



mobile Dev Diary



Insights about
Swift Testing
Tags

scroll to find out



#1 Tags

What are Swift Testing Tags? 🤔

- Tags are the new feature that comes together with Xcode 16.0 and Swift Testing framework.
- Designed to help you categorize your tests.
- Tags can help you divide tests within a test target into few executable test plans.

#2 Tags for Test

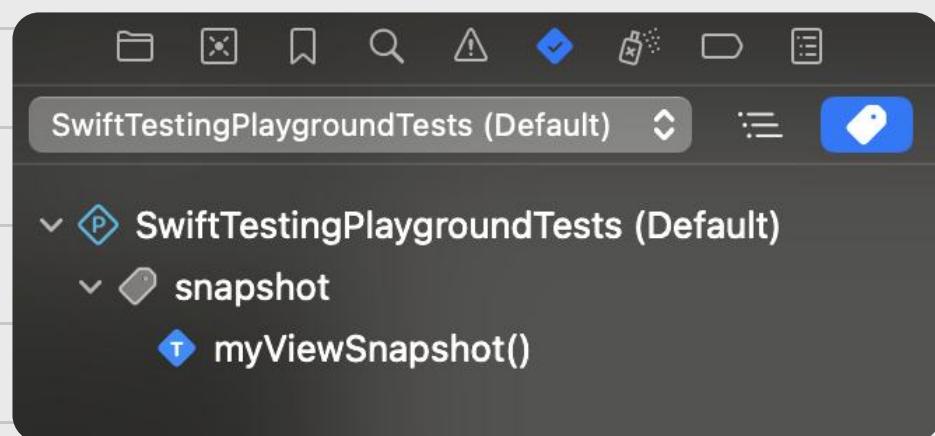
The first step is our tag definition.
Do it by defining a static property in an extension to the Tag using @Tag macro.

```
extension Tag {  
    @Tag static let snapshot: Self  
}
```

Next, apply it to the test

```
@Test(.tags(.snapshot))  
func myViewSnapshot() { ... }
```

and then it's categorized in the test navigator



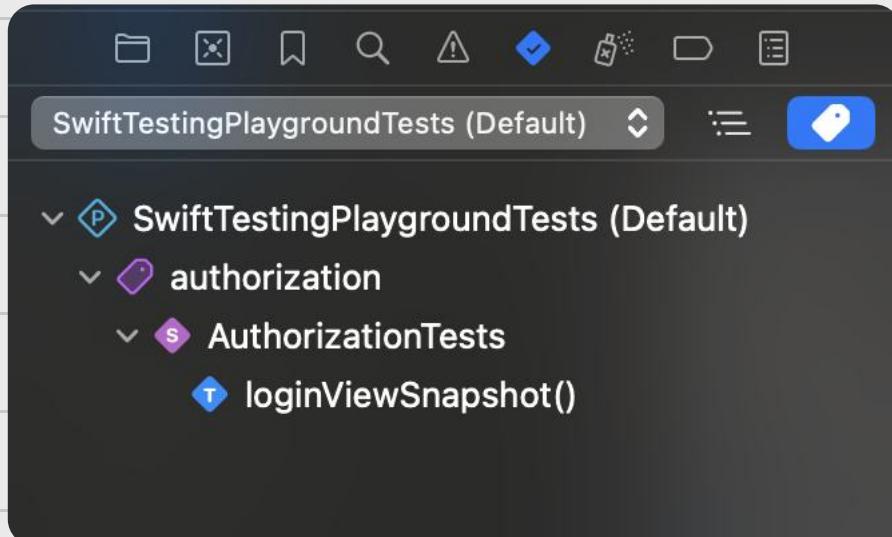
#3 Tags for Suite

Is it possible to tag Suites? - Yes! 

It's as simple as for a single Test 

```
extension Tag {  
    @Tag static let authorization: Self  
}  
  
@Suite(.tags(.authorization))  
struct AuthorizationTests {  
    @Test func loginViewSnapshot() { }  
}
```

