How to build a debug view (almost) for free

Vincent Pradeilles (@v_pradeilles) & Benoît Caron – Worldline 🇫🇷

Why?

Why?

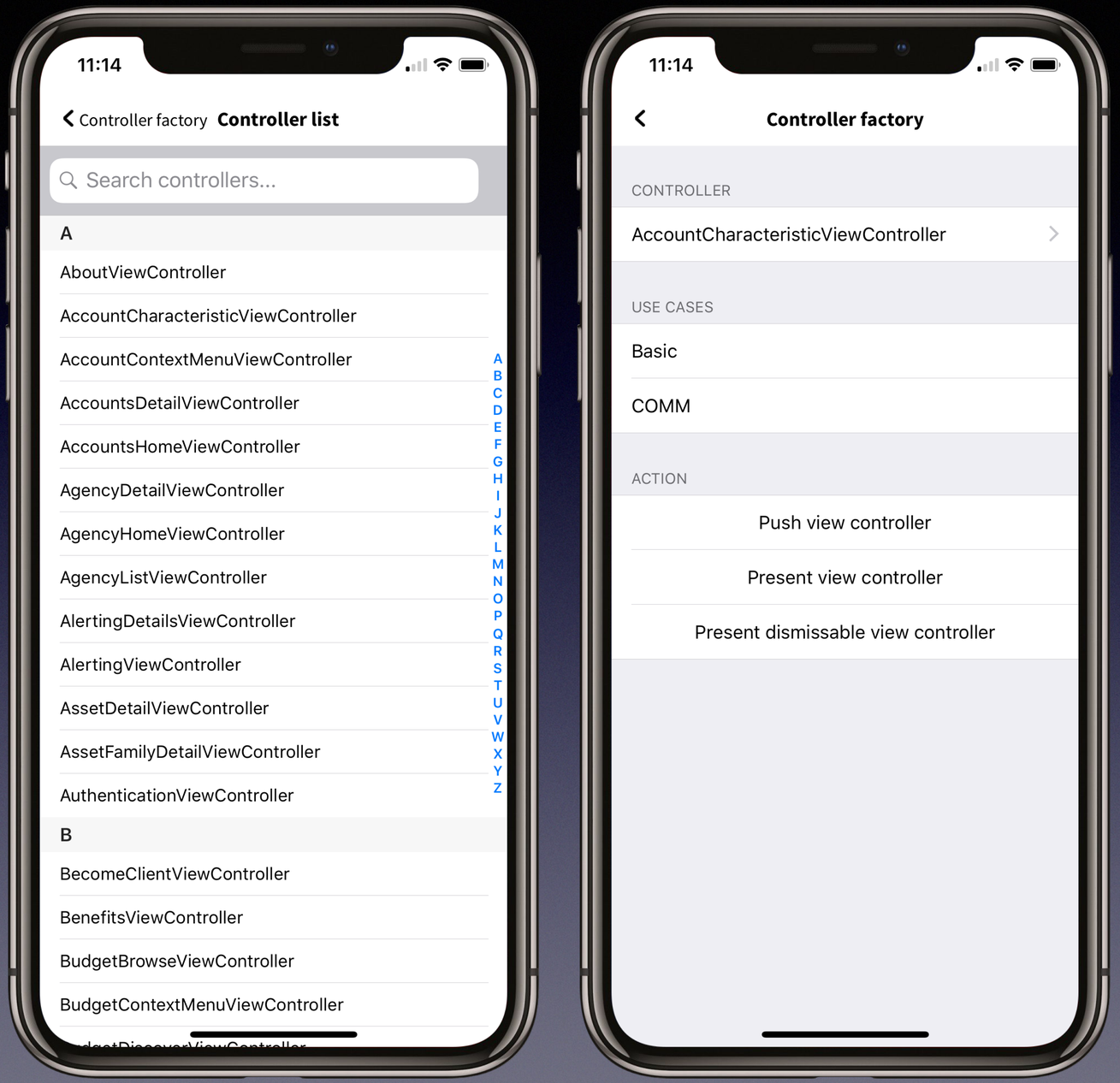
You have to make sure an app displays properly on the iPhone X, but:

- Data is driven by web-services
- No mocks are available
- More than 150 different screens
- Some screens are part of complex business scenarios

What would be the right tool?

A debug view!

- Lists all available UIViewControllers
- Indicate a specific use case, if needed
- Instantiates, then pushes or presents the controller



How do we get the data?

Objective-C

```
import ObjectiveC
```

The ObjectiveC module

- Exposes C APIs to interact with the ObjectiveC runtime
- Perfectly OK to use it with Swift

The ObjectiveC module

```
func objc_copyClassNamesForImage(_ image: UnsafePointer<Int8>,
                                 _ outCount: UnsafeMutablePointer<UInt32>?)
                                 -> UnsafeMutablePointer<UnsafePointer<Int8>>?
```

- This function returns a list of all the classes defined in a given bundle
- That’s all the data you’ll ever need to build a basic debug view!

