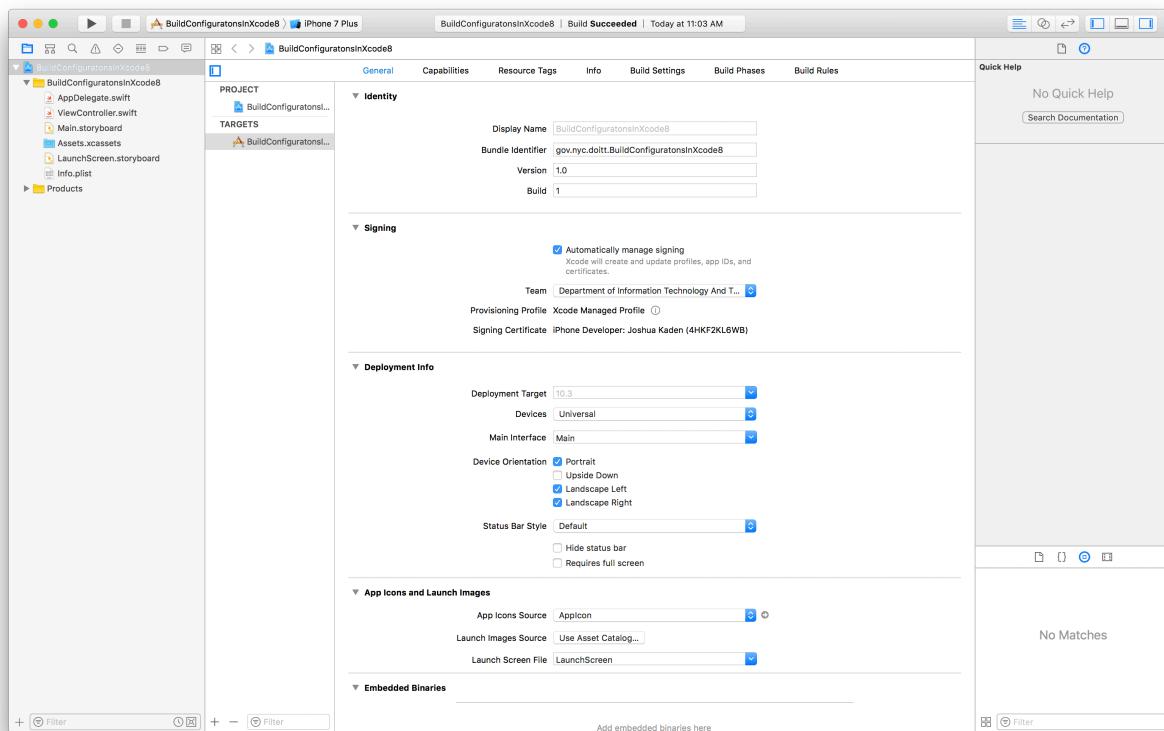


This article describes the technique of using schemes in Xcode to control configuration settings, for iOS mobile application development. This method allows for great flexibility in the configuration of an app, while minimizing the possibility of human error.

Step 1

NEW PROJECT

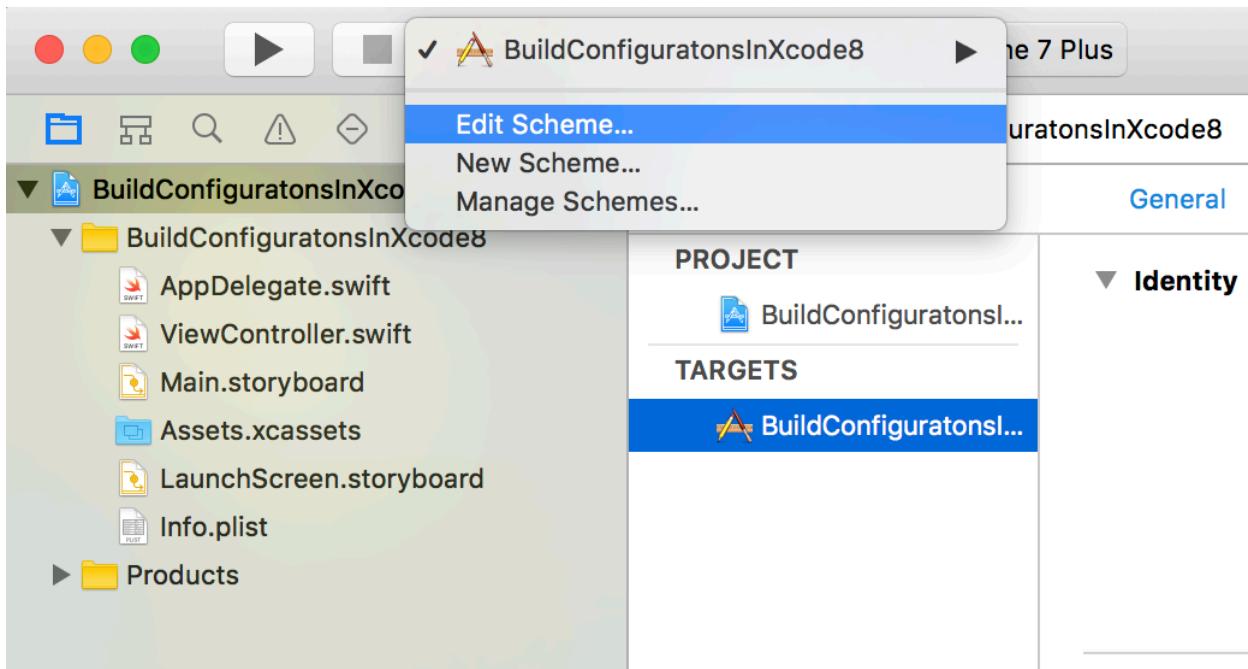
Open Xcode, and create a new iOS project. Select the Single View Application option.



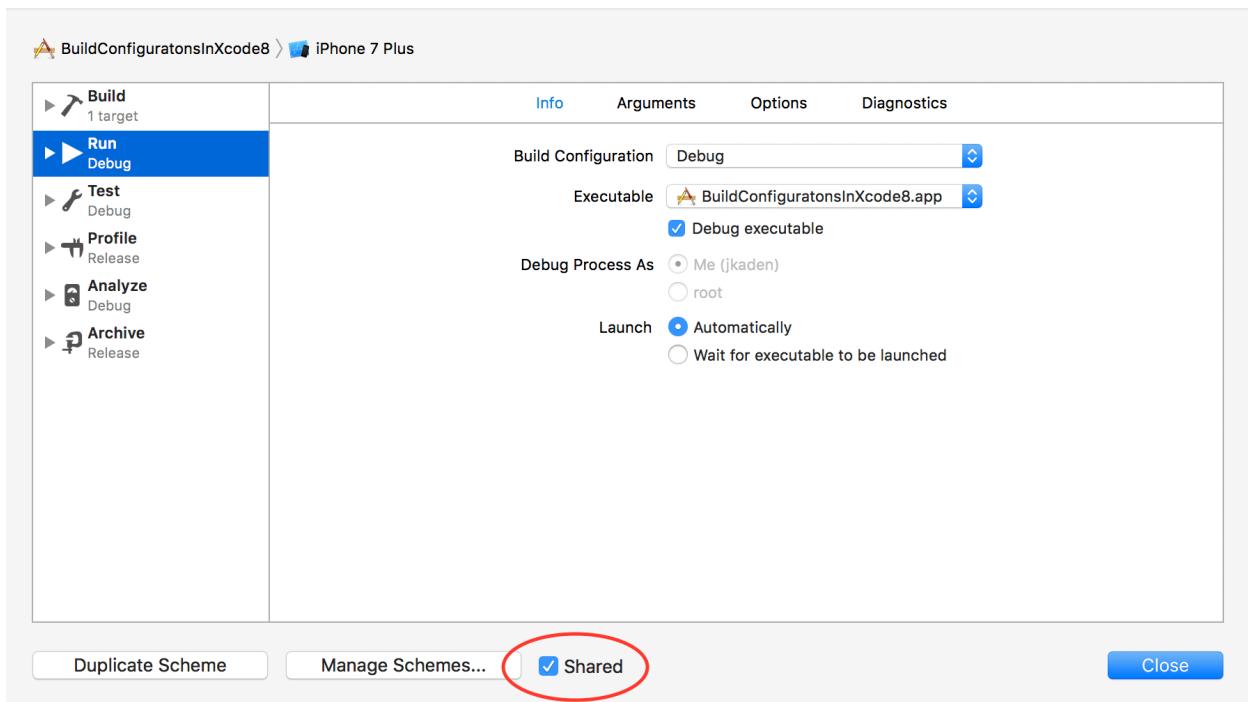
Step 2

SHARED SCHEME

Edit the default scheme by selecting it, and clicking “Edit Scheme”.



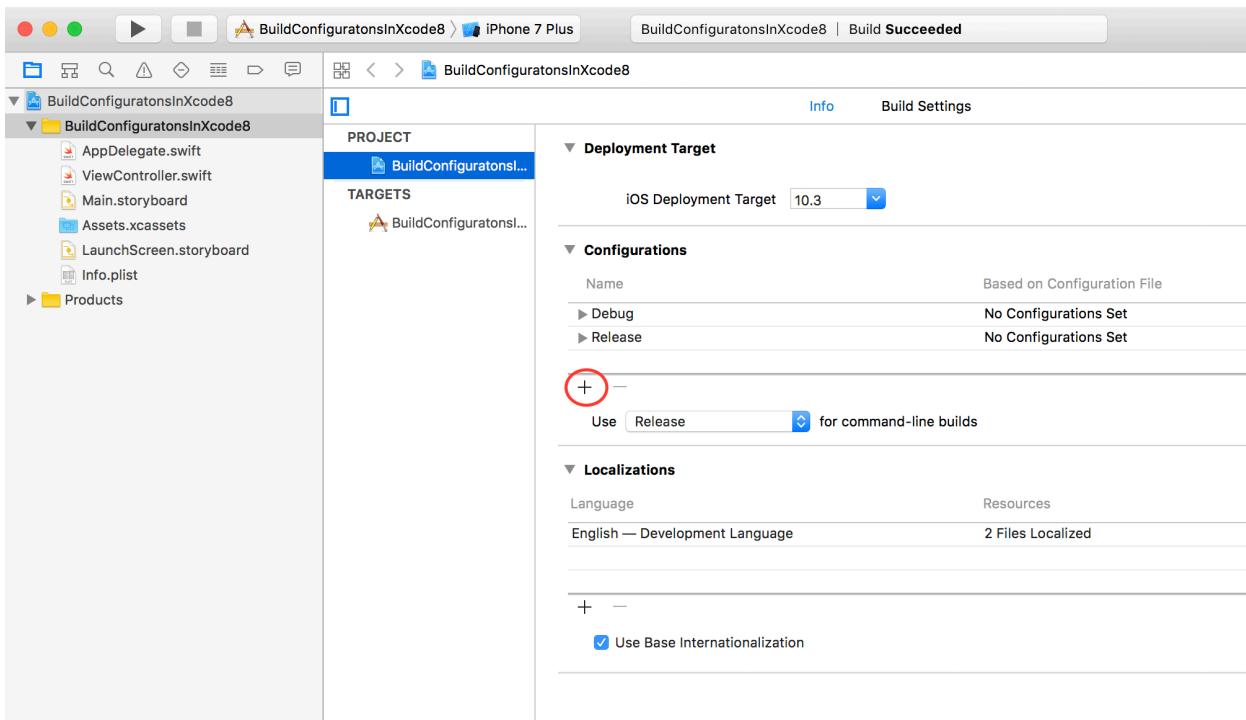
Check the “Shared” checkbox.



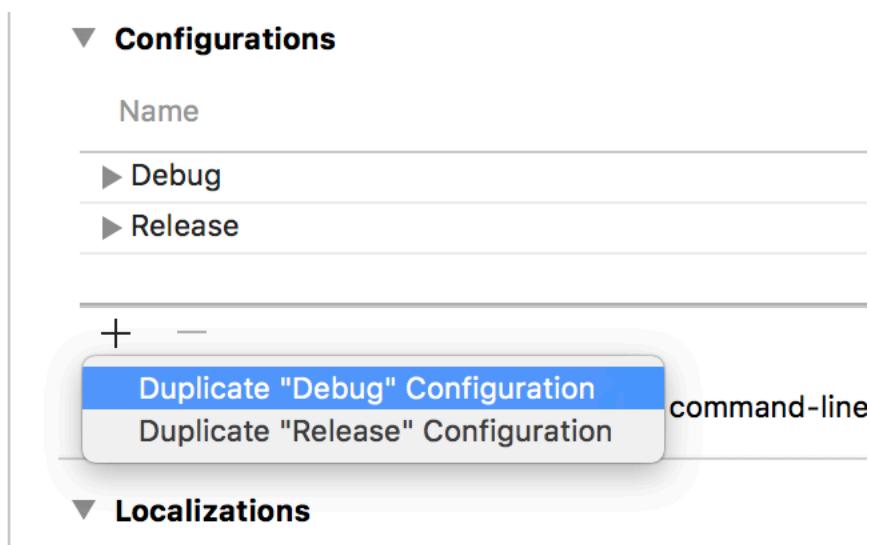
Step 3

DUPLICATE CONFIGURATIONS

Select the project in the Project Navigator. On the Info tab of the project, click the “+” symbol at the bottom left of the Configurations section.



On the pop-up menu, click “Duplicate ‘Debug’ Configuration”.



Name the new configuration: “Debug-Development”. Repeat the process by duplicating the “Release” configuration, and use the new name “Release-Development”.

▼ Configurations	
Name	Based on Configuration File
▶ Debug	No Configurations Set
▶ Debug-Development	No Configurations Set
▶ Release	No Configurations Set
▶ Release-Development	No Configurations Set
+ -	

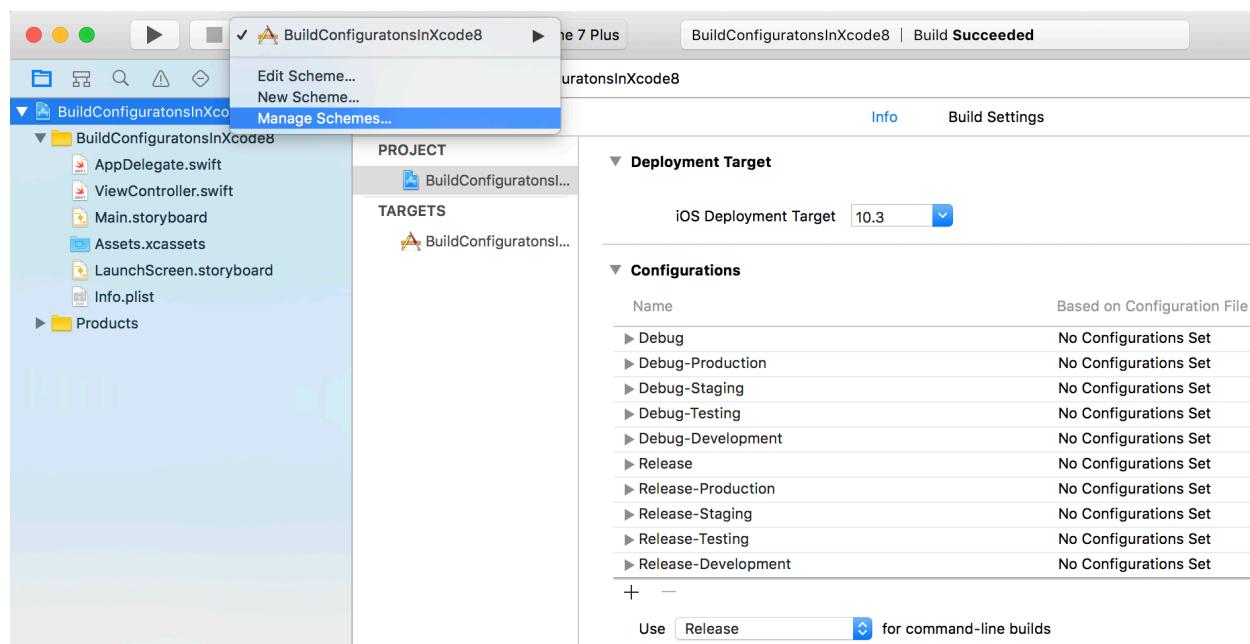
Repeat these two steps for the other three environments: Testing, Staging, and Production.

▼ Configurations	
Name	Based on Configuration File
▶ Debug	No Configurations Set
▶ Debug-Production	No Configurations Set
▶ Debug-Staging	No Configurations Set
▶ Debug-Testing	No Configurations Set
▶ Debug-Development	No Configurations Set
▶ Release	No Configurations Set
▶ Release-Production	No Configurations Set
▶ Release-Staging	No Configurations Set
▶ Release-Testing	No Configurations Set
▶ Release-Development	No Configurations Set
+ -	

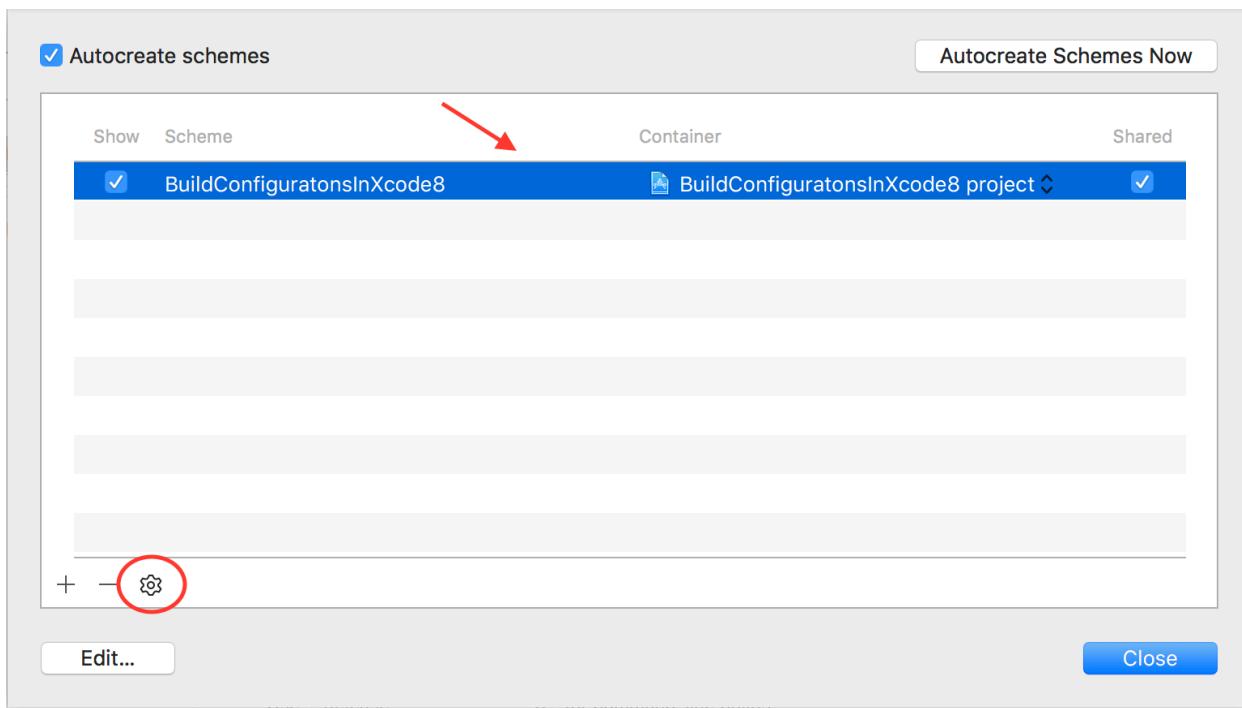
Step 4

DUPLICATE SCHEMES

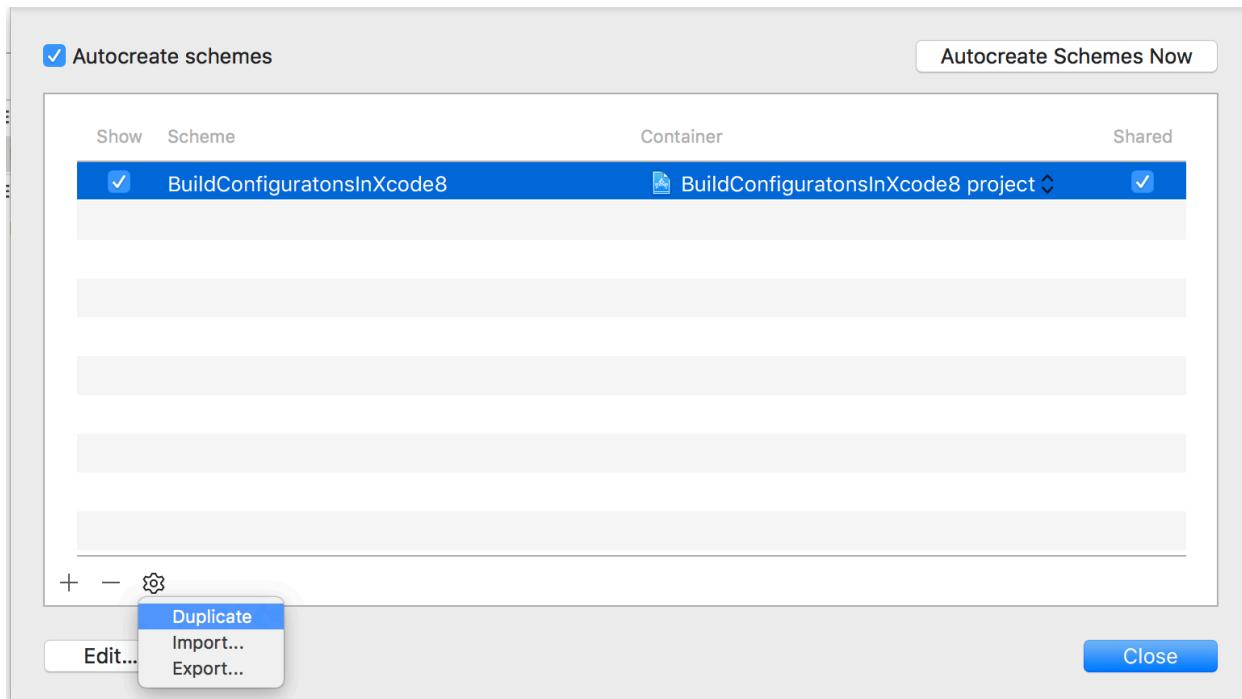
Launch the list of schemes by selecting the current scheme, and clicking “Manage Schemes”.



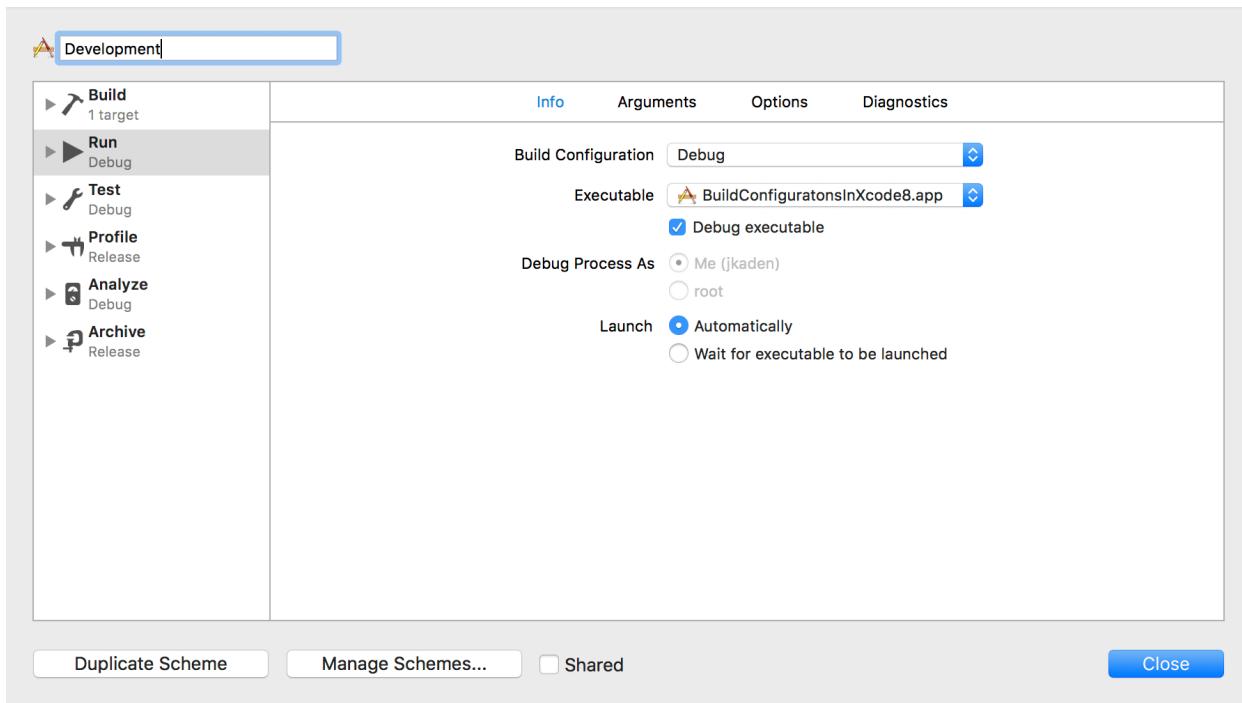
Select the listed scheme, and click the gear icon, at the window's lower left.



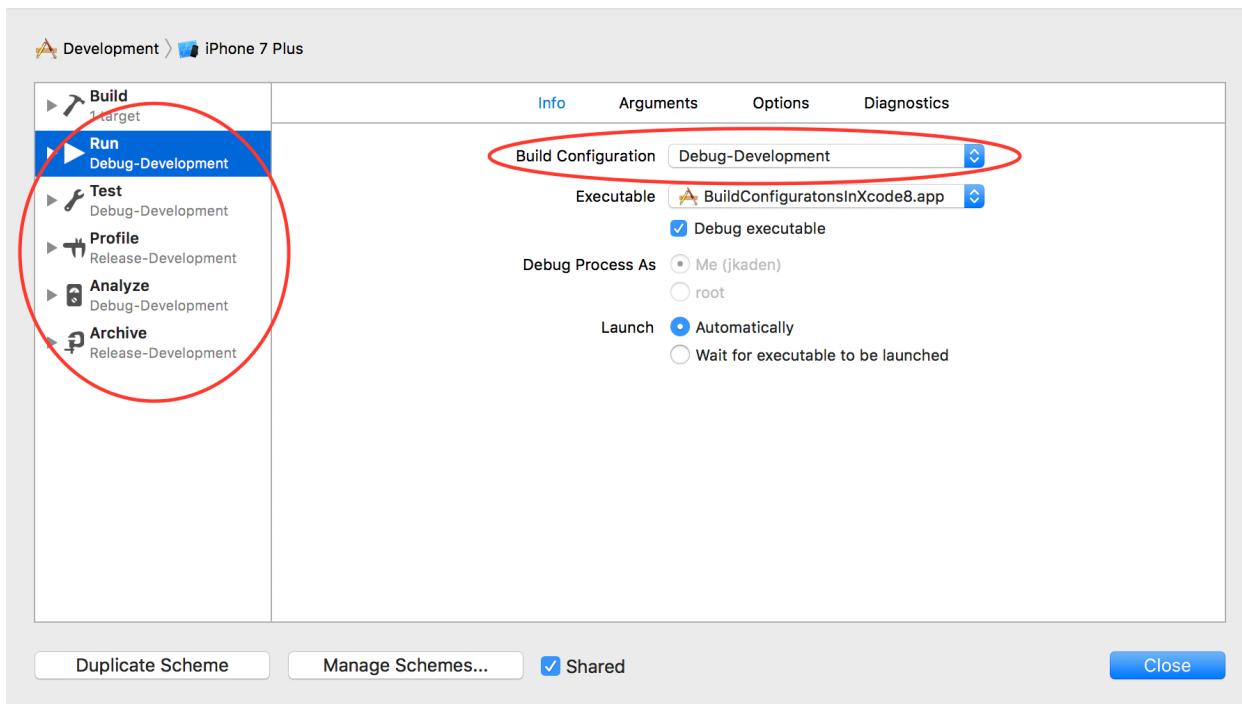
On the pop-up menu, click “Duplicate”.