test are
displayed
under the Tag
in the test
navigator 



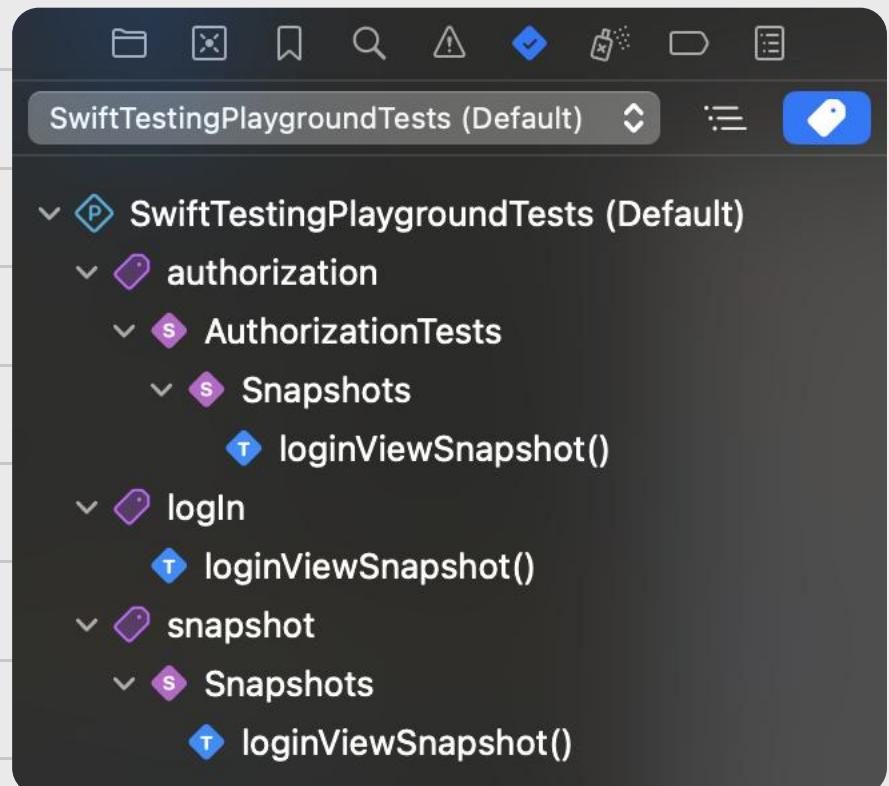
#4 Tags Nesting

It's possible to nest Tags freely in Suites .

```
@Suite(.tags(.authorization))
struct AuthorizationTests {

    @Suite(.tags(.snapshot))
    struct Snapshots {
        @Test(.tags(.logIn))
        func loginViewSnapshot() { }
    }
}
```

tests
navigator
with nested
Tags 



#5 Multiple Tags

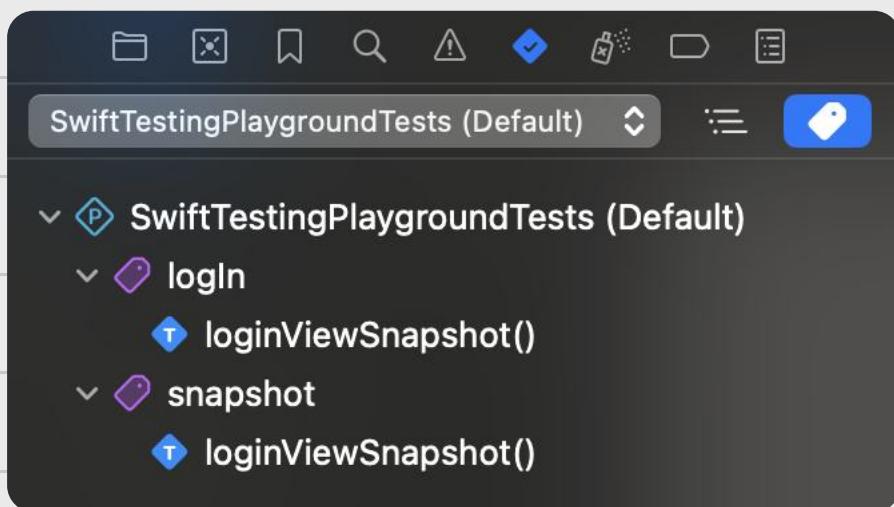
Need more than one tag? - No worries, it's possible! 

When constructing the list of Tags using `.tags()` function, it accepts zero or more than one tag as an argument (because it's variadic argument - "Tag...").

```
@Suite
struct AuthorizationTests {
    @Test(.tags(.logIn, .snapshot))
    func loginViewSnapshot() { }
}
```

tests

navigator 



#6 Tags in Test Plan

Thanks to tagging, it's possible to define test plans based on them.

My use case:

Unit and snapshot tests mixed in one target. By tagging tests it's easy to create two testing plans: one for unit tests, second for snapshot tests.

The screenshot shows the Xcode Test Navigator interface. At the top, there are sections for "Include Tags:" (with "snapshot" selected) and "Exclude Tags:". Below this, there are tabs for "All", "Included", and "Excluded". Under "Included", there is a "Test Preview" section and a "Tags" section. The "Tags" section lists several test cases with their status (e.g., checked or unchecked), the test name, and associated tags (e.g., "snapshot", "login", "unit"). Some tests are marked as "Excluded".

Test Name	Status	Tags
loginViewSnapshot()	Checked	snapshot login
signUpViewSnapshot()	Checked	snapshot
loginFlowSuc...	unchecked	login unit
signUpFlowSu...	unchecked	Excluded unit

#7 Excluded Tags in Test Plan

It's possible to exclude tests using tags. Remember that exclusion has priority over inclusion. If a test is marked with tag A and B, tag A is included and tag B excluded - the test will not execute



SwiftTestingPlaygroundTests Choose Targets...

Include Tags: ANY login

Exclude Tags: ANY unit

All Included Excluded | All Swift Testing XCTest Filter

Included	Test Preview	Tags
-	SwiftTestingPlaygroundTests Execute in parallel	
-	AuthorizationTests	
<input checked="" type="checkbox"/>	loginViewSnapshot()	login snapshot
<input type="checkbox"/>	signUpViewSnapshot()	Excluded snapshot
<input type="checkbox"/>	loginFlowSucceeded()	Excluded unit login
<input type="checkbox"/>	signUpFlowSucceede...	Excluded unit

#8 Custom Tags

When defining tags remember they have to be defined inside the tag extension. If not, Xcode will remind you about that.



```
struct TestSuite {  
    @Tag static let units: Tag  
}  
✖ Attribute 'Tag' cannot be applied to a property except in an extension to 'Tag' (from macro 'Tag')
```

```
@Tag let units: Tag  
✖ Attribute 'Tag' cannot be applied to a global variable (from macro 'Tag')  
   ⚡ Declare in an extension to 'Tag' Fix  
   ⚡ Remove attribute 'Tag' Fix
```

Find this Interesting?

Follow me for more!

