



Přibližný počet výsledků: 1 830 (0,35 s)

Metaprogramování je skupinou programovacích technik, které umožňují psaní programů, které vytváří další programy nebo s nimi nakládají jako se svými daty. Případně se může jednat o programy, které při svém běhu mohou dělat činnosti, které by jinak musely provádět již při kompilaci.

Metaprogramování – Wikipedie

<https://cs.wikipedia.org/wiki/Metaprogramování>



Informace o výsledku



Zpětná vazba

Metaprogramování – Wikipedie

<https://cs.wikipedia.org/wiki/Metaprogramování> ▼

Metaprogramování je skupinou programovacích technik, které umožňují psaní programů, které vytváří další programy nebo s nimi nakládají jako se svými daty.

[Způsoby metaprogramování](#) · [Statické jazyky](#) · [Příklady](#)

Příklady metaprogramování v C++ | Digitální knihovna VUT v Brně

<https://dspace.vutbr.cz/handle/11012/56108> ▼

Tato práce pojednává o **metaprogramování** v jazyce C++. Obsahuje přehledovou část zaměřenou na **metaprogramování** obecně a dále sadu příkladů ...

Programovací jazyk Julia: metaprogramování, makra a AST - Root.cz

<https://www.root.cz> › [Programovací jazyky](#) ▼

11. 8. 2016 - Sedmá část seriálu o jazyce Julia je věnována konceptu **metaprogramování** a tvorbě maker. V jazyce Julia lze pracovat přímo s parsovaným ...

mDevTalk – Setkání mobilních vývojářů

<https://www.mdevtalk.cz/> ▼

mDevTalk je série pravidelných setkání, na kterých si budeme ukazovat zdrojáky a vzájemně se posouvala dál. Každý druhý měsíc Tě čeká aktuální téma a ...



Seznámení se Sourcery

aneb Základy metaprogramování ve Swiftu

Jan Čislinský, Lead iOS Developer





Jan Čislinský
Lead iOS Developer



O čem to dnes bude

- Co to je metaprogramování a jeho výhody
- Představení Sourcery a jak s ním začít
- Jak na mockování dat
- Equatable a Hashable pro všechny
- Šablony zdarma
- Vliv na výkon



Co to je metaprogramování

*Metaprogramování je skupinou programovacích technik, které umožňují **psaní programů, které vytváří další programy** nebo s nimi nakládají jako se svými daty. Případně se může jednat o programy, které při svém běhu mohou dělat činnosti, které by jinak musely provádět již při kompilaci.*

— Wikipedia



Výhody

- Generování kódu často opakovaných patternů
equality, hashing, data persistence, json parsing, psaní a aktualizace mock dat
- Jednou napíšeme, jednou otestujeme, x-krát použijeme
- Jednodušší údržba šablony oproti dvaceti implementacím
- Vyhneme se lidským chybám





Sourcery

Nástroj umožňující generování type-safe kódu.

- Využívá Apple SourceKit.
- Generuje Swift kód buď inline, a nebo do vedlejšího souboru.
- Umožňuje pokročilou práci s anotacemi.



Jak začít

Jak začít

- Instalace Sourcery
- Vytvoření šablony & swift souboru
- Spuštění
`sourcery --sources ./ --templates ./Templates/ --output ./`
- 🤖

Podrobný postup



--watch

Démon, který sleduje změny šablon a zdrojových souborů a **automaticky spouští generování kódu.**



Koho to zaujalo?

**A teď si představte,
že píšete testy**



Znáte to, klasické TDD

- Před implementací začnete deklarací rozhraní.
- Napíšete si testy, které neprocházejí 🚨.
- Postupně doplňujete implementaci, aby se začalo zelenat ✅.
- S tím, jak se rozšiřuje implementace tak pořád dokola upravujete třídy, které používáte na mockování dat. 😡



Znáte to, klasické

Test
Driven
Development

- Před implementací začnete deklarací rozhraní.
- Napíšete si testy, které neprocházejí 🚨.
- Postupně doplňujete implementaci, aby se začalo zelenat ✅.
- S tím, jak se rozšiřuje implementace tak **pořád dokola** upravujete třídy, které používáte na mockování dat. 😡



Má to řešení

AutoMockable

AutoMockable

- Automaticky vygeneruje třídu, která implementuje oannotovaný protokol.
- Třída implementuje proměnné a funkce s podporou pro testování.
- Vygenerovaná třída se aktualizuje, jak se mění protokol.
- Anotace
// sourcing: AutoMockable

Equatable a Hashable

Každý z nás někdy potřeboval...