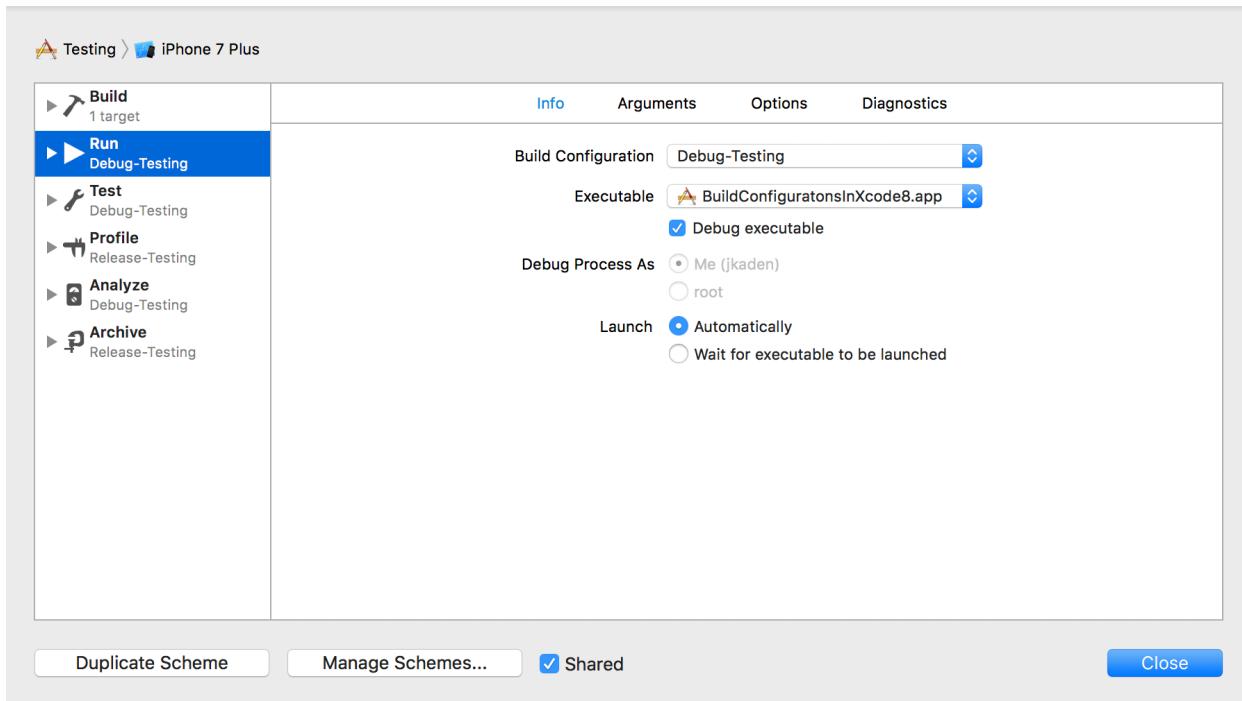
Name the new scheme “Development”.



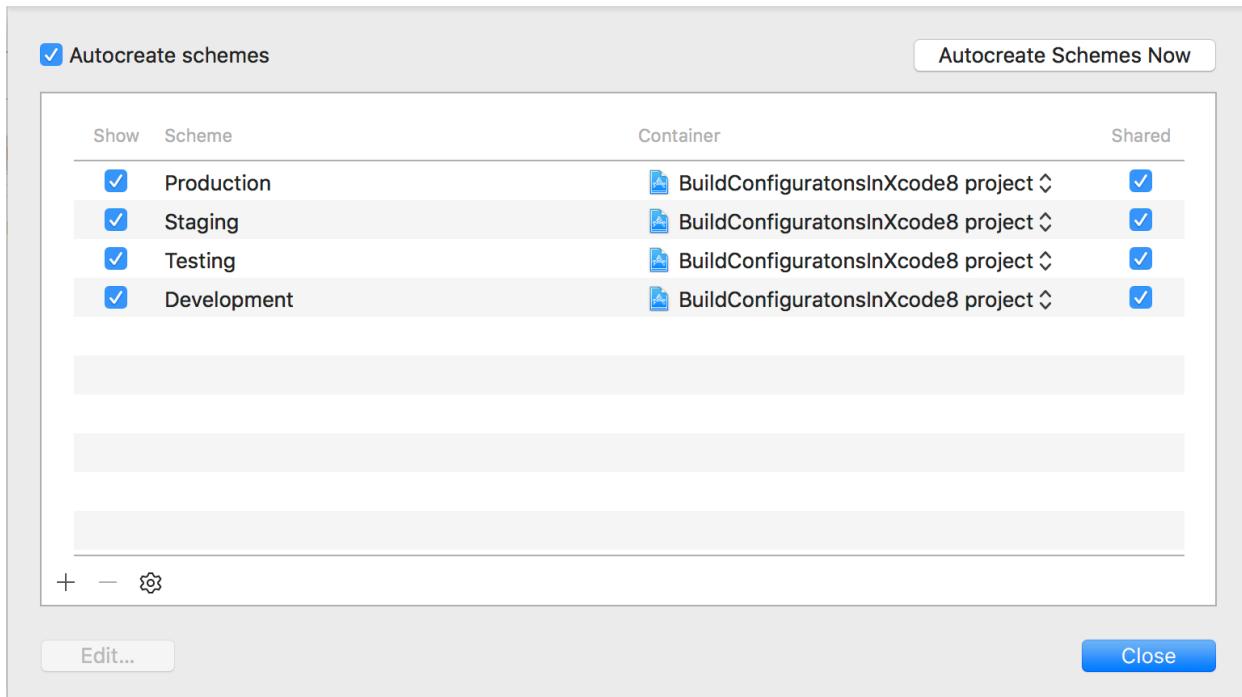
Select the Run step, and, on the Info tab, change the Build Configuration to “Debug-Development”. Do the same for Test and Analyze. For Profile and Release, change the Build Configuration to “Release-Development”. Also, make sure that “Shared” is checked.



Click Close to get back on the list of schemes. Select the default scheme, and duplicate it again, this time using the new name “Testing”. Change the Testing scheme’s Build Configurations by following the same pattern as before, as shown. Note that this time, we use the “Debug-Testing” and “Release-Testing” configurations.



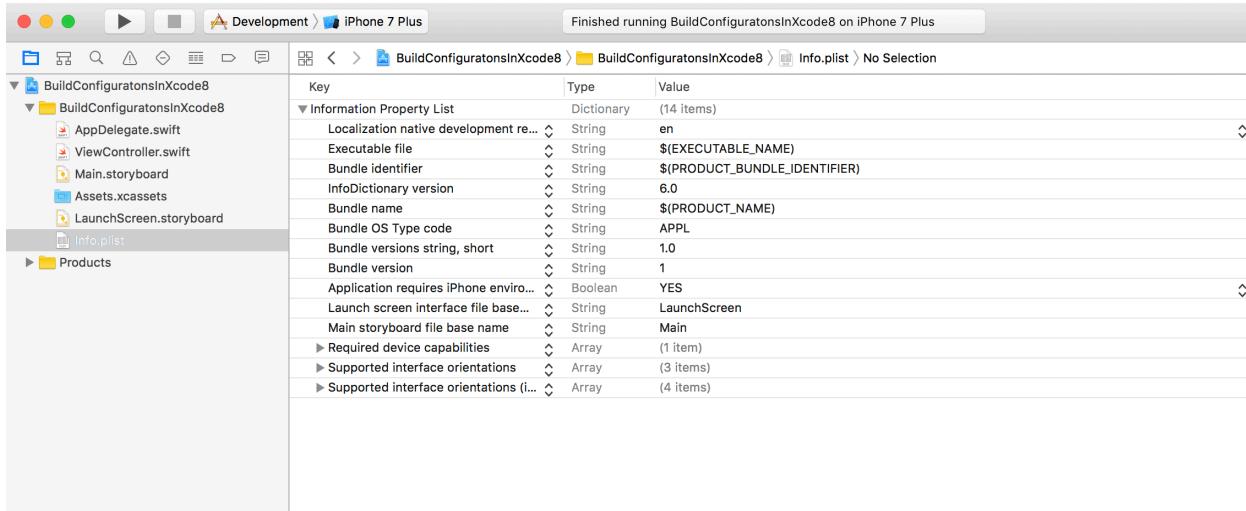
Create a new scheme named Staging, and one named Production, both by duplicating the default scheme. Set the proper Build Configurations for each of the new schemes. When you’re done, you may delete the default scheme, if you wish.



