EE443 / EE593
Mobile Application Development

Prof. Phillip De Leon

Lecture 25:
Chapter 36: URL Session

Prof. Phillip De Leon

Programming To-Do List
Chapter 36

- URLSession API (what the pros use!)
- Handle HTTP status codes (200, 404, 999)
- Parse JSON data
- Error handling
- Cancel ongoing search operations
- UISegmentedControl to specify search categories
- Download artwork for table view cells

URLSession

- Closure-based API—just pass a closure with code to perform once response from server is received, i.e. *completion handler*
- Replace old networking code with

```
let url = iTunesURL(searchText: searchBar.text!)
let session = URLSession.shared
let dataTask = session.dataTask(with: url, completionHandler: { data, response, error in
  if let error = error {
    print("Failure! \(error)")
  } else {
    print("Success! \(response!)")
  }
})
dataTask.resume()
```

Create shared URLSession (default config)

Create a data task for fetching URL contents. Completion handler invoked when data task receives server response

data holds JSON data fetched from server
response holds server's response code
error holds error object if something goes wrong

Starts data task which sends request to server on background thread

Programming To-Do List
Chapter 36

- ~~URLSession API (what the pros use!)~~
- Handle HTTP status codes (200, 404, 999)
- Parse JSON data
- Error handling
- Cancel ongoing search operations
- UISegmentedControl to specify search categories
- Download artwork for table view cells

Handle HTTP Status Codes

- Codes: 200 (OK), 403 (Forbidden), 404 (web page Not Found)
- Change completion handler with

```
if let error = error {
  print("Failure! \(error)")
} else if let httpResponse = response as? HTTPURLResponse,
  httpResponse.statusCode == 200 {
  print("Success! \(data!)")
} else {
  print("Failure! \(response!)")
}
```

error like "no network"

data object can't be printed so we just get byte count

other http status codes

Programming To-Do List
Chapter 36

- ~~URLSession API (what the pros use!)~~
- ~~Handle HTTP status codes (200, 404, 999)~~
- Parse JSON data
- Error handling
- Cancel ongoing search operations
- UISegmentedControl to specify search categories
- Download artwork for table view cells

Parse JSON Data

- In completion handler, replace print("Success! \(data!)") with

```
if let data = data {
  self.searchResults = self.parse(data: data)
  self.searchResults.sort(by: <)
  DispatchQueue.main.async {
    self.isLoading = false
    self.tableView.reloadData()
  }
  return
}
```

Unwrap the data object/closure param

Call *parse(data:)* to turn JSON into searchResult objects

Programming To-Do List
Chapter 36

- ~~URLSession API (what the pros use!)~~
- ~~Handle HTTP status codes (200, 404, 999)~~
- ~~Parse JSON data~~
- Error handling
- Cancel ongoing search operations
- UISegmentedControl to specify search categories
- Download artwork for table view cells

Error Handling

- At the bottom of completion handler, add the following (we only go here if something went wrong)

```
DispatchQueue.main.async {
  self.hasSearched = false
  self.isLoading = false
  self.tableView.reloadData()
  self.showNetworkError()
}
```

Let the user know about the problem

Programming To-Do List
Chapter 36

- ~~URLSession API (what the pros use!)~~
- ~~Handle HTTP status codes (200, 404, 999)~~
- ~~Parse JSON data~~
- ~~Error handling~~
- Cancel ongoing search operations
- UISegmentedControl to specify search categories
- Download artwork for table view cells

Cancel Ongoing Search Operations

- Suppose a search takes a long time and the user starts a second search. What could happen?
- Add an ivar (optional because we won't have a data task until user searches)

```
var dataTask: URLSessionDataTask?
```

- Unwrap dataTask in searchBarSearchButtonClicked(_ searchBar: UISearchBar)

```
dataTask?.resume()
```

- Before we set isLoading, add the following to cancel any existing search before we search

```
dataTask?.cancel()
```

Cancel Ongoing Search Operations

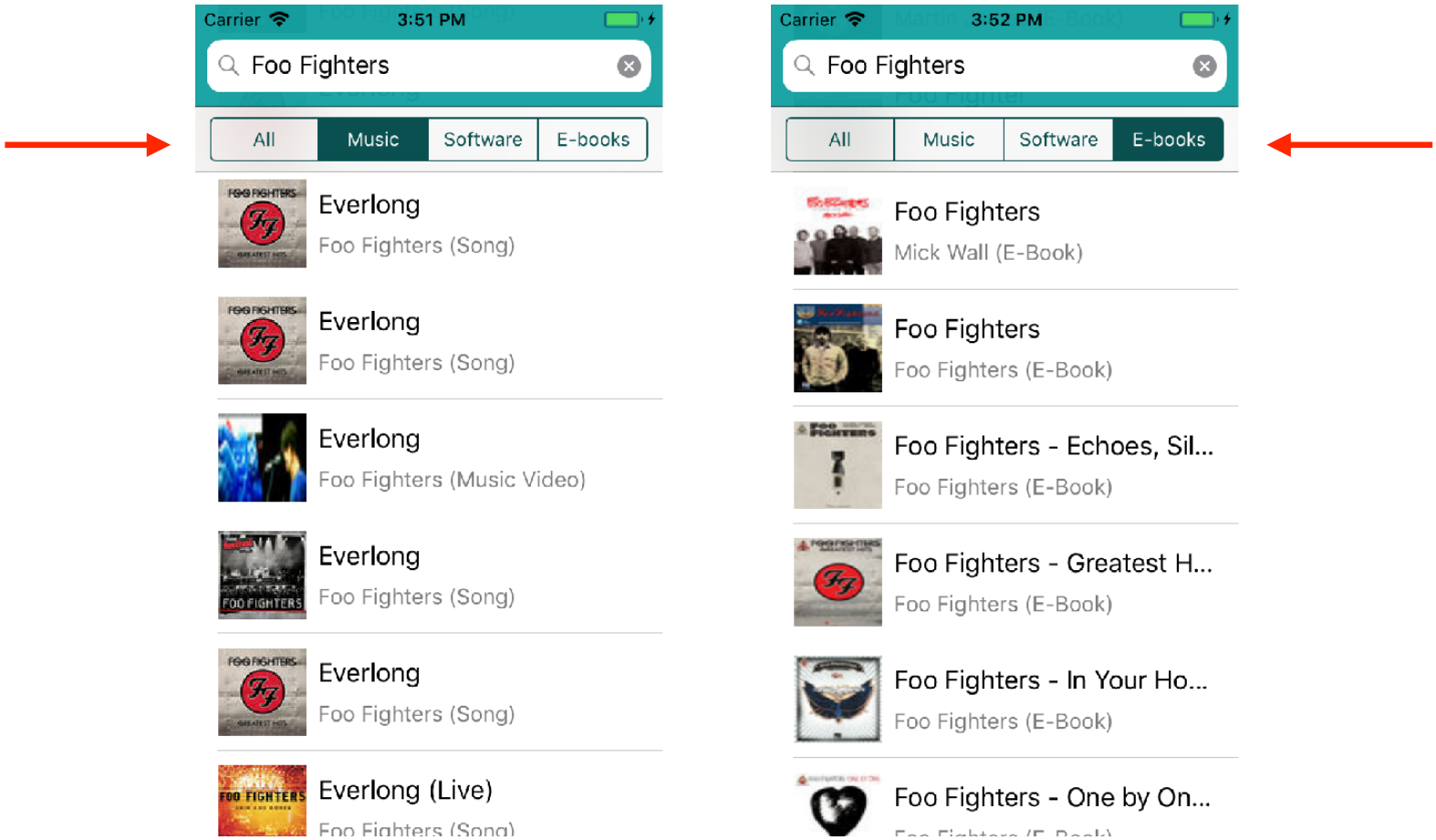
- Even though the data task gets canceled, we still invoke the completion handler (see code) with an Error object with error code -999
- We just need to ignore this error and exit the closure
- Modify the code block

```
if let error = error as NSError?, error.code == -999 {
  return
} else if let httpResponse = ...
```

