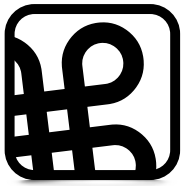


Macoun

Code-Generierung mit Sourcery

am Beispiel ObjectBox

— **Christian Tietze** —
@ctietze • christian@objectbox.io

Ablauf

1. Grundlagen: Code Gen Tools in 2018
2. Code Generierung für ObjectBox
3. Beispiel-App

1. Grundlagen

Warum Code Gen?

- Schlankes ObjectBox.framework **vs** invasives Core Data
- Compile-time **vs** Runtime Performance
- DRY

Eigener Code

Eigener Code

```
public let entityTypes = [
    PersonEntity.self,
    MonkeyEntity.self,
    BananaEntity.self,
    // ...

]
```

Eigener Code

```
public let entityTypes = [
{% for entity in types
    where entity.name|hasSuffix:"Entity" %}
    {{ entity.name }}.self{%
            if not forloop.last %},{% endif %}
{% endfor %}
]
```

Demo 00

Swift stdlib Code

Swift stdlib Code

```
/// Offers a String representation understandable
/// by the database
protocol DatabaseType {
    var dbTypeName: String { get }
}
```

```
    var dbTypeName: String { return "String" }
}

extension Date: DatabaseType {
    var dbTypeName: String { return "Date" }
}

extension Float: DatabaseType {
    var dbTypeName: String { return "Float" }
}

extension Double: DatabaseType {
    var dbTypeName: String { return "Double" }
}

extension Data: DatabaseType {
    var dbTypeName: String { return "Data" }
}

extension Int: DatabaseType {
    var dbTypeName: String { return "Int" }
}

extension Int64: DatabaseType {
    var dbTypeName: String { return "Int64" }
}

extension Int32: DatabaseType {
    var dbTypeName: String { return "Int32" }
}

extension Int16: DatabaseType {
    var dbTypeName: String { return "Int16" }
}

extension Bool: DatabaseType {

extension UInt: DatabaseType {
    var dbTypeName: String { return "

extension UInt64: DatabaseType {
    var dbTypeName: String { return "

extension UInt32: DatabaseType {
    var dbTypeName: String { return "

extension UInt16: DatabaseType {
    var dbTypeName: String { return "
```

Swift stdlib Code

```
{% set databaseTypeNames %}
Bool;Int8;Int16;Int32;Int64;Int;UInt8;UInt16;UInt32;
UInt64;UInt;Float;Double;Date;NSDate;Data;NSData
{% endset %}
```

Swift stdlib Code

```
{% for typeName in databaseTypeNames|split:";" %}
extension {{ typeName }}: DatabaseType {
    var dbTypeName: String { return "{{ typeName }}" }
}
{% endfor %}
```

Swift stdlib Code

```
extension String: DatabaseType {
    var dbTypeName: String { return "String" }
}

extension Date: DatabaseType {
    var dbTypeName: String { return "Date" }
}
```

Equatable

Equatable

```
extension Customer: Equatable {
    static func ==(lhs: Customer, rhs: Customer) -> Bool {
        guard lhs.firstName == rhs.firstName else { return false }
        guard lhs.lastName == rhs.lastName else { return false }
        guard lhs.birthDate == rhs.birthDate else { return false }
        return true
    }
}
```

Equatable

- `Swift.Equatable` generiert bereits viel Code
- Tests generieren?

AutoEquatable

AutoEquatable

- AutoEquatable: NSObject, Swift enums
- `==` ohne `Equatable` generieren
  - für Tests (non-production code)
  - Generics: `==(_ lhs: Self, _ rhs: Self)`

Demo 02

Tools in 2018

- SwiftGen
- Sourcery
- SourceKitten
- Stencil & StencilSwiftKit

SwiftGen

Mission: brüchtige String-based APIs obsolet machen

```
// List of all Assets
let image = NSImage(named: Asset.AssetBundleName.imageName)

// "Compiled" static JSON values
let port: Int = JSONFiles.serverConfiguration.port
```

Sourcery

Mission: Generierung repetetiven Codes

- AutoEquatable
- ObjectBox Datenbankadapter :)

SourceKitten

- Swift Code AST
- Benutzt u.a. von SwiftGen & Sourcery & SwiftLint
- Limitierung: "known types", keine Einsicht in Swift stdlib

Demo 09

Stencil / StencilSwiftKit

- Template-Sprache wie Mustache / Handlebars.js
- Swift Bibliothek, keine Skriptsprache verfügbar
- Filter-DSL, if-else, for-in-empty

```
{% if type.supertype|annotated:"AutoEquatable" %}
```

- Erweiterung via `sourcery --args`

2. Code Gen bei ObjectBox

ObjectBox.io

- Objekt-orientierte Datenbank
- Superschneller Kern: flatbuffers, memcpy

OBJECTBOX

Swift Binding

Idealvorstellung: *Plain Old Swift Objects*

```
class Person {
    let name: String
    let age: Int
    let address: Address
}
```

