

<https://github.com/FrenchKit/UIAutomationClassroom>

The screenshot shows the GitHub repository page for **FrenchKit / UIAutomationClassroom**. The repository has 10 watchers, 1 star, and 0 forks. It contains 16 commits, 11 branches, 0 releases, and 1 contributor. The repository is licensed under MIT.

Under the heading "Your recently pushed branches:", there are three branches listed:

- step8** (about 1 hour ago) with a button to "Compare & pull request".
- step9** (about 1 hour ago) with a button to "Compare & pull request".
- step10** (about 1 hour ago) with a button to "Compare & pull request".

Below the branches, there is a section for "Branch: master" with a "New pull request" button. To the right, there are buttons for "Create new file", "Upload files", "Find file", and "Clone or download".

The commit history shows a commit by **DmitryBespalov** titled "Added steps comments". Below this, a list of files and their commit messages is shown:

File	Commit Message	Time
Assets	Added controller implementation	
Localization	Added localization example	
UITestingWorkshop.xcodeproj	Some fixes	
UITestingWorkshop	Added mock server example	20 days ago
UITestingWorkshopUITests	Added steps comments	2 hours ago
deps	Added mock server example	20 days ago
mock-server	Added mock server example	20 days ago

A modal dialog is open for cloning the repository with SSH. It provides the SSH URL: `git@github.com:FrenchKit/UIAutomationC`. There are buttons for "Open in Desktop" and "Download ZIP". A link to "Use HTTPS" is also present.

XCUITest

Dmitry Bespalov

dmtrbsp@gmail.com

[@dnbespalov](#)

<https://github.com/DmitryBespalov>

<https://github.com/FrenchKit/UIAutomationClassroom>

Overview

- Basics of UI testing
- Writing your test
- Launch arguments
- Table views and collection views
- Localizations
- Working with network

<https://github.com/FrenchKit/UIAutomationClassroom>

Basics of UI testing

<https://github.com/FrenchKit/UIAutomationClassroom>

Basics of UI testing

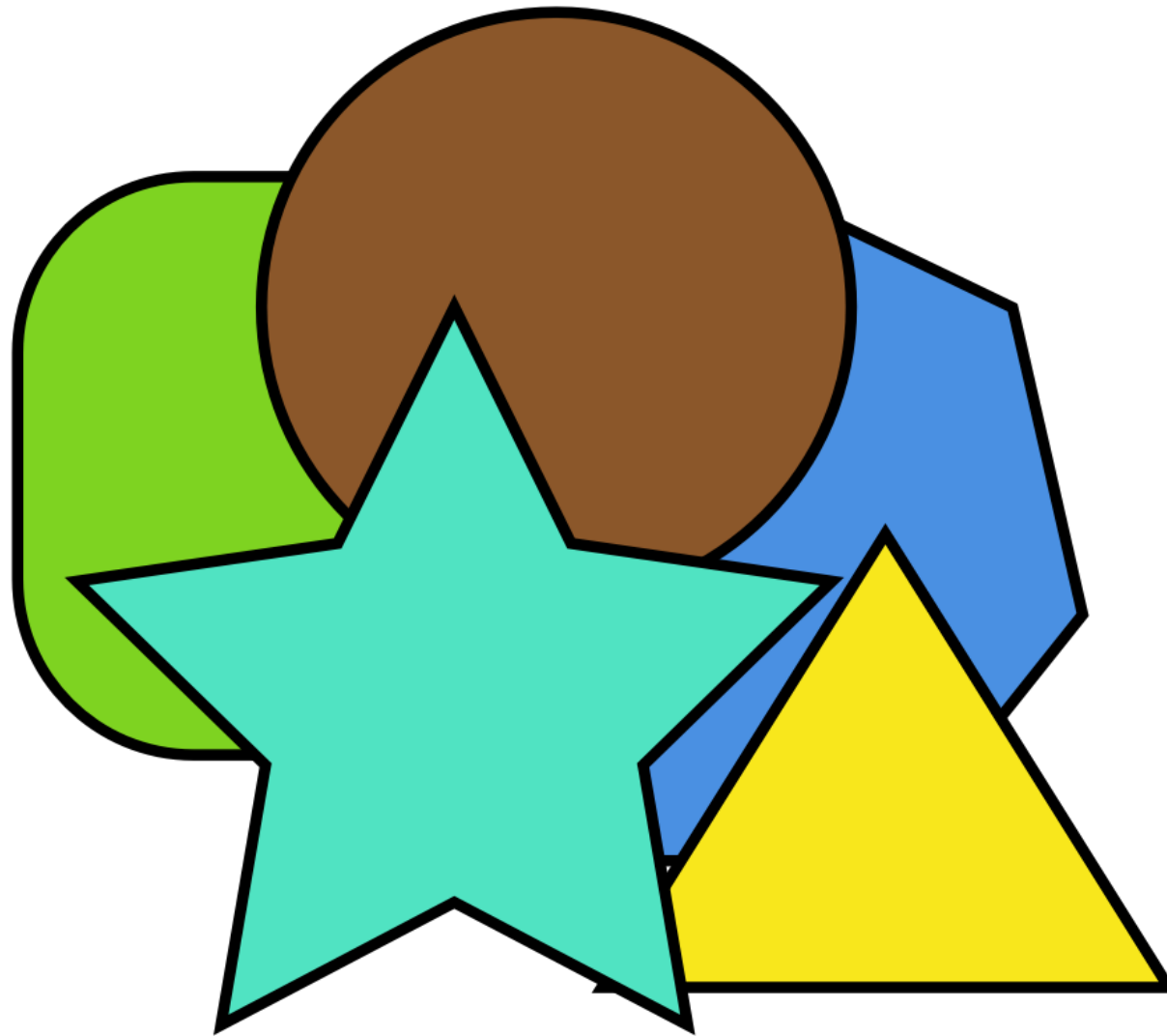
- XCUIApplication
- XCUIElementQuery
- XCUIElement
- XCTAssertEqual, XCTAssertTrue, ...
- UIView.accessibilityIdentifier

