Swift On Linux/Haupteil/Installation von Swift/

Aus SwiftWiki

Navigation

Inhaltsverzeichnis

Installation von Swift

Da Swift eine Sprache die vorrängig für MacOS erstellt wurde, ist es erst mit ein wenig Aufwand auf Linux Distributionen lauffähig. Dazu muss die Toolchain von Swift.org in ein Verzeichnis geladen werden, und je nach Wunsch in der Systemumgebung definiert werden. Eine genau Installationsanleitung kann ebenfalls unter Swift.org nachgelesen werden. Für die Code Beispiele ist der user gegen den tatsächlichen username auszutauschen. Um dies zu vereinfachen ist der Username unseres Beispielusers "user". Lade die aktuelle stabile Toolchain auf der Download Seite von Swift.org herunter und speicher es in einem beliebigen Verzeichnis z.B. im Downloads-Verzeichnis des aktuellen users. Danach muss clang installiert werden, um das compilieren von Swift Programmen zu ermöglichen. Dies ist mit dem Befehl auf einer Linux Bash möglich:

```
sudo apt-get install clang git libicu-dev libcurlpp
```

Danach muss die heruntergeladenen Swift Toolchain in ein geeignetes Verzeichnis extrahiert werden und der PATH Variable hinzugefügt werden: Dazu erstellen wir im ersten Schritt das Verzeichnis "install" im user Home-Verzeichnis

```
sudo mkdir /home/user/install
```

Entpacken der toolchain in das zuvor erstelle "intall" Verzeichnis:

```
sudo tar -xf /home/user/Downloads/swift-3.1.1-RELEASE-ubuntu16.10.tar.gz -C /home/user
/install
```

Um das ausführen von Swift leichter zu gestallten kann Swift der PATH-Variable temporär hinzugefügt werden:

```
export PATH=${PATH}:/home/user/install/swift-3.1.1-RELEASE-ubuntu16.10/usr/bin
```

Um dauerhaft, z.B. nach einem Neustart, Swift im Terminal zur Verfügung zu haben, kann der Pfad zur

Toolchain in die Datei /etc/environment eingetragen werden. Dazu wird die Datei environment mit einem Texteditor, in unserem Fall "VIM" geöffnet und erweitert:

```
sudo vim /etc/environment
```

und folgenden Text angefügen:

:/home/user/install/swift-3.1.1-RELEASE-ubuntu16.10/usr/bin

Danach sollte die Installation mit der Ausgabe der Version von Swift überprüft werden:

```
swift --version
```

welches die Ausgabe:

```
Swift version 3.1.1 (swift-3.1.1-RELEASE)
```

zur Folge hat. Eventuell muss die Besitzrechte des Swift-Installationsordner geändert werden, da eventuell Zugriffrechte nicht gegeben sind oder der root als Besitzer hinterlegt ist. Dazu wird folgender Befehl auf den Ordner angewand:
`sudo chown -R user:user ~/install/swift-3.1.1-RELEASE-ubuntu16.10/`

Damit ist die Installation abgeschlossen.

Swift REPL

Swift bietet bei dieser Installation ein Terminal Tool namens Swift-REPL (Swift-Read-eval-print-loop) welches es ermöglicht Swift direkt auf der Konsole zu test und auszuführen Da dies für das ausführen des Servers nicht notwendig ist, wird hier darauf nicht näher eingegangen, dazu aber jedenfalls mehr auf Swift.org .

Swift Package Manager

Der Package Manager ist eine Möglichkeit ein Skelton für ein Projekt zu erstellen. Der Package Manger erstellt einen Verzeichnisbaum der eine gewisse Grundstruktur ermöglicht. Diese Struktur wird auf Swift.org noch in der Version 3.0 beschrieben jedoch änderte sich dies mit dem RELEASE von Version 3.1. Zuerst wird ein Verzeichnis erstellt, dass das neue Projekt repräsentiert und in weiterer folge als Projektordner bezeichnet wird.

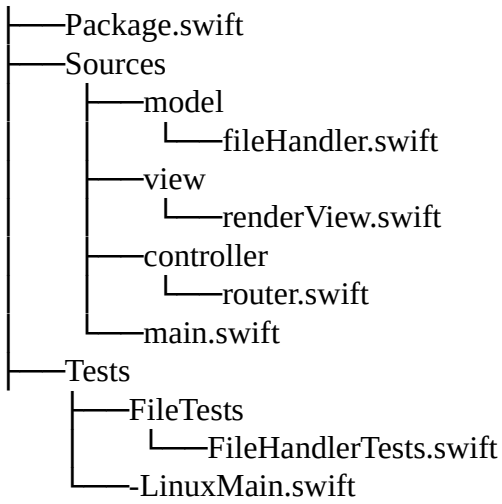
```
mkdir Hello
cd Hello
swift package init
```

Mit dem Ausführen des Init-Befehls wird unter anderem eine Package.swift Datei erstellt, in dem alle Abhängigkeiten des Projekt gespeichert werden. Weiter wir dein Verzeichnis Sources erstellt, in dem sich nur .swift-Datein befinden dürfen. Im Verzeichnis Tests befinden sich die Testklassen.

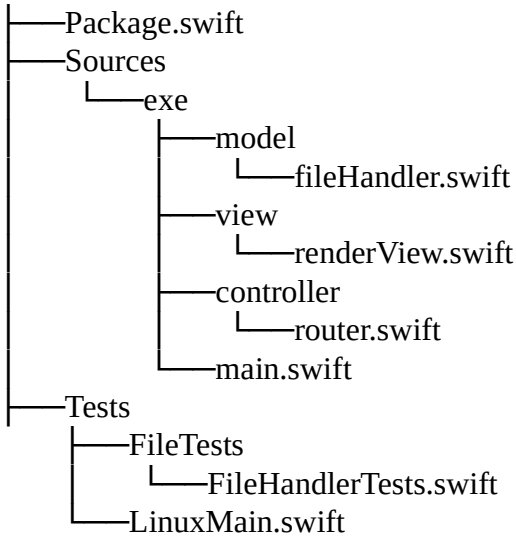
```
├──Package.swift
├──Sources
│   └──Hello.swift
├──Tests
    ├──HelloTests
    │   └──HelloTests.swift
    └──LinuxMain.swift
```

Sollte eine Strukturierung z.B. mit Hilfe von Ordner in Sources erfolgen, ist direkt unter dem Sources Verzeichnis ein weiteres Verzeichnis z.b. "exe" anzulegen, in dem die gewünschte Ordnerstruktur z.b. für ein MVC-Pattern erstellt werden kann. Es ist aufgrund des Compilers und der Modularität nicht möglich, mehrere Ordner oder Ordner und .swift-Dateien gemischt im Sources-Verzeichnis zu compilieren. Dies liegt daran, dass der Compiler Verzeichnisse als Module versteht und jeweils nur ein Modul erstellt werden kann. Daher ignoriert er oder compiliert entweder nur die Hello.swift oder den ersten gefunden Ordner.

falsche Struktur



richtige Struktur



Swift compiling

Der Swift Compiler kann mit dem Befehl

```
swift build
```

gestartet werden, dazu muss man sich im Projektordner befinden. Durch das aufrufen des Compilers werden die im Package.swift eingetragenen Abhängigkeiten z.B. von Github heruntergeladen und compiliert. Dabei wird das Verzeichnis ".build" erstellt in dem sich der ausführbare Code befindet. Erst danach wird der eigene produzierte Code compiliert und ebenfalls ins Verzeichnis "/.build/debug" gespeichert. Das compilieren von Abhängigkeiten erfolgt nur einmal nach dem Download oder dem Download von Aktualiesierungen. Die Packages werden im Ordnerbaum abgespeichert damit sie nicht jedesmal

compilieren heruntergeladen werden müssen. In der alten Version 3.0 wurde dazu ein eignes Verzeichnis "Packages" angelegt, und die aus Github ausgecheckten Module in Form von Source Code gespeichert. Seit dem Update auf die Version 3.1 wird diese Verzeichnis "Packages" im Überverzeichnis ".build" mit abgespeichert. Auch hier ist der Source Code der Module zu finden. Der Compiler ingnoriert beim compilieren grundsätlich alle Verzeichnisse ab dem Sources-Ordner was zur Folge hat, das Swift-Datein bzw. Klassen nicht wie in bekannt andern Sprachen z.B. mit einem "import"-Befehl zur Verfügung gestellt werden müssen, sondern automatisch als eine große Swift-Datei angesehen werden.

Der Swift Compiler unterstütz die Systemsprachen von C und Objective-C was durch den Clang Importer erreicht wird. Der Clang-Importer importiert Clang-Module, die die C und Objective-C APIs auf die Swift APIs mappen, wodurch das compilieren auf Linux bewerkstelligt wird.

Zurück zur Startseite des Buches