Step 5

MODIFY INFO PLIST

Select Info.plist on the project navigator.



Create a new key named “Configuration”. For its value, enter “\$(CONFIGURATION)”. Now the system-supplied configuration value (as determined by the active scheme) is

The screenshot shows the Xcode Project Navigator with the following details:

- Project: BuildConfigurationsInXcode8
- Target: iPhone 7 Plus
- File: Info.plist
- Description: No Selection

The Info.plist table now includes a new key "Configuration" with the value "\$(CONFIGURATION)". The "Information Property List" key has been circled in red.

Key	Type	Value
▼ Information Property List	Dictionary	(15 items)
Configuration	String	\$(CONFIGURATION)
Localization native development region	String	en
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle version	String	1
Application requires iPhone environment	Boolean	YES
Launch screen interface file base name	String	LaunchScreen
Main storyboard file base name	String	Main
► Required device capabilities	Array	(1 item)
► Supported interface orientations	Array	(3 items)
► Supported interface orientations (i...)	Array	(4 items)

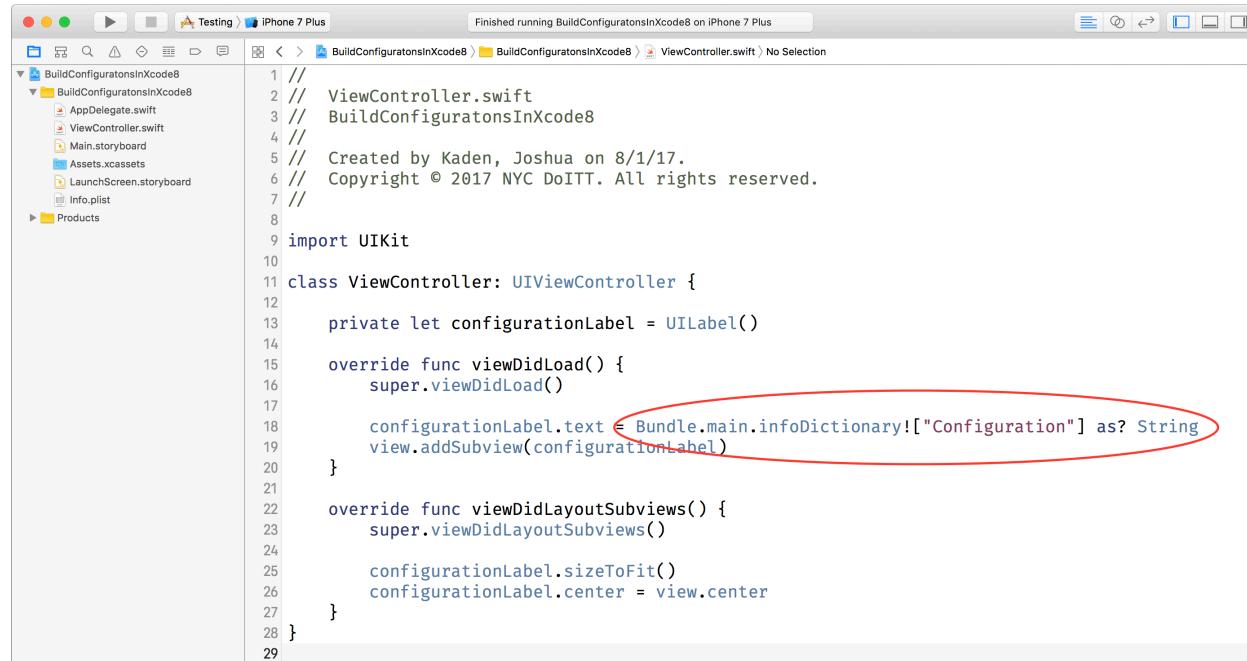
accessible to your code via the info plist.

Step 6

READ CONFIGURATION AND DISPLAY IT

Info.plist is exposed as a dictionary via the `Bundle.main.infoDictionary` property.

Modify the ViewController class as shown below. This code reads the active configuration, and displays it on the screen.



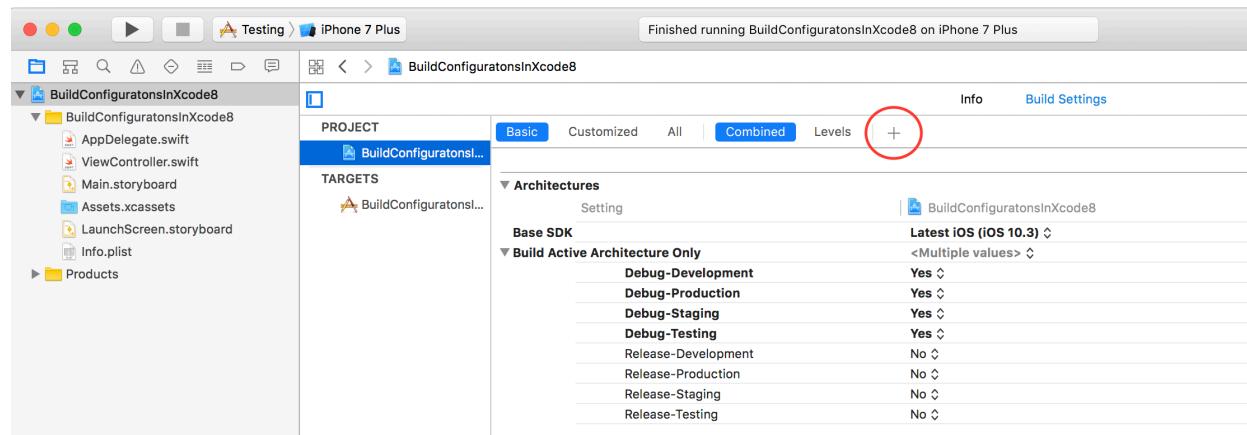
```

1 ///////////////////////////////////////////////////////////////////////////////
2 ///////////////////////////////////////////////////////////////////////////////
3 // View Controller.swift
4 // BuildConfiguratonsInXcode8
5 ///////////////////////////////////////////////////////////////////////////////
6 // Created by Kaden, Joshua on 8/1/17.
7 // Copyright © 2017 NYC DoITT. All rights reserved.
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     private let configurationLabel = UILabel()
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17
18         configurationLabel.text = Bundle.main.infoDictionary!["Configuration"] as? String
19         view.addSubview(configurationLabel)
20     }
21
22     override func viewDidLayoutSubviews() {
23         super.viewDidLayoutSubviews()
24
25         configurationLabel.sizeToFit()
26         configurationLabel.center = view.center
27     }
28 }
29 
```