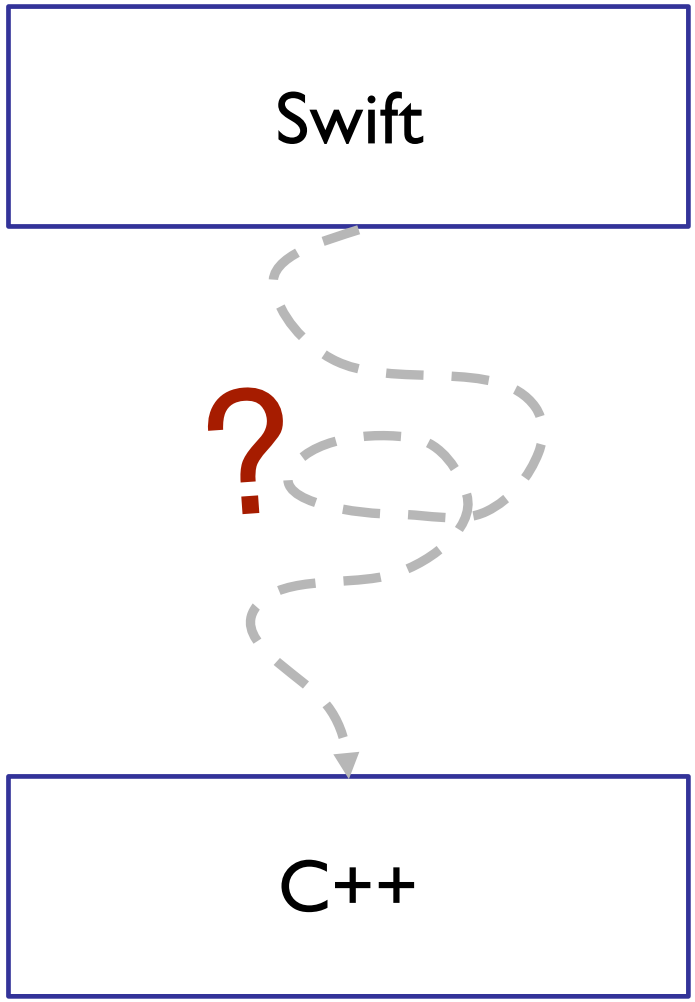
```
struct Address {
    let street: String
    let city: String
}
```

Problem:

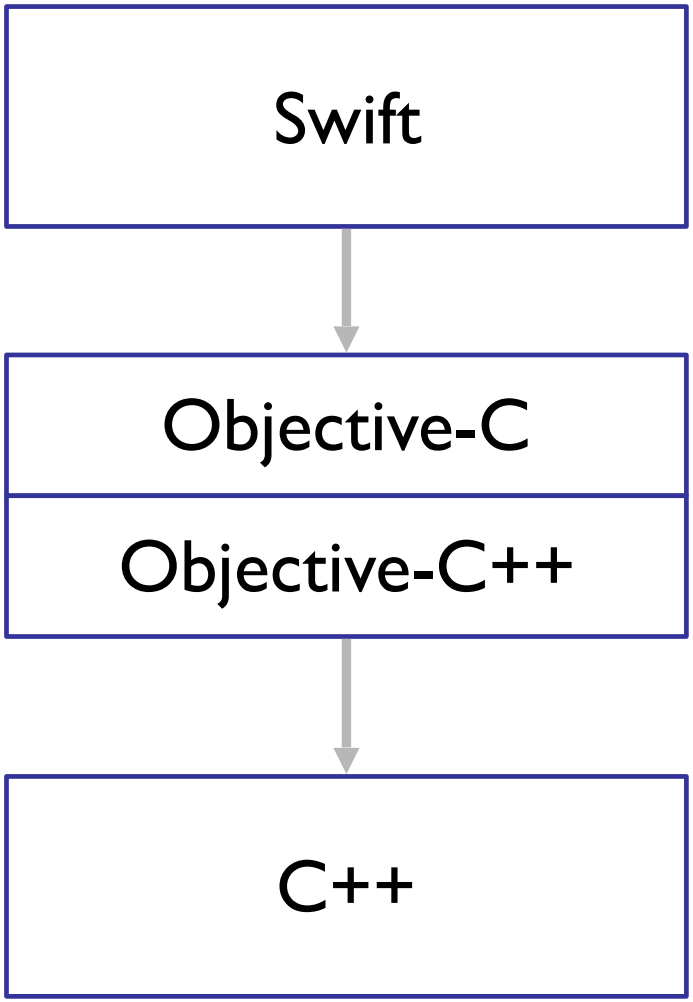
Wie erfährt die Bibliothek von den Typen, um sie zu speichern?

```
class Monkey {
    var name: String
    var age: Int
}
```