<https://github.com/FrenchKit/UIAutomationClassroom>

Example app overview

- Tab bar with 2 tabs: “First” and “Second”
- First tab’s screen leads to table view and collection view screens
- Table view has 50 items which can be opened
- Collection view has 50 items which can be opened
- Second tab’s screen is a placeholder

<https://github.com/FrenchKit/UIAutomationClassroom>



<https://github.com/FrenchKit/UIAutomationClassroom>

Step 1: “Show TableView”

UITestingWorkshopUITests.swift:

```
// TODO: STEP 1: write a test case that asserts  
that "Show TableView" button exists
```

if stuck, check out branch “step1” from git:
\$> git checkout step1

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 2: Assertions

AssertionTests.swift:

```
// TODO: STEP 2: assert that testing label's text  
is "UI test in progress"
```

if stuck, check out branch “step2” from git:
\$> git checkout step2

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 3: Open collection view

AssertionTests.swift:

```
// TODO: STEP 3: write a test method that opens  
collection view screen
```

if stuck, check out branch “step3” from git:
\$> git checkout step3

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 4: Second tab button

AssertionTests.swift:

```
// TODO: STEP 4: write a test method that asserts  
that second tab button is hittable after app enters  
collection view.
```

if stuck, check out branch “step3” from git:
\$> git checkout step3

<https://github.com/FrenchKit/UIAutomationClassroom>

Writing your test

<https://github.com/FrenchKit/UIAutomationClassroom>

Writing your test

- Create 1 test method for 1 logical assertion
- Exception: testing user flow, smoke tests
- Structure your test: arrange, act, assert
- Use “waitUntil()” instead of “sleep()”
- Use screen objects

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 5: Make it fail

ArrangeActAssertExample.swift:

```
// TODO: STEP 5: Introduce a bug in the app that  
breaks this test
```

if stuck, check out branch “step5” from git:
\$> git checkout step5

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 6: Make it fail

ArrangeActAssertExample.swift:

```
// TODO: STEP 6: extract the following steps into a  
separate test method
```

```
if stuck, check out branch "step6" from git:  
$> git checkout step6
```

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 7: Make it fail

ArrangeActAssertExample.swift:

```
// TODO: STEP 7: refactor the code using screen  
objects.
```

if stuck, check out branch “step7” from git:
\$> git checkout step7

<https://github.com/FrenchKit/UIAutomationClassroom>

Launch arguments

<https://github.com/FrenchKit/UIAutomationClassroom>


Launch arguments

- Process arguments, environment variables
- Turn on/off “ui testing” mode
- Disable 3rd-party frameworks (i.e., a/b testing)
- Use mock server instead of real server in tests
- Reset the app to clean state

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 8: Launch argument

ApplicationObjectTestCase.swift:

```
// TODO: STEP 8: Write a test that expects seeing a  
 emoji on the main screen  
// TODO: modify the app so that it displays a  
French flag when uiTest flag is passed to it.
```

if stuck, check out branch “step8” from git:
\$> git checkout step8

<https://github.com/FrenchKit/UIAutomationClassroom>

Table views and collection views

<https://github.com/FrenchKit/UIAutomationClassroom>

Table views and collection views

- Cells are “opaque” to UI accessibility system by default
- UI Accessibility container informal protocol
- Only visible cells on the screens accessible through XCTest
- Use accessibility identifier with row or item number to distinguish between cells
- You can use screen objects, too!

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 9: Table and collection

`ScrollTillVisibleTests.swift:`

`// TODO: STEP 9: write a test that opens collection
view screen and scrolls till 11th item`

`if stuck, check out branch “step9” from git:`

`$> git checkout step9`

<https://github.com/FrenchKit/UIAutomationClassroom>

Localizations

<https://github.com/FrenchKit/UIAutomationClassroom>

Localizations

- Use accessibilityIdentifier instead of UIButton's text or UILabel's text in tests
- Reuse localization keys from your main bundle

<https://github.com/FrenchKit/UIAutomationClassroom>

Step 10: Localization

LocalizationExample.swift:

```
// TODO: STEP 10: write a test method that asserts  
second tab's text is in French
```

if stuck, check out branch “step10” from git:
\$> git checkout step10

<https://github.com/FrenchKit/UIAutomationClassroom>

Working with network

<https://github.com/FrenchKit/UIAutomationClassroom>

Working with network

- You can stub URLSessionProtocol to return mock responses - I think, not feasible in XCUITest
- You can use a mock server running on your local machine

<https://github.com/FrenchKit/UIAutomationClassroom>

Resources

<https://github.com/FrenchKit/UIAutomationClassroom>

Resources

- Cloud testing:
 - <https://testcloud.xamarin.com/>
 - <https://bitbar.com/>
 - UI testing cheat sheet
 - <https://github.com/joemasilotti/UI-Testing-Cheat-Sheet>
- <https://github.com/FrenchKit/UIAutomationClassroom>