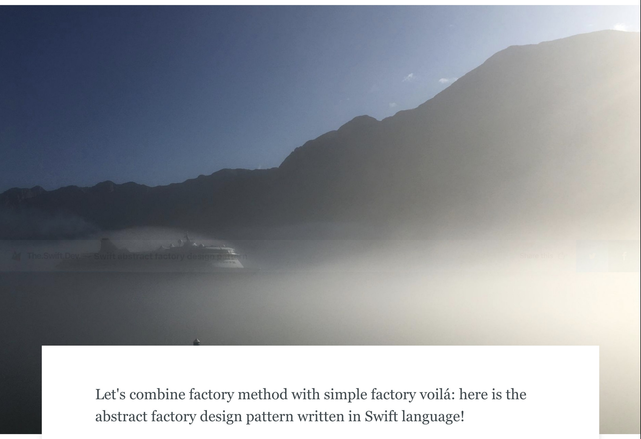
4 JUNE 2018 / DESIGN PATTERNS

Swift abstract factory design pattern



Let's combine factory method with simple factory voilá: here is the abstract factory design pattern written in Swift language!

Abstract factory in Swift

> *The abstract factory pattern provides a way to encapsulate a group of individual factories that have a common theme without specifying their concrete classes.*

So abstract factory is there for you to create families of related objects. The implementation usually combines simple factory & factory method principles. Individual objects are created through factory methods, while the whole thing is wrapped in an "abstract" simple factory. Now check the code! 😅

AbstractFactory.swift 1.19 KB on GitLab

```swift
// service protocols
protocol ServiceFactory {
    func create() -> Service
}

protocol Service {
    var url: URL { get }
}

// staging
class StagingService: Service {
    var url: URL { return URL(string: "https://dev.localhost/")! }
}

class StagingServiceFactory: ServiceFactory {
    func create() -> Service {
        return StagingService()
    }
}

// production
class ProductionService: Service {
    var url: URL { return URL(string: "https://live.localhost/")! }
}

class ProductionServiceFactory: ServiceFactory {
    func create() -> Service {
        return ProductionService()
    }
}

// abstract factory
class AppServiceFactory: ServiceFactory {

    enum Environment {
        case production
        case staging
    }

    var env: Environment

    init(env: Environment) {
        self.env = env
    }

    func create() -> Service {
        switch self.env {
        case .production:
            return ProductionServiceFactory().create()
        case .staging:
            return StagingServiceFactory().create()
        }
    }
}

let factory = AppServiceFactory(env: .production)
let service = factory.create()
print(service.url)
```

As you can see using an abstract factory will influence the whole application logic, while factory methods have effects only on local parts. Implementation can vary for example you could also create a standalone protocol for the abstract factory, but in this example I wanted to keep things as simple as I could.

Abstract factories are often used to achieve object independence. For example if you have multiple different SQL database connectors (PostgreSQL, MySQL, etc.) written in Swift with a common interface, you could easily switch between them anytime using this pattern. Same logic could be applied for anything with a similar scenario. 🤔

External sources

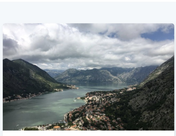
- Abstract factory pattern
- The Abstract Factory
- Differences between Abstract Factory Pattern and Factory Method

Tibor Bödecs
Read more posts by this author.
Read More

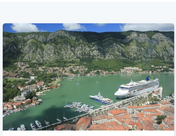
— The.Swift.Dev. —
Design patterns

Lazy initialization in Swift
Swift object pool design pattern
Swift facade design pattern

See all 18 posts →



DESIGN PATTERNS
Comparing factory design patterns
Learn what's the difference between static factory, simple factory, factory method and abstract factory using the Swift language.
1 MIN READ



DESIGN PATTERNS
Swift factory method design pattern
The factory method design pattern is a dedicated non-static method for hiding the creation logic of an object. Let's make it in Swift!
1 MIN READ