1. One of the big things that separates working on an iOS device from desktop is always working in full screen.
2. Multiple View

Scene: Each view in an app is called a scene on a storyboard.

Segues: The arrows indicate transitions between the views, the segue s an object that is created to prepare the new view and pass data between views.

1. Storyboards can layout custom table view cell

In some case, the default table view cell doesn’t satisfy our requirement, we need customize the cell.

* Create a new class like AlbumTableViewCell which is a subclass of UITableViewCell.
* Go storyboard to use your custom table view cell instead of the default one. Change style to “Custom”, and fill in your custom class name.
* Then back to storyboard to define the UI.
* Change master view function “cellForRowAtIndexPath”

1. Change default detail view controller

By default, the detail view controller is UIViewController, sometimes we want to use a more specific view controller, like UITableViewController.

* Change class to be subclass of UITableViewController
* Remove the detail controller scene
* Drag a talbe view controller onto the storