31 MAY 2018 / 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!

## Factory method is just a non-static method

Let's face it, this pattern is just a method usually backed by simple protocols & classes. Start with a really simple example: imagine a class that can create a base URL for your service endpoint. Let's call it service factory. 😅

FactoryMethod.swift 187 Bytes on GitLab

```
class ServiceFactory {
    func createProductionUrl() -> URL {
        return URL(string: "https://localhost/")!
    }
}
let factory = ServiceFactory()
factory.createProductionUrl()
```

You might think, that hey, this is not even close to a factory method pattern, but wait for it... let's make things a little bit complicated by creating a protocol for the service class and a protocol for returning the url as well. Now we can implement our base production url protocol as a separate class and return that specific instance from a production service factory class. Just check the code you'll get it:

FactoryMethod.swift 420 Bytes on GitLab

```
protocol ServiceFactory {
    func create() -> Service
}

protocol Service {
    var url: URL { get }
}

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

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

let factory = ProductionServiceFactory()
let request = factory.create()
```

Why did we separated all the logic into two classes and protocols? Please believe me decoupling is a good thing. From now on you could easily write a mocked service with a dummy url to play around with. Obviously that'd need a matching factory class.

Those mock instances would also implement the service protocols so you could add new types in a relatively painless way without changing the original codebase. The factory method solves one specific problem of a simple factory pattern. If the list - inside the switch-case - becomes too long, maintaining new objects will be hell with just one factory. Factory method solves this by introducing multiple factory objects.

## External sources

- Factory Method in Swift
- Swift World: Design Patterns—Factory Method
- Factory Pattern. When to use factory methods?



**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

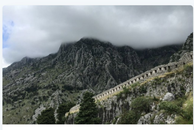
See all 18 posts →

DESIGN PATTERNS

**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!

1 MIN READ



DESIGN PATTERNS

**Swift simple factory design pattern**

This time let's talk about the simple factory design pattern to encapsulate object creation in a really simple way using Swift.

1 MIN READ