BEGINNING

CORE DATA

HANDS-ON CHALLENGES

Beginning Core Data

Luke Parham

Challenge #5: Deleting Users

By Luke Parham

In the demo you saw how to set up relationships between various types of managed objects. Now, your challenge is to add the ability to delete users to this app and have the bow ties react appropriately do the deletions.

Deleting Users

To get started, go to **UsersViewController.swift** and add

```
    func didDeleteUser()
```

to the `BowTieDelegate` protocol definition.

Then, add the following method at the bottom of the file.

```
override func tableView(_ tableView: UITableView,
    commit editingStyle: UITableViewCellEditingStyle,
    forRowAt indexPath: IndexPath) {
  guard let appDelegate =
      UIApplication.shared.delegate as? AppDelegate else {
    return
  }

  if editingStyle == .delete {
    let user = fetchedResultsController.object(at: indexPath)
    appDelegate.persistentContainer.viewContext.delete(user)

    do {
      try appDelegate.persistentContainer.viewContext.save()
      bowTieDelegate?.didDeleteUser()
      reloadData()
    } catch let error {
      print(error)
    }
  }
}
```

Here you use the built-in ability of table views to allow your users to delete cells by swiping on them from right to left and tapping the delete button.

Inside the conditional you call `delete(_:)` on `viewContext` and pass in the corresponding user.

Then, if the save is successful, you call `didDeleteUser()` on `bowTieDelegate` to propagate the event to the delegate.

Deleting Users

Next, navigate to **ViewController.swift**. Now that `UsersViewController` is trying to notify `ViewController` of the delete event, you'll implement the delegate method to react to it.

Go to the `BowTieDelegate` extension and add the following method

```
func didDeleteUser() {
    populate(bowtie: currentBowtie)
}
```

Since the UI only shows one Bowtie at a time, all you need to do is reload the current Bowtie just in case it was deleted as a result of the user being deleted.

Next, since it is possible the Bowtie for one of the tabs has been deleted, you need a way to indicate it has been destroyed.

Add the following method after the existing `populate(_:)` method.

```
func populateEmptyBowtie() {
    imageView.image = nil
    nameLabel.text = ""
    ratingLabel.text = ""
    timesWornLabel.text = ""

    lastWornLabel.text = ""
    favoriteLabel.isHidden = true
    ownerButton.setTitle("", for: .normal)
    view.tintColor = UIColor.gray
}
```

Finally, you need to conditionally load the correct UI in two places.

Go to `viewDidLoad()` and replace the contents of the `do` block with the following.

```
let results = try managedContext.fetch(request)
if results.count == 0 {
    populateEmptyBowtie()
} else {
    currentBowtie = results.first
    populate(bowtie: currentBowtie)
}
```