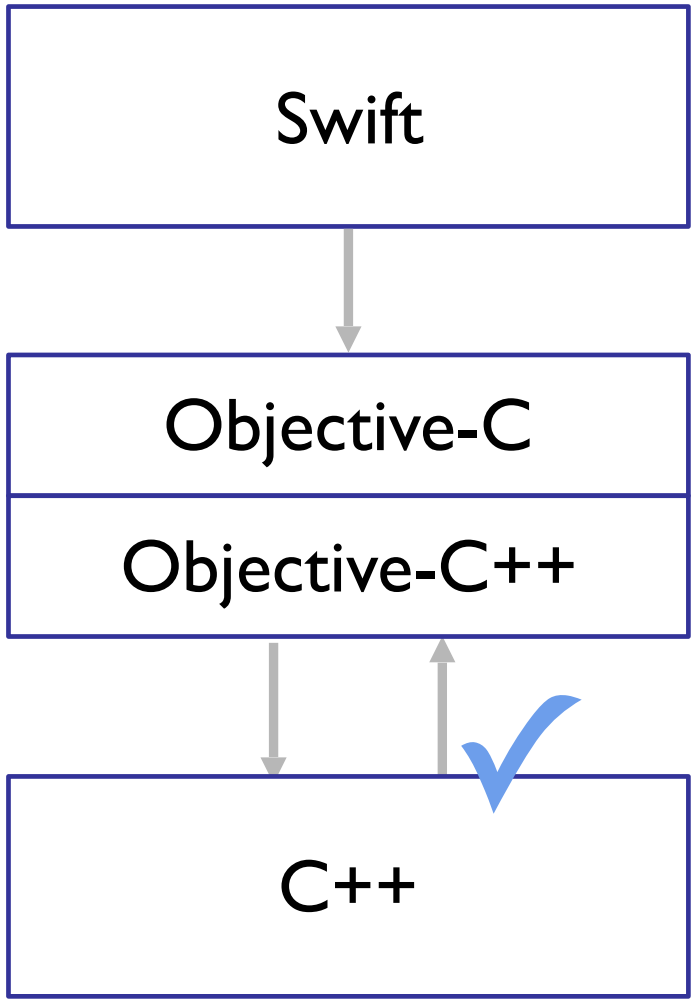
```
class Monkey {
    var name: String
    var age: Int
}
```



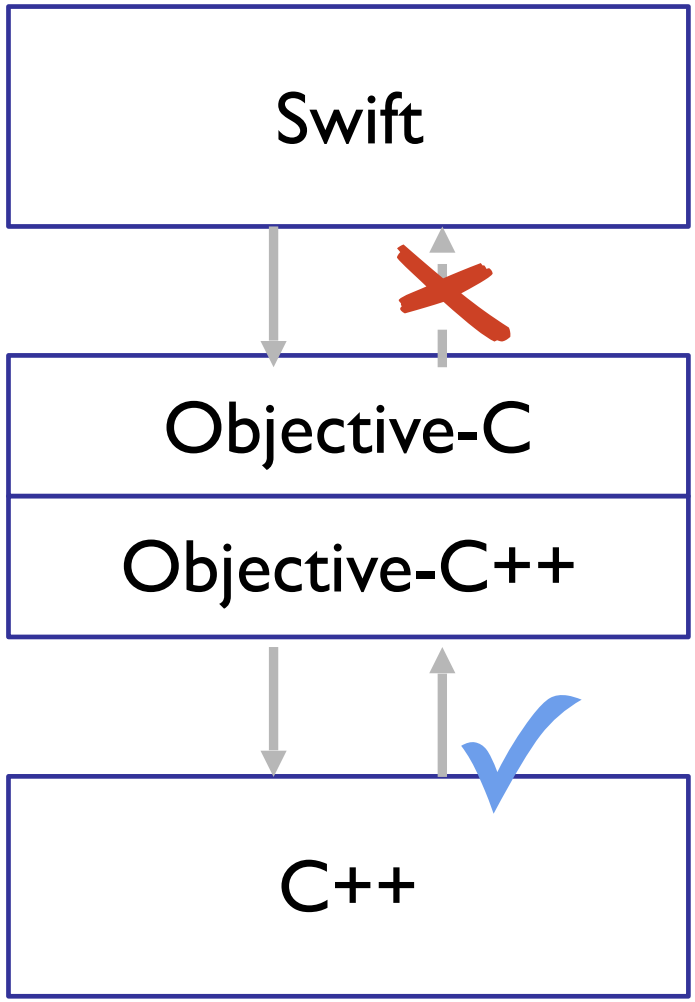
```
class Monkey {
    var name: String
    var age: Int
}
```



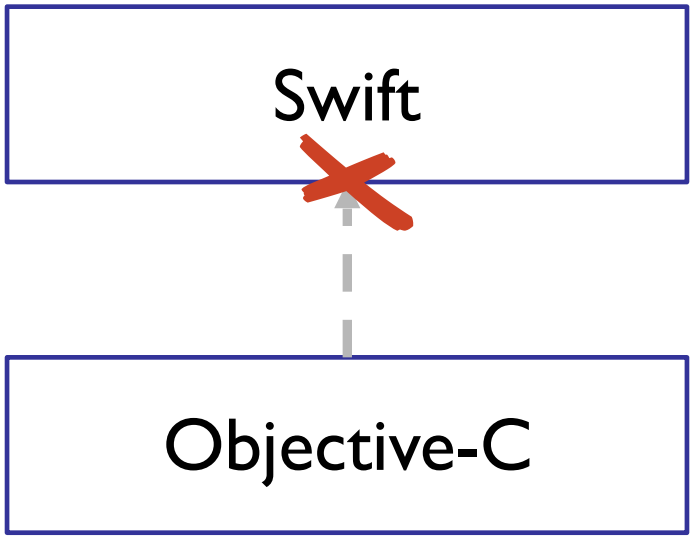
```
class Monkey {
    var name: String
    var age: Int
}
```

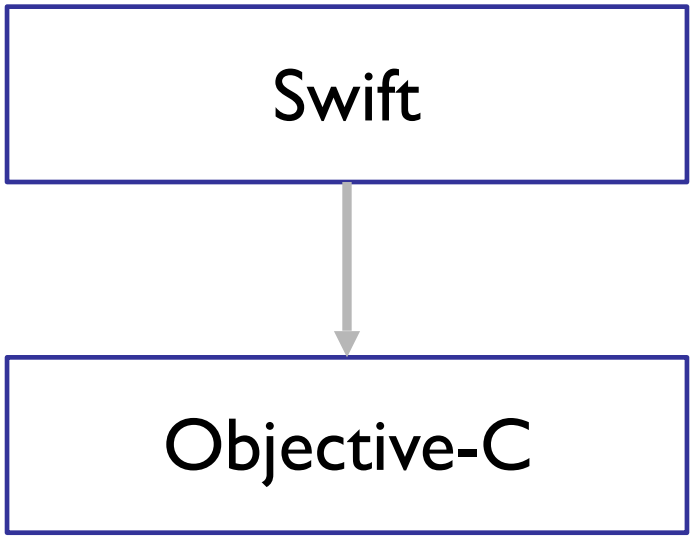


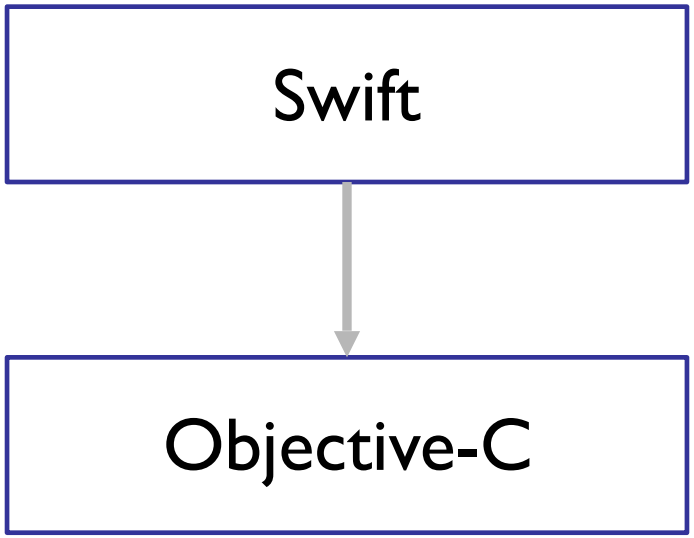
```
class Monkey {
    var name: String
    var age: Int
}
```



```
@objc class Monkey: NSObject {
    @objc dynamic var name: String
    @objc dynamic var age: Int
}
```

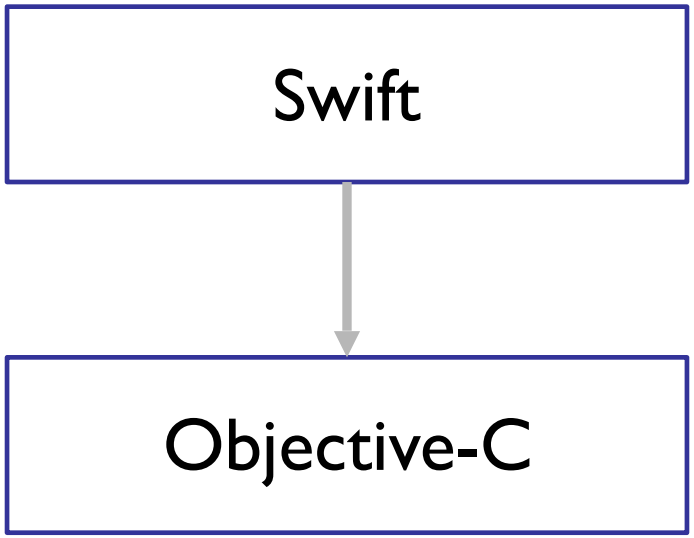

Swift
Objective-C


Swift
Objective-C




```
@interface EntityStore
```

```
entityStore.register(Monkey.entityInfo)
```




```
@interface EntityStore
```

