだろうか？
- ドキュメント生成自体は可能
- 使って有益な場面があるかどうか？

アプリ開発でドキュメントが欲しい場面

- ビルド方法や開発環境についてのドキュメント
- 設計について記述したドキュメント
- アプリ内部のユーティリティクラスやメソッドのドキュメント
- ユーザー目線でのアプリの仕様

どうドキュメントを作成するか

- ビルド方法や開発環境についてのドキュメント
 - コードのリポジトリのREADMEかそれに近い場所
- 設計について記述したドキュメント
 - これはどうする？
- アプリ内部のユーティリティクラスやメソッドのドキュメント
 - コード内のドキュメントコメント
- ユーザー目線でのアプリの仕様
 - コードとは別管理のドキュメント

設計ドキュメントをドキュメント生成ツールでカバー

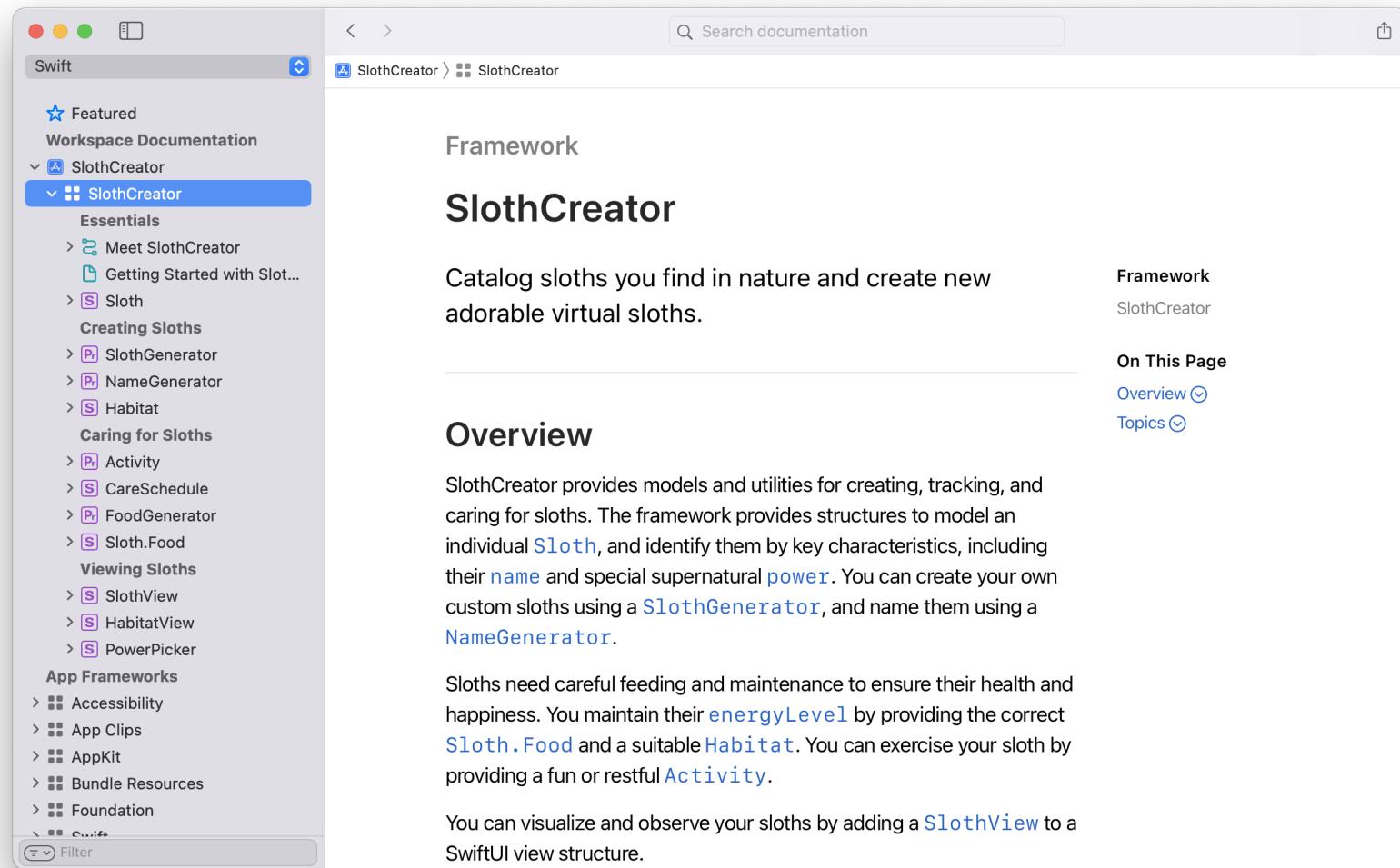
- 設計について記述したドキュメント
- コードのリポジトリに`docs` フォルダとか作ってMarkdownで書く？



- それなら、ドキュメント生成ツールを活用すると良いかも
- 実は、コメントからドキュメントを生成するだけでなく独立したMarkdownファイルを扱える

Swift-DocCの場合

- Documentationカタログ機能で、独立したMarkdownドキュメントを扱える
- Xcode上でDocumentationカタログを作成してMarkdownファイルを入れればよい



Dokkaの場合

- 設定オプションで、独立したMarkdownドキュメントを扱える
 - <https://kotlinlang.org/docs/kotlin-doc.html#module-and-package-documentation>
 - https://kotlin.github.io/dokka/1.6.10/user_guide/gradle/usage/#configuration-options
- 例えば`build.gradle`で以下のようにする

```
dokkaHtml.configure {  
    dokkaSourceSets {  
        includes.from("extra.md")  
    }  
}
```

設計ドキュメントをSwift-DocCやDokkaで扱う利点

- コードとドキュメントを同じ場所で管理できる
 - `docs` フォルダとかだとメンテナンスされなくなりがち
- リファレンスドキュメントとセットで生成できる
- 設計のドキュメントからアプリ内のクラスやメソッドに直接リンクできる

まとめ

- アプリのドキュメントにもSwift-DocCやDokkaが活用できる
 - アプリ内部のユーティリティクラスやメソッドのドキュメント
 - 設計について記述したドキュメント