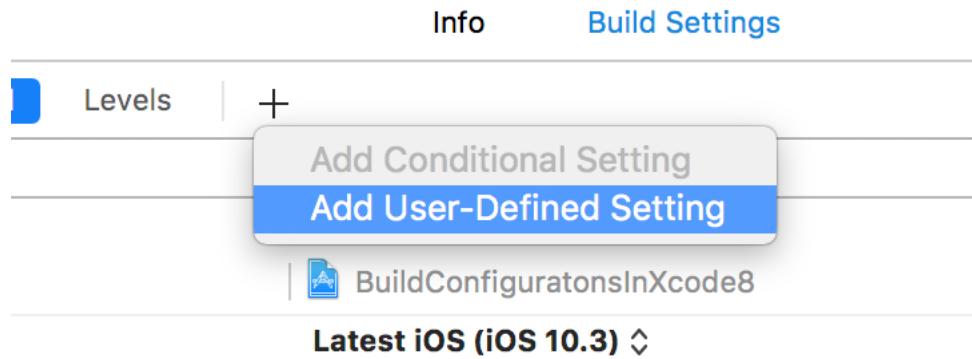
Step 7

USER-DEFINED BUILD SETTING

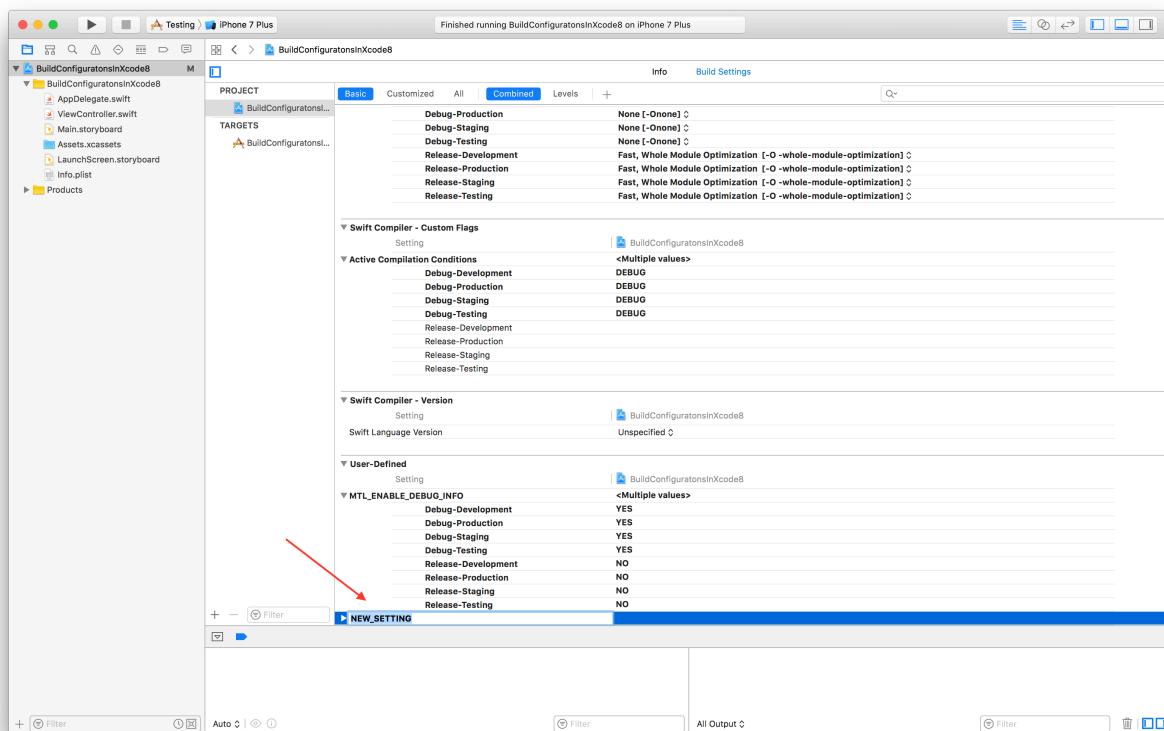
Select the project in the Project Navigator. On the Build Settings tab of the project, click the “+” symbol at the top.



On the pop-up menu, click “Add User-Defined Setting”.



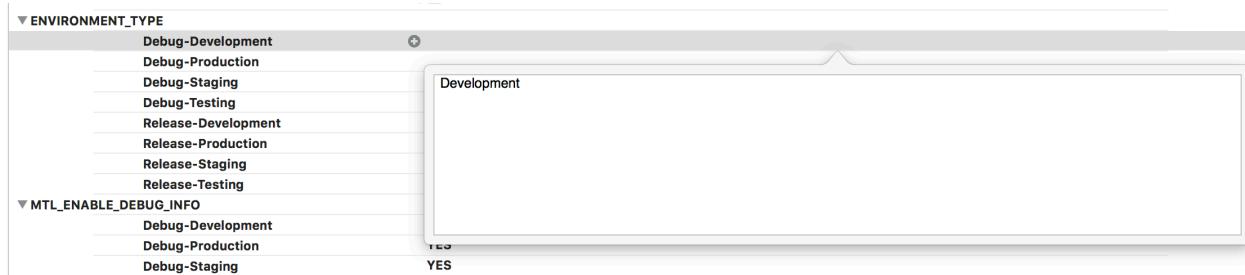
A new row will be added at the bottom of the list, in the User-Defined section.



Name the new row “ENVIRONMENT_TYPE”. Your list will look like this:

▼ ENVIRONMENT_TYPE
Debug-Development
Debug-Production
Debug-Staging
Debug-Testing
Release-Development
Release-Production
Release-Staging
Release-Testing

Double-click the Value for the Debug-Development row. In the pop-up editor, enter “Development”.



Add Values for the remaining configurations, so that your list looks like this:

User-Defined		Setting	BuildConfigurationsInXcode8
▼ ENVIRONMENT_TYPE			<Multiple values>
Debug-Development			Development
Debug-Production			Production
Debug-Staging			Staging
Debug-Testing			Testing
Release-Development			Development
Release-Production			Production
Release-Staging			Staging
Release-Testing			Testing
▼ MTL_ENABLE_DEBUG_INFO			<Multiple values>
Debug-Development			YES
Debug-Production			YES

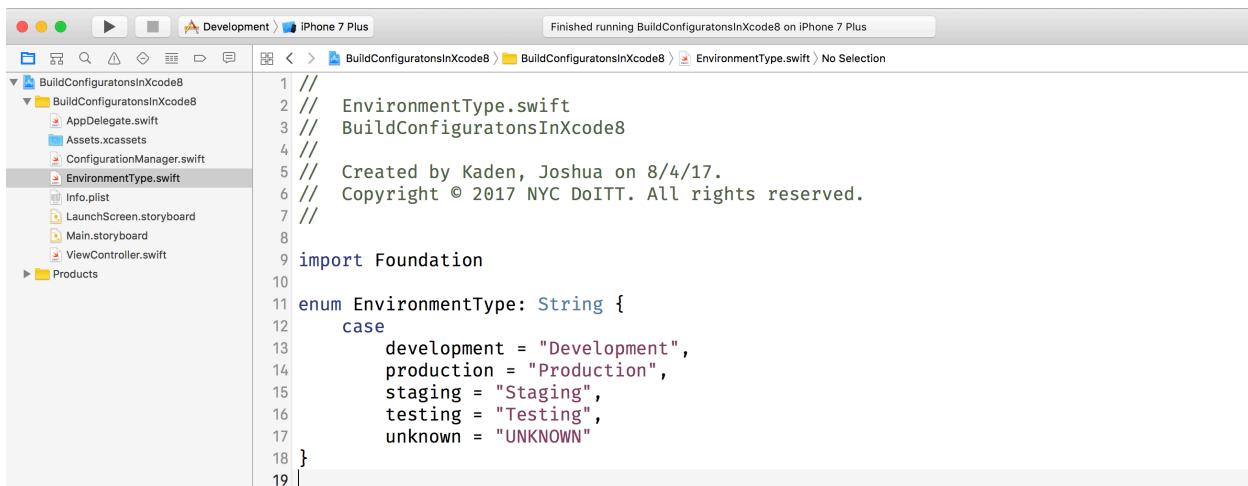
Back on Info.plist, create a new row named “Environment Type”. Give it a value of “\$(ENVIRONMENT_TYPE)”.