Problem:

Wie erfährt Sourcery, für welche Typen die Registrierung generiert werden soll?

```
class Monkey {
    var name: String
    var age: Int
}
```

@Entity // Java annotations

```
class Monkey {
    var name: String
    var age: Int
}
```

```
// sourcery: ObjectBoxEntity
class Monkey {
    var name: String
    var age: Int
}
```

Demo 10

```
class Monkey: Entity {
    var name: String
    var age: Int
}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  static var entityInfo: EntityInfo {
    return EntityInfo(
      name: "{{ type.name }}",
      cursorClass: {{ type.name }}Cursor.self)
  }
}
{% endfor %}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  static var entityInfo: EntityInfo {
    return EntityInfo(
      name: "{{ type.name }}",
      cursorClass: {{ type.name }}Cursor.self)
  }
}
{% endfor %}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  static var entityInfo: EntityInfo {
    return EntityInfo(
      name: "{{ type.name }}",
      cursorClass: {{ type.name }}Cursor.self)
  }
}
{% endfor %}
```

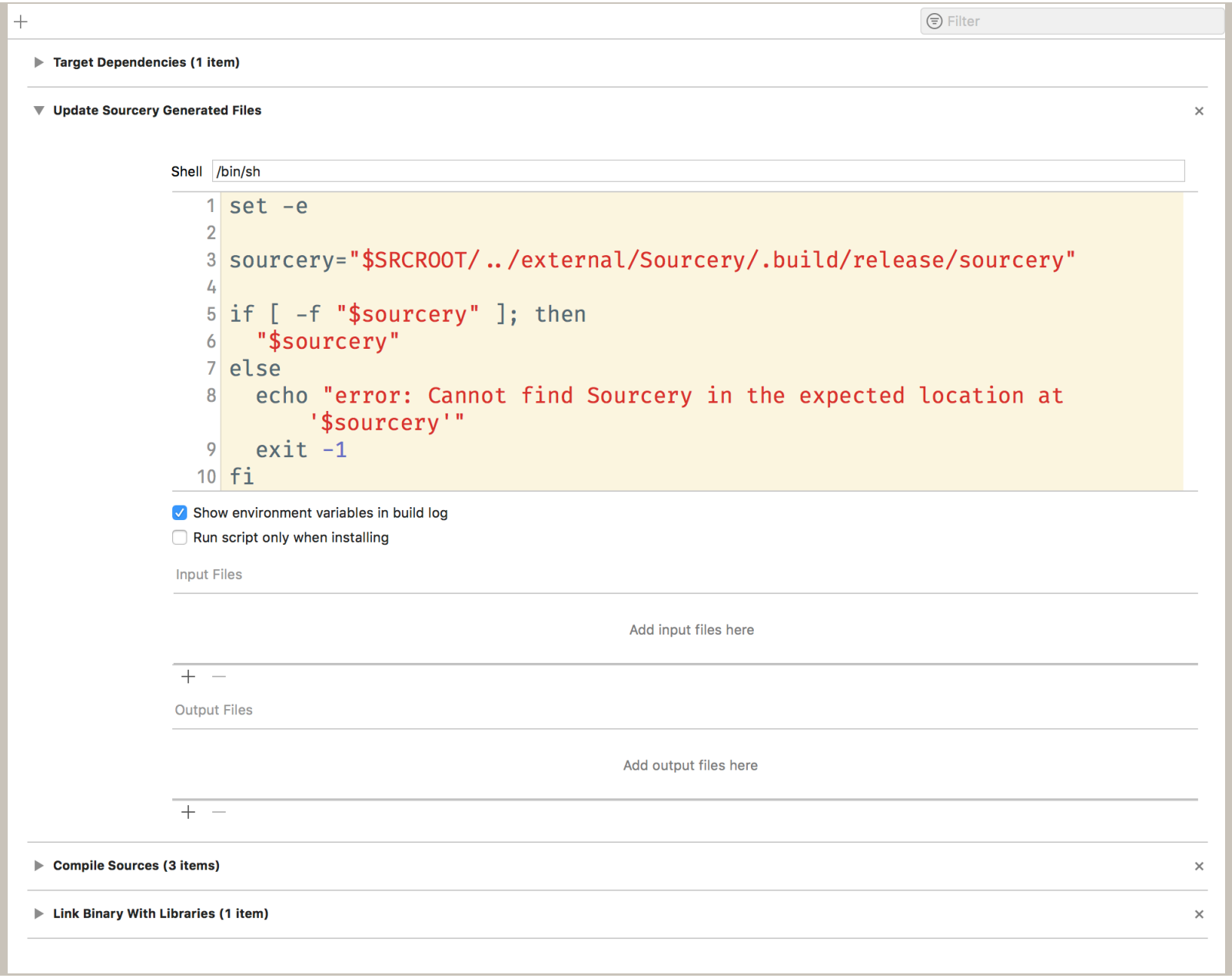
```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  static var entityInfo: EntityInfo {
    return EntityInfo(
      name: "{{ type.name }}",
      cursorClass: {{ type.name }}Cursor.self)
  }
}
{% endfor %}
```

```
extension Monkey {
  static var entityInfo: EntityInfo {
    return EntityInfo(
      name: "Monkey",
      cursorClass: MonkeyCursor.self)
  }
}
```