Programming To-Do List
Chapter 36

- ~~URLSession API (what the pros use!)~~
- ~~Handle HTTP status codes (200, 404, 999)~~
- ~~Parse JSON data~~
- ~~Error handling~~
- ~~Cancel ongoing search operations~~
- UISegmentedControl to specify search categories
- Download artwork for table view cells

Specifying Search Categories



Use the Segmented Control

- Since tapping Search button on keyboard or selecting a Segment triggers a search, rename searchBarSearchButtonClicked(_:) to performSearch()
- Add:

```
func searchBarSearchButtonClicked(_ searchBar: UISearchBar) {
  performSearch()
}
```

- Fix:

```
@IBAction func segmentChanged(_ sender: UISegmentedControl) {
    performSearch()
}
```

Use the Segmented Control

- Change iTunesURL(searchText:) to accept Segment Control's Int as input param and build URL

```
func iTunesURL(searchText: String, category: Int) -> URL {
  let kind: String
  switch category {
    case 1: kind = "musicTrack"
    case 2: kind = "software"
    case 3: kind = "ebook"
    default: kind = ""
  }
  let encodedText = searchText.addingPercentEncoding(withAllowedCharacters:
      CharacterSet.urlQueryAllowed)!
  let urlString = "https://itunes.apple.com/search?" +
      "term=\(encodedText)&limit=200&entity=\(kind)"
  let url = URL(string: urlString)
  return url!
}
```

Convert Segmented Control's Int selection to kind of search (String)

Add search param into URL

Use the Segmented Control

- In performSearch() include the new input param when getting iTunesURL

```
let url = iTunesURL(searchText: searchBar.text!, category:
segmentedControl.selectedSegmentIndex)
```

Programming To-Do List
Chapter 36

- ~~URLSession API (what the pros use!)~~
- ~~Handle HTTP status codes (200, 404, 999)~~
- ~~Parse JSON data~~
- ~~Error handling~~
- ~~Cancel ongoing search operations~~
- ~~UISegmentedControl to specify search categories~~
- Download artwork for table view cells

Download Artwork for TableViewCell

- JSON data does not contain images, e.g. Album Art but rather URLs to images
- We'll use another URLSession to fetch the image
- We'll extend UIImageView class to download image and automatically display it via the image view on the table view cell
- Refactor code to prepare. Pull code piece from tableView(_:cellForRowAt:) and make it a method in SearchResultCell.swift. Call new method from tableView(_:cellForRowAt:)

```
func configure(for result: SearchResult) {
  nameLabel.text = result.name
  if result.artistName.isEmpty {
    artistNameLabel.text = "Unknown"
  } else {
    artistNameLabel.text = String(format: "%@ (%@)", result.artistName, result.type)
  }
}
```

Download Artwork for TableViewCell

```
import UIKit
extension UIImageView {
  func loadImage(url: URL) -> URLSessionDownloadTask {
    let session = URLSession.shared
    let downloadTask = session.downloadTask(with: url,
                                completionHandler: { [weak self] url, response, error in
      if error == nil, let url = url,
        let data = try? Data(contentsOf: url),
        let image = UIImage(data: data) {
        DispatchQueue.main.async {
          if let weakSelf = self {
            weakSelf.image = image
          }
        }
       }
    })
    downloadTask.resume()
    return downloadTask
  }
}
```

Extend the UIImageView class

Create a shared URL session and create a downloadTask

If no error and we have a URL, load image file into a data object, and create an image from it

Put image in UIImageView's image property. Do it on main thread because this is UI code

Start download task
Return URLSessionDownloadTask object to the caller so app can cancel()

* Please see text for details on [weak self]

Use the Image Downloader Extension

- Add an ivar (SearchResultCell.swift) to get reference to the image downloader

```
var downloadTask: URLSessionDownloadTask?
```

- Add to the end of configure(for:): Use placeholder file while image is downloading

```
artworkImageView.image = UIImage(named: "Placeholder")
if let smallURL = URL(string: result.imageSmall) {
  downloadTask = artworkImageView.loadImage(url: smallURL)
}
```

- Since we cannot download files over http (since iOS9, only via https) we have to override this security feature. We'll modify the Info.plist…