- **Equatable**
 - Pro porovnání identity dvou instancí.
- **Hashable**
 - Pro uložení instance do **Setu**.
 - Při použití instance jako klíče v **Dictionary**.
 - Pro porovnání hodnoty dvou instancí.






Swift 4.1

Automatic Synthesis*



Swift 4.1

Automatic Synthesis*





Swift 4.1

Automatic Synthesis*

*

- Pouze pro **struct** a **enum**.
- Pouze pokud všechny proměnné jsou **Equatable**, resp. **Hashable**, případně všechny associated values jsou **Equatable**, resp. **Hashable**.
- Nelze aplikovat na **extension**.





Swift 4.1

Automatic Synthesis*

Jak udělat

*

- Pouze pro `struct` a `enum`.
- Pouze pokud všechny proměnné jsou `Equatable`, resp. `Hashable`, případně všechny associated values jsou `Equatable`, resp. `Hashable`.
- Nelze aplikovat na `extension`.

Equatable & Hashable closure?



allyShow



AutoEquatable

- Stačí **class**, **enum**, **struct**, nebo **protocol** označit anotací (`// sourcery: AutoEquatable`), případně implementovat protokol **AutoEquatable** (nutná vlastní prázdná deklarace)
- Automaticky vygeneruje operátor `==`, ve kterém se porovnají všechny proměnné.



AutoEquatable

- Stačí **class**, **enum**, **struct**, nebo **protocol** označit anotací (`// sourcery: AutoEquatable`), případně implementovat protokol **AutoEquatable** (nutná vlastní prázdná deklarace)
- Automaticky vygeneruje operátor `==`, ve kterém se porovnají všechny proměnné.

Jak tohle řeší Equatable & Hashable closure?



AutoEquatable

- Stačí **class**, **enum**, **struct**, nebo **protocol** označit anotací (`// sourcery: AutoEquatable`), případně implementovat protokol **AutoEquatable** (nutná vlastní prázdná deklarace)
- Automaticky vygeneruje operátor `==`, který porovná všechny proměnné.
- Přeskočí proměnné označené anotací
`// sourcery: skipEquality`



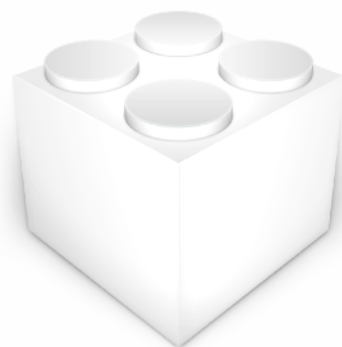
AutoHashable

- Stačí v **class**, **enum**, **struct**, nebo **protocolu** implementovat protokol **AutoHashable** (nutná vlastní prázdná deklarace)
- Automaticky vygeneruje **var hashValue: Int**, ve kterém se zkombinuje hashValue všech proměnných.
- Přeskočí proměnné označené anotací
// sourcery: skipHashing



Šablony zdarma

- AutoCases
- AutoMockable
- AutoCodable
- Diffable
- AutoEquatable
- Decorator
- AutoHashable
- LinuxMain
- AutoLenses



Bundled Templates



Koho to zaujalo?

Jaký to má vliv na build time?

Jaký to má vliv na build time?

zanedbatelný

Shrnutí

- Sourcing generuje type-safe kód při kompilaci
- Díky němu
 - Eliminujeme boilerplate kód
 - Jednou napíšeme, jednou otestujeme, x-krát použijeme
 - Vyhneme se lidským chybám
- Sourcing obsahuje řadu užitečných šablon
- Sourcing umožňuje psát vlastní šablony



Dotazy



Děkuji za pozornost!



Jan Čislinský
@jcislinsky



etnetera mobile

Etnetera a.s.

Jankovcova 1037/49
170 00 Praha 7

27. 9. 2018

16:00-18:00

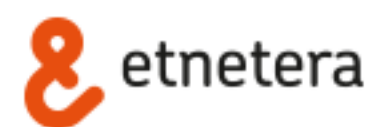
Workshop

Tvorba šablon pro Sourcery

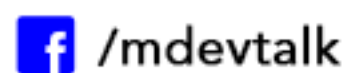
Registruj se do dnešních 23:59 h

<http://etn.cz/sourcery-registrace>

Workshop se **bude konat** pouze pokud se zaregistruje **min 10 zájemců**.



WWW.MDEVTALK.CZ



Odkazy

- [Prezentace i s ukázkovým projektem](#)
- [SourceryWorkshops](#)
- [Dokumentace](#)
- [Writing templates](#)
- [StencilSwiftKit](#)
- [Stencil: Built-in template tags and filters](#)
- [Krzystof Zablocki – Type-safe metaprogramming in Swift \(záznam\)](#)