Demo 11 & 12

Sourcery als Build Tool

- Schnittstellen-Protokolle für App-Entities in ObjectBox
- Schneller als runtime reflection, aber hässlich zu coden
- Boilerplate-Code generieren lassen


Filter
Target Dependencies (1 item)
Update Sourcery Generated Files
Shell /bin/sh
1 set -e
2
3 sourcery="$SRCROOT/../external/Sourcery/.build/release/sourcery"
4
5 if [-f "$sourcery"]; then
6 "$sourcery"
7 else
8 echo "error: Cannot find Sourcery in the expected location at '$sourcery'"
9 exit -1
10 fi
Show environment variables in build log
Run script only when installing
Input Files
Add input files here
Output Files
Add output files here
Compile Sources (3 items)
Link Binary With Libraries (1 item)

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    {% for prop in type.storedVariables %}
    entityBuilder.addPropertyNamed("{{ prop.name }}",
        type: {{ prop.typeName }}.entityPropertyType)
    {% endfor %}
  }
}
{% endfor %}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    {% for prop in type.storedVariables %}
    entityBuilder.addPropertyNamed("{{ prop.name }}",
        type: {{ prop.typeName }}.entityPropertyType)
    {% endfor %}
  }
}
{% endfor %}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    {% for prop in type.storedVariables %}
    entityBuilder.addPropertyNamed("{{ prop.name }}",
        type: {{ prop.typeName }}.entityPropertyType)
    {% endfor %}
  }
}
{% endfor %}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    {% for prop in type.storedVariables %}
    entityBuilder.addPropertyNamed("{{ prop.name }}",
        type: {{ prop.typeName }}.entityPropertyType)
    {% endfor %}
  }
}
{% endfor %}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    {% for prop in type.storedVariables %}
    entityBuilder.addPropertyNamed("{{ prop.name }}",
        type: {{ prop.typeName }}.entityPropertyType)
    {% endfor %}
  }
}
{% endfor %}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    {% for prop in type.storedVariables %}
    entityBuilder.addPropertyNamed("{{ prop.name }}",
        type: {{ prop.typeName }}.entityPropertyType)
    {% endfor %}
  }
}
{% endfor %}
```

```
{% for type in types.based.Entity %}
extension {{ type.name }} {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    {% for prop in type.storedVariables %}
    entityBuilder.addPropertyNamed("{{ prop.name }}",
        type: {{ prop.typeName }}.entityPropertyType)
    {% endfor %}
  }
}
{% endfor %}
```

```
extension Monkey {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    entityBuilder.addPropertyNamed("name",
        type: String.entityPropertyType)
    entityBuilder.addPropertyNamed("age",
        type: Int.entityPropertyType)
  }
}
```

```
extension Monkey {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    entityBuilder.addPropertyNamed("name",
        type: String.entityPropertyType)
    entityBuilder.addPropertyNamed("age",
        type: Int.entityPropertyType)
  }
}
```

```
extension Monkey {
  fileprivate static func buildEntity(modelBuilder: ModelBuilder) {
    let entityBuilder = modelBuilder.entityBuilder(for: entityInfo)
    entityBuilder.addPropertyNamed("id", type: .long, flags: [.id])
    entityBuilder.addPropertyNamed("name",
        type: .string)
    entityBuilder.addPropertyNamed("age",
        type: .int)
  }
}
```