Key	Type	Value
Environment Type	String	\$(ENVIRONMENT_TYPE)
Configuration	String	\$(CONFIGURATION)
Localization native development re...	String	en
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle version	String	1

Step 8

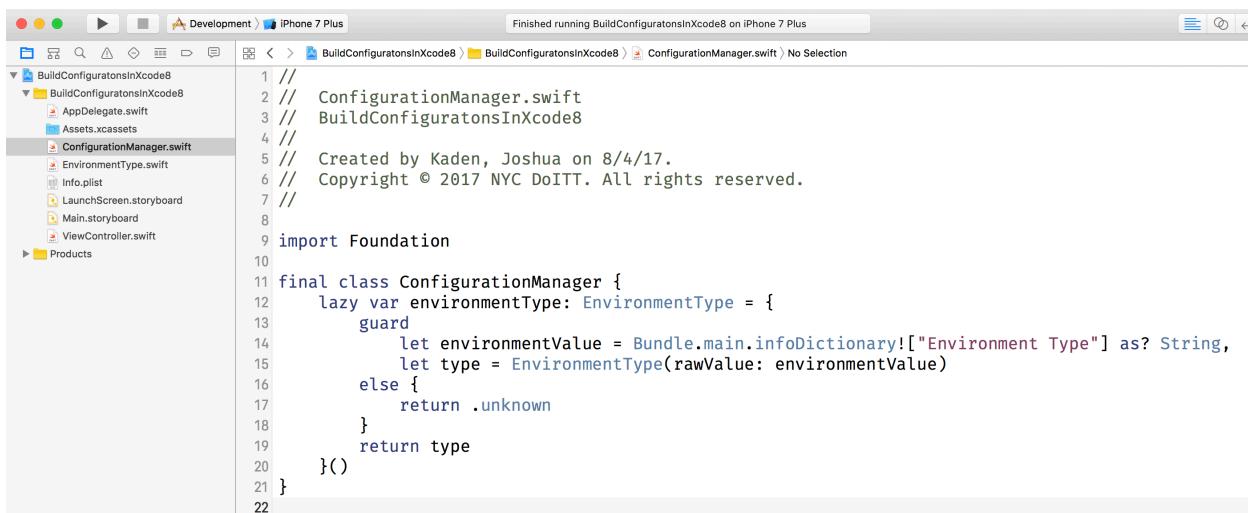
CREATE ENUM AND MANAGER

Create an enum named “EnvironmentType”. This will act as a link between the string values in Info.plist and your code.



```
1 //
2 //  EnvironmentType.swift
3 //  BuildConfiguratosInXcode8
4 //
5 //  Created by Kaden, Joshua on 8/4/17.
6 //  Copyright © 2017 NYC DoITT. All rights reserved.
7 //
8
9 import Foundation
10
11 enum EnvironmentType: String {
12     case
13         development = "Development",
14         production = "Production",
15         staging = "Staging",
16         testing = "Testing",
17         unknown = "UNKNOWN"
18 }
19 |
```

Create a class named “ConfigurationManager”. Its job is to read from Info.plist and supply the current EnvironmentType.



```
1 //
2 //  ConfigurationManager.swift
3 //  BuildConfiguratosInXcode8
4 //
5 //  Created by Kaden, Joshua on 8/4/17.
6 //  Copyright © 2017 NYC DoITT. All rights reserved.
7 //
8
9 import Foundation
10
11 final class ConfigurationManager {
12     lazy var environmentType: EnvironmentType = {
13         guard
14             let environmentValue = Bundle.main.infoDictionary!["Environment Type"] as? String,
15             let type = EnvironmentType(rawValue: environmentValue)
16         else {
17             return .unknown
18         }
19         return type
20     }()
21 }
22 |
```

Modify ViewController to display the current EnvironmentType.

```
1 //  
2 //  ViewController.swift  
3 //  BuildConfigurationsInXcode8  
4 //  
5 //  Created by Kaden, Joshua on 8/1/17.  
6 //  Copyright © 2017 NYC DoITT. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController {  
12  
13     private let configurationManager = ConfigurationManager()  
14     private var environmentLabel = UILabel()  
15  
16     override func viewDidLoad() {  
17         super.viewDidLoad()  
18  
19         let environmentValue = configurationManager.environmentType.rawValue  
20         environmentLabel.text = "Environment Type: \(environmentValue)"  
21         view.addSubview(environmentLabel)  
22     }  
23  
24     override func viewDidLayoutSubviews() {  
25         super.viewDidLayoutSubviews()  
26  
27         environmentLabel.sizeToFit()  
28         environmentLabel.center = view.center  
29     }  
30 }
```

Step 9

MAKE DIFFERENT URLs FOR EACH ENVIRONMENT

Enhance the EnvironmentType enum to return a different URL for each value.

```
1 //  
2 //  EnvironmentType.swift  
3 //  BuildConfigurationsInXcode8  
4 //  
5 //  Created by Kaden, Joshua on 8/4/17.  
6 //  Copyright © 2017 NYC DoITT. All rights reserved.  
7 //  
8  
9 import Foundation  
10  
11 enum EnvironmentType: String {  
12     case  
13         development = "Development",  
14         production = "Production",  
15         staging = "Staging",  
16         testing = "Testing",  
17         unknown = "UNKNOWN"  
18  
19     func serverURL() -> URL? {  
20         switch self {  
21             case .development:  
22                 return URL(string: "https://dev-server.nyc.gov")  
23             case .production:  
24                 return URL(string: "https://prod-server.nyc.gov")  
25             case .staging:  
26                 return URL(string: "https://stg-server.nyc.gov")  
27             case .testing:  
28                 return URL(string: "https://tst-server.nyc.gov")  
29             case .unknown:  
30                 return nil  
31         }  
32     }  
33 }
```

Here is code to simulate the launch of the URL associated with the current environment.

```
1 // ViewController.swift
2 // BuildConfigurationsInXcode8
3 //
4 //
5 // Created by Kaden, Joshua on 8/1/17.
6 // Copyright © 2017 NYC DoITT. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     private let configurationManager = ConfigurationManager()
14     private let urlButton = UIButton()
15
16     override func viewDidLoad() {
17         super.viewDidLoad()
18
19         let environmentValue = configurationManager.environmentType.rawValue
20         urlButton.setTitle("Open \(environmentValue) URL", for: .normal)
21         urlButton.setTitleColor(.blue, for: .normal)
22         urlButton.addTarget(self, action: #selector(didTapUrlButton(_:)), for: .touchUpInside)
23         view.addSubview(urlButton)
24     }
25
26     func didTapUrlButton(_ sender: UIButton) {
27         guard let url = configurationManager.environmentType.serverURL() else { return }
28         let message = "URL is: \(url.absoluteString)"
29         let alert = UIAlertController(title: "Button Tap", message: message, preferredStyle: .alert)
30         alert.addAction(UIAlertAction(title: "Ok", style: .default, handler: { _ in }))
31         present(alert, animated: true)
32     }
33
34     override func viewDidLayoutSubviews() {
35         super.viewDidLayoutSubviews()
36
37         urlButton.frame.size = CGSize(width: 250, height: 60)
38         urlButton.center = view.center
39     }
40 }
41
```

The sample code for this article can be found at <https://github.com/JoshuaKaden/BuildConfigurationsInXcode8>.

Hopefully this will be enough to get you started with implementing build configurations in your project.