INTERMEDIATE

CORE DATA

HANDS-ON CHALLENGES

Intermediate Core Data

Luke Parham

Notice of Rights

Notice of Liability

Trademarks

Challenge #3: Adding More Tests

By Luke Parham

In the demo, you got a good feel for how testing units of your system works. Your challenge this time is to keep practicing what you've learned by adding a few more tests to the `CampSiteService`.

Here are the features you need to add:

1. Make sure that deleting a campsite actually removes the object and makes subsequent fetches return nil.
2. Campsites should be ordered sequentially and `getNextCampsiteNumber()` should return one more than the current highest numbered campsite. Write a test to make that the first campsite number starts at **1**.
3. Along these lines, write a test for when the highest number is something other than **0**.

Deleting Campsites

To get started, navigate to **CampSiteServiceTests.swift** and scroll to the bottom of the file.

Add the following test

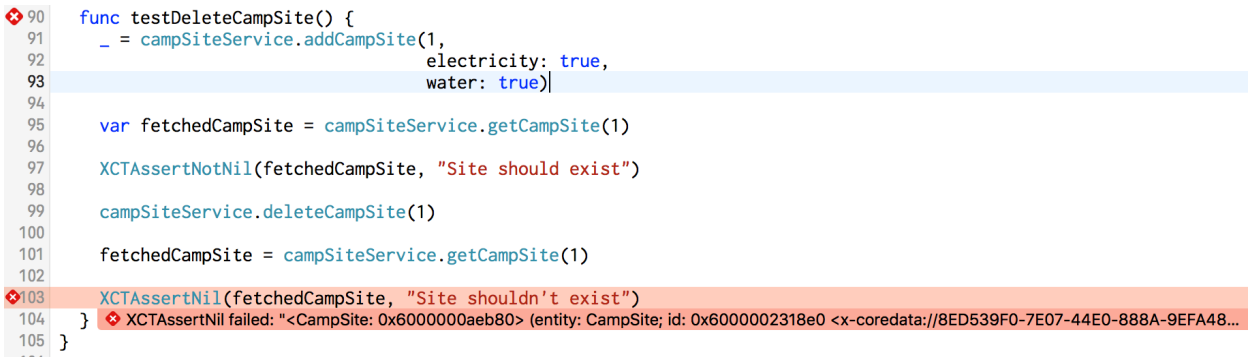
```
func testDeleteCampSite() {
_ = campSiteService.addCampSite(1,
                                electricity: true,
                                water: true)

var fetchedCampSite = campSiteService.getCampSite(1)

XCTAssertNotNil(fetchedCampSite, "Site should exist")

campSiteService.deleteCampSite(1)

fetchedCampSite = campSiteService.getCampSite(1)
```

```
  XCTAssertNil(fetchedCampSite, "Site shouldn't exist")
  }
```

Here, you create a campsite with an id of **1**, retrieve that campsite and make sure it exists.

Then, you call `deleteCampSite(_:)`, also with an id of **1** and try to fetch it again. Your assertion says that `fetchedCampSite` should be `nil`.

Run the tests with cmd+u to see your first failing test, which is just what you wanted!



To move to that coveted green state, navigate to `deleteCampSite(_:)` in **CampSiteService.swift**.

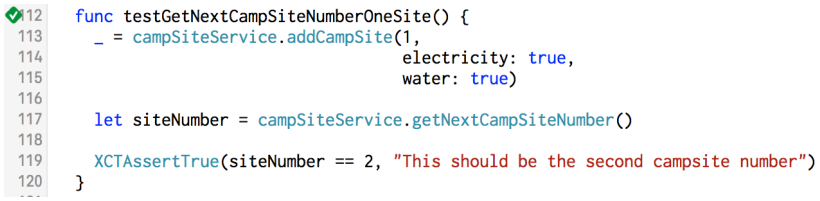
Replace the existing TODO comment with the following code:

```
guard let campSite = getCampSite(siteNumber) else { return }

managedObjectContext.delete(campSite)

do {
  try managedObjectContext.save()
} catch let error {
  print(error)
}
```

The implementation does what the method promises by looking up the image with the provided id and performing a delete on it.

Go back to **CampSiteServiceTests.swift** and run your tests again.

This time you should see a passing test.



Getting the Next Campsite Number

The next two tests will test `getNextCampSiteNumber()`.

Your first test will make sure that the numbering system starts at **1** if no other campsites have been added.

```
  func testGetNextCampSiteNumberNoSites() {
    let siteNumber = campSiteService.getNextCampSiteNumber()

    XCTAssertTrue(siteNumber == 1, "This should be the first campsite
number")
  }
```

Run the tests to once again see a failure. Go back to **CampSiteService.swift** and take a look at `getNextCampSiteNumber()`. Obviously **-1** will never be the right answer. Replace the current return with:

```
return 1
```

If making the test pass this way offends your sense of over-engineering, then good! It's important to remember that you should always try to do the simplest thing that can make the test pass and nothing more. There's no need to worry about fetching or counting objects at this point.

Run the tests to see another passing test.

Of course, that doesn't mean you're done yet. Another scenario to test is numbers greater than zero. Add the following test to round out your test coverage a little bit.

```
func testGetNextCampSiteNumberOneSiteGapFrom1() {
    _ = campSiteService.addCampSite(10,
                                    electricity: true,
                                    water: true)
    let siteNumber = campSiteService.getNextCampSiteNumber()

    XCTAssertTrue(siteNumber == 11, "This should be the second campsite
number")
}
```

By now you should know the drill. Run the test to see another failure.

As you may have guessed, head on back to `getNextCampSiteNumber()` to add a little more functionality.

With this test, there's no way around it. Add the following code before the return statement from before.

```
let sites = getCampSites()

if sites.count > 0 {
```

```
    let lastSiteNumber = sites.last!.siteNumber!
    return NSNumber(value: lastSiteNumber.intValue + 1)
  }
```

Run the tests to see them all passing. You've sucessfully improved the functionality of the app and improved its test coverage along the way.

And wouldn't ya know it, your `return 1` is still useful.