```swift
extension Int32: EntityPropertyTypeConvertible {
    public static var entityPropertyType: EntityPropert { return .int }
}

extension Int64: EntityPropertyTypeConvertible {
    public static var entityPropertyType: EntityPr
}

extension Int: EntityPropertyTypeConvertible {
    public static var entityPropertyType: EntityPropertyType { retu
}

extension Float: EntityPropertyTypeConvertible {
    public static var entityPropertyType: EntityPropertyType { return .float }
}

extension Double: EntityPropertyTypeConvertible {
    public static var entityPropertyType: EntityPropertyType { return .double }
}

extension String: EntityPropertyTypeConvertible {
    public static var entityPropertyType: EntityPropertyType { return .string }
}
```

2. Beispiel-App

Projektvorlage

```
- PROJECT_DIRECTORY/
  +-- .sourcery.yml
  +-- External/
  |   +-- Sourcery/
  +-- TARGET_NAME/
  +-- TARGET_NAME-generated/
  +-- templates/
```

Sourcery Projektkonfiguration

```
project:
  file: EntityInfoTest.xcodeproj
  target:
    name: EntityInfoTest
    module: EntityInfoTest
templates:
  - ./templates/
output:
  path: ./EntityInfoTest-generated/
```

Objekterzeugung & known types

Objekterzeugung & known types

```
func createEntity(entityReader: EntityReader) -> Any {
    let entity = {{ type.name }}()
    {% for prop in type.storedVariables %}
        {% set objectBoxTypeName %}{{ prop.typeName.name }};{% endset %}
        {% if objectBoxTypeScalarNames|contains:objectBoxTypeName
                or prop.typeName.name == "String" %}
            entity.{{ prop.name }} = entityReader.read(at: /* offset calculation */)
        {% endif %}
    {% endfor %}
    return entity
}
```

Objekterzeugung & known types

```
func createEntity(entityReader: EntityReader) -> Any {
    let entity = {{ type.name }}()
    {% for prop in type.storedVariables %}
        {% set objectBoxTypeName %}{{ prop.typeName.name }};{% endset %}
        {% if objectBoxTypeScalarNames|contains:objectBoxTypeName
                or prop.typeName.name == "String" %}
            entity.{{ prop.name }} = entityReader.read(at: /* offset calculation */)
        {% endif %}
    {% endfor %}
    return entity
}
```

Objekterzeugung & known types

```
func createEntity(entityReader: EntityReader) -> Any {
    let entity = {{ type.name }}()
    {% for prop in type.storedVariables %}
        {% set objectBoxTypeName %}{{ prop.typeName.name }};{% endset %}
        {% if objectBoxTypeScalarNames|contains:objectBoxTypeName
                or prop.typeName.name == "String" %}
            entity.{{ prop.name }} = entityReader.read(at: /* offset calculation */)
        {% endif %}
    {% endfor %}
    return entity
}
```

Objekterzeugung & known types

```
func createEntity(entityReader: EntityReader) -> Any {
    let entity = {{ type.name }}()
    {% for prop in type.storedVariables %}
        {% set objectBoxTypeName %}{{ prop.typeName.name }};{% endset %}
        {% if objectBoxTypeScalarNames|contains:objectBoxTypeName
                or prop.typeName.name == "String" %}
            entity.{{ prop.name }} = entityReader.read(at: /* offset calculation */)
        {% endif %}
    {% endfor %}
    return entity
}
```

Objekterzeugung & known types

```
func createEntity(entityReader: EntityReader) -> Any {
    let entity = {{ type.name }}()
    {% for prop in type.storedVariables %}
        {% set objectBoxTypeName %}{{ prop.typeName.name }};{% endset %}
        {% if objectBoxTypeScalarNames|contains:objectBoxTypeName
              or prop.typeName.name == "String" %}
            entity.{{ prop.name }} = entityReader.read(at: /* offset calculation */)
        {% endif %}
    {% endfor %}
    return entity
}
```

Makros

- Nur EJS und Swift Templates erlauben *helper*
- StencilSwiftKit bietet Makros für Stencil an
- Eigene "where" Filter nur durch Kompilieren möglich
- Einrückung im Makro gewinnt :(

```
{% macro EntityIdGetter objName type %}
{% for var in type.storedVariables|annotated:"entityId" %}
    return {{ objName }}.{{ var.name }}
{% empty %}
    {% set idTypeName %}Id<{{ type.name }}>{% endset %}
    {% for var in type.storedVariables
            where var.name == "id"
            and var.typeName.name == idTypeName %}
        return {{ objName }}.{{ var.name }}
    {% empty %}
        {% for var in type.storedVariables
                where var.typeName.name == idTypeName %}
            return {{ objName }}.{{ var.name }}
        {% empty %}
            #error("Implement an {{idTypeName}} property somewhere")
        {% endfor %}
    {% endfor %}
{% endfor %}
{% endmacro %}
```