How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

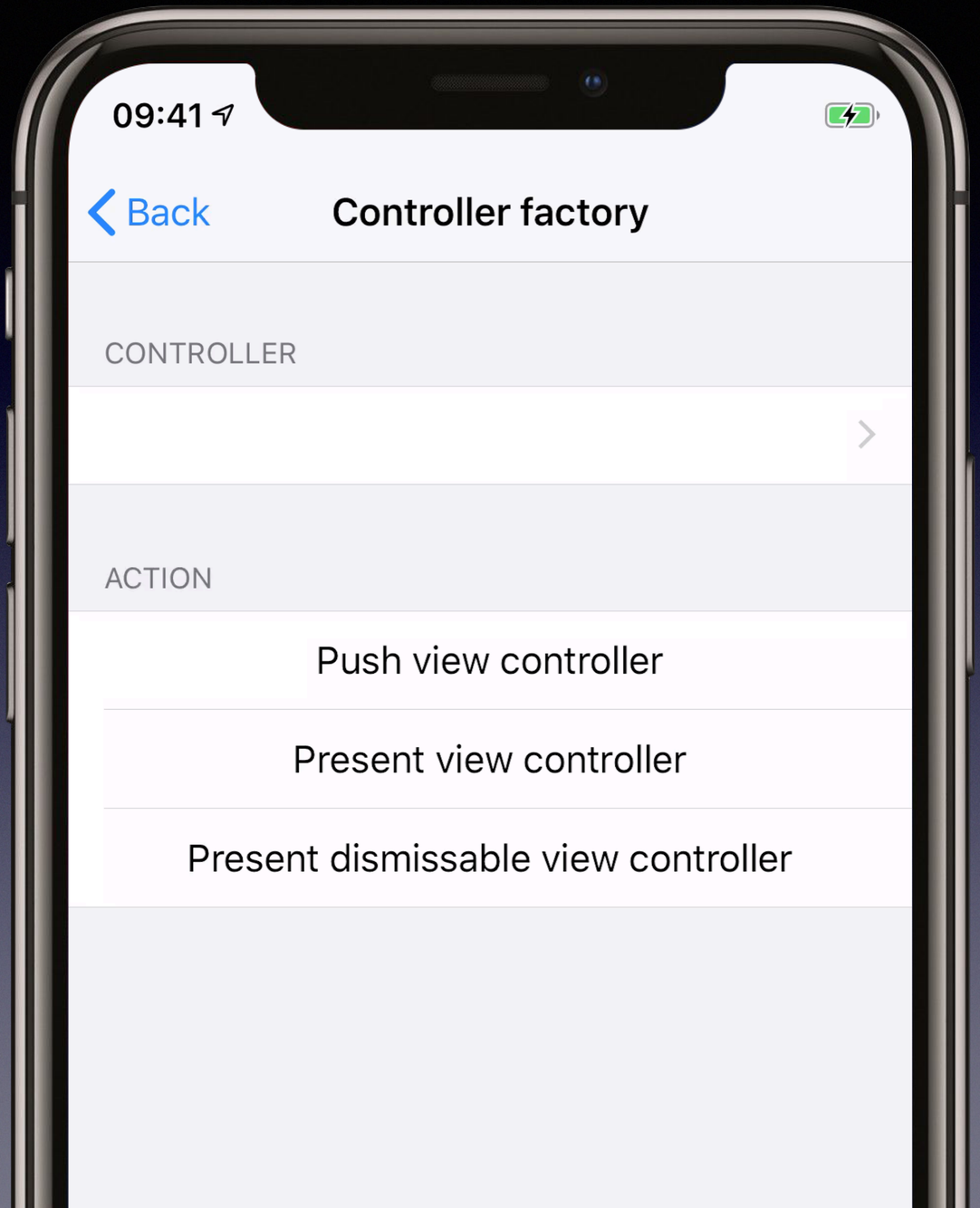
How to use it?

```
extension Bundle {
    func retrieveAllViewControllers() -> [String] {

        guard let bundlePath = self.executablePath else { return [] }

        var viewControllers = [String]()
        var size: UInt32 = 0

        let classes = objc_copyClassNamesForImage(bundlePath, &size)

        for index in 0..<size {
            if let className = classes?[Int(index)],
                let name = NSString.init(utf8String:className) as String?,
                NSClassFromString(name) is UIViewController.Type
            {
                viewControllers.append(name)
            }
        }

        return viewControllers
    }
}
```

How to use it?

```
func instantiateController(named name: String) -> UIViewController? {
    let controllerClass = NSClassFromString(name) as? UIViewController.Type
    return controllerClass?.init()
}
```

That’s it!


09:41
Back
Controller factory
CONTROLLER
ACTION
Push view controller
Present view controller
Present dismissable view controller

We’ve just seen the « free » part…

…let’s look at the « almost » side of things

Providing Initial Data

```
@objc public protocol DebugViewCompliant {
    func prepareForDebugView()
}

extension MyViewController: DebugViewCompliant {
    func prepareForDebugView() {
        // do some configuration
    }
}
```

Handling Use Cases

```
@objc public protocol DebugViewUseCaseCompliant {
    static func getUseCases() -> [String]
    func prepareForDebugView(useCase: String)
}


extension MyViewController: DebugViewUseCaseCompliant {

    static func getUseCases() -> [String] {
       return ["User Not Logged", "User Logged In"]
    }

    func prepareForDebugView(useCase: String) {
        // do some configuration according to `useCase`
    }
}
```

Working with Storyboards

```
@objc public protocol DebugViewInstantiable {
    static func initForDebugView() -> UIViewController
}

extension MyViewController: DebugViewInstantiable {
    static func initForDebugView() -> UIViewController {

        let storyboard = UIStoryboard(name: "Main", bundle: nil)

        let viewController =
            storyboard.instantiateViewController("MyViewController") as!
        MyViewController

        return viewController
    }
}
```

Does it work on a real project?

Does it work on a real project?

YES!

How can I use it?

How can I use it?

- Available on Github : https://github.com/worldline/ControllerFactory
- Works with CocoaPods and Carthage
- You just need to call:

```
ControllerFactory.instantiate(bundle: Bundle.main)
```

- Only one limitation: it does not work with generic classes 😕

See you tomorrow!