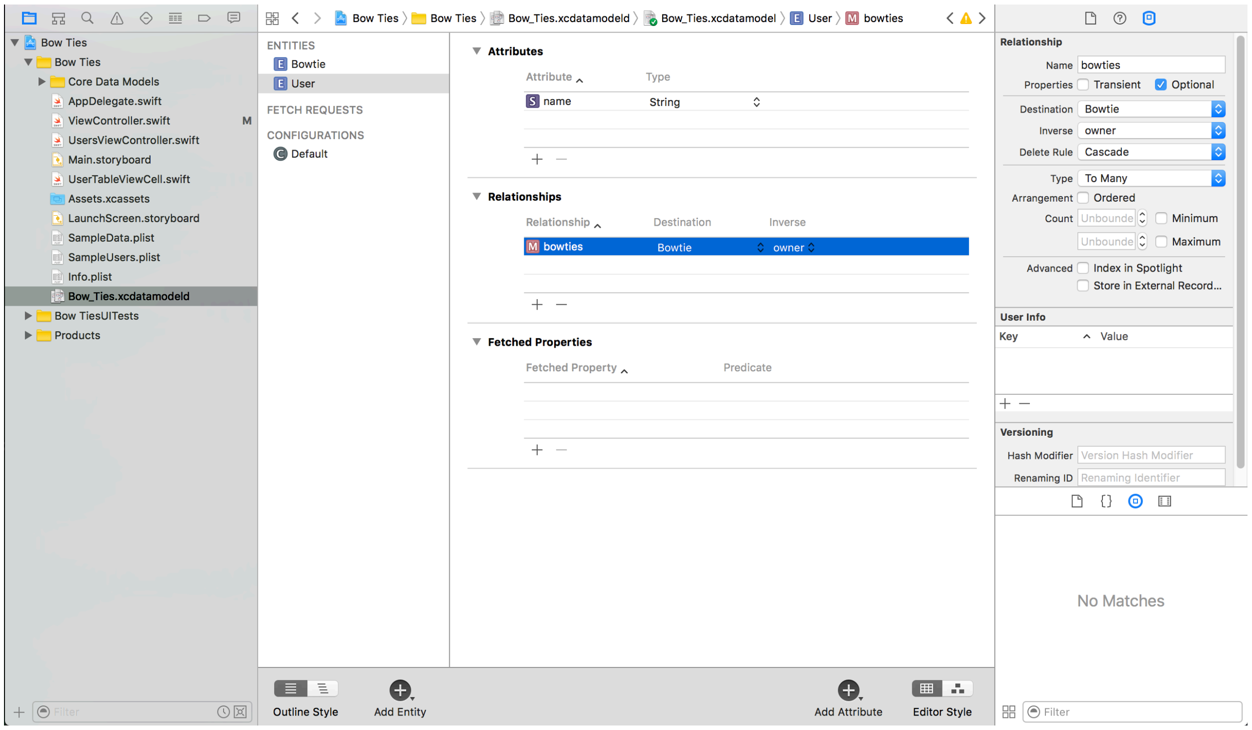
Then go to `segmentedControl(_:)` and do the same thing.

Now, the UI will be ready to react to the deletion of users correctly.

Adding a Cascade Delete

To play with the various deletion options, go to Bow_Ties.xcdatamodeld and open the Utilities panel on the right.

Click on the `User` entity, and then click on the `bowties` relationship. Finally, choose the **Data Model Inspector** to view attributes of the relationship.



Relationship

Name	bowties
Properties	☐ Transient ☑ Optional
Destination	Bowtie
Inverse	owner
Delete Rule	Cascade
Type	To Many
Arrangement	☐ Ordered
Count	Unbounded ☐ Minimum
	Unbounded ☐ Maximum
Advanced	☐ Index in Spotlight
	☐ Store in External Record...

User Info

Key	Value

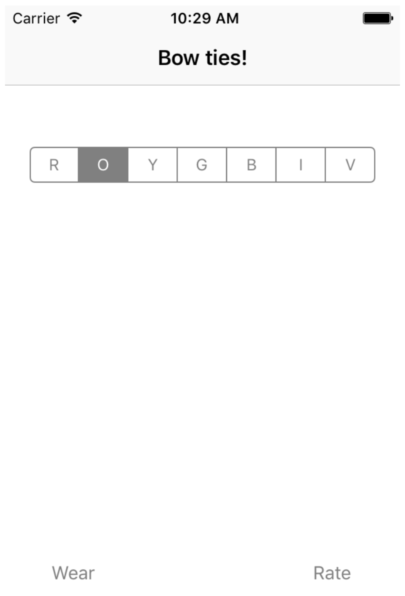
Versioning

Hash Modifier	Version Hash Modifier
Renaming ID	Renaming Identifier

By default, the **Delete Rule** is set to Cascade. As you saw in the the video, this means that when a `User` is deleted bowties that have that user set as their owner will have that property set to `nil`. You can build and run at his point to see this in action.

If you set a `Bowtie`'s owner to one of the users and then delete that user, the `Bowtie`'s owner will automatically show "None".

For a little more fun, change the delete rule to **Delete**. Now build and run and do the same thing to see `BowTie`'s getting deleted along with their owners!


Carrier
10:29 AM
Bow ties!
R
O
Y
G
B
I
V
Wear
Rate