```
{% macro EntityIdGetter objName type %}
{% for var in type.storedVariables|annotated:"entityId" %}
    return {{ objName }}.{{ var.name }}
{% empty %}
    {% set idTypeName %}Id<{{ type.name }}>{% endset %}
    {% for var in type.storedVariables
            where var.name == "id"
            and var.typeName.name == idTypeName %}
        return {{ objName }}.{{ var.name }}
    {% empty %}
        {% for var in type.storedVariables
                where var.typeName.name == idTypeName %}
            return {{ objName }}.{{ var.name }}
        {% empty %}
            #error("Implement an {{idTypeName}} property somewhere")
        {% endfor %}
    {% endfor %}
{% endfor %}
{% endmacro %}
```

```
{% macro EntityIdGetter objName type %}
{% for var in type.storedVariables|annotated:"entityId" %}
    return {{ objName }}.{{ var.name }}
{% empty %}
    {% set idTypeName %}Id<{{ type.name }}>{% endset %}
    {% for var in type.storedVariables
            where var.name == "id"
            and var.typeName.name == idTypeName %}
        return {{ objName }}.{{ var.name }}
    {% empty %}
        {% for var in type.storedVariables
                where var.typeName.name == idTypeName %}
            return {{ objName }}.{{ var.name }}
        {% empty %}
            #error("Implement an {{idTypeName}} property somewhere")
        {% endfor %}
    {% endfor %}
{% endfor %}
{% endmacro %}
```

```
{% macro EntityIdGetter objName type %}
{% for var in type.storedVariables|annotated:"entityId" %}
    return {{ objName }}.{{ var.name }}
{% empty %}
    {% set idTypeName %}Id<{{ type.name }}>{% endset %}
    {% for var in type.storedVariables
            where var.name == "id"
            and var.typeName.name == idTypeName %}
        return {{ objName }}.{{ var.name }}
    {% empty %}
        {% for var in type.storedVariables
                where var.typeName.name == idTypeName %}
            return {{ objName }}.{{ var.name }}
        {% empty %}
            #error("Implement an {{idTypeName}} property somewhere")
        {% endfor %}
    {% endfor %}
{% endfor %}
{% endmacro %}
```

```
{% macro EntityIdGetter objName type %}
{% for var in type.storedVariables|annotated:"entityId" %}
    return {{ objName }}.{{ var.name }}
{% empty %}
    {% set idTypeName %}Id<{{ type.name }}>{% endset %}
    {% for var in type.storedVariables
            where var.name == "id"
            and var.typeName.name == idTypeName %}
        return {{ objName }}.{{ var.name }}
    {% empty %}
        {% for var in type.storedVariables
                where var.typeName.name == idTypeName %}
            return {{ objName }}.{{ var.name }}
        {% empty %}
            #error("Implement an {{idTypeName}} property somewhere")
        {% endfor %}
    {% endfor %}
{% endfor %}
{% endmacro %}
```

```
{% macro EntityIdGetter objName type %}
{% for var in type.storedVariables|annotated:"entityId" %}
    return {{ objName }}.{{ var.name }}
{% empty %}
    {% set idTypeName %}Id<{{ type.name }}>{% endset %}
    {% for var in type.storedVariables
            where var.name == "id"
            and var.typeName.name == idTypeName %}
        return {{ objName }}.{{ var.name }}
    {% empty %}
        {% for var in type.storedVariables
                where var.typeName.name == idTypeName %}
            return {{ objName }}.{{ var.name }}
        {% empty %}
            #error("Implement an {{idTypeName}} property somewhere")
        {% endfor %}
    {% endfor %}
{% endfor %}
{% endmacro %}
```

```
{% macro EntityIdGetter objName type %}
{% for var in type.storedVariables|annotated:"entityId" %}
    return {{ objName }}.{{ var.name }}
{% empty %}
    {% set idTypeName %}Id<{{ type.name }}>{% endset %}
    {% for var in type.storedVariables
            where var.name == "id"
            and var.typeName.name == idTypeName %}
        return {{ objName }}.{{ var.name }}
    {% empty %}
        {% for var in type.storedVariables
                where var.typeName.name == idTypeName %}
            return {{ objName }}.{{ var.name }}
        {% empty %}
            #error("Implement an {{idTypeName}} property somewhere")
        {% endfor %}
    {% endfor %}
{% endmacro %}
```

Live-Demo in Xcode 🎉

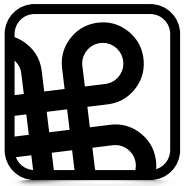
Fragen?

V I E L E N D A N K !

OBx.frmwrk https://objectbox.io/ios

email christian@objectbox.io
Twitter @ctietze
Web https://christiantietze.de

Zeit	Großer Saal	Terrassensaal	hier
11:00			Du
12:30	Schon mal an CloudKit gedacht?	Voice Commands mit Siri in iOS 12	Moderne App-Architektur
13:30 – 15:00	Mittagspause		
15:00	Reverse-Engineering mit Hopper	Dynamisches Swift	Typographie in Apps
16:30	Datenschutz-grundverordnung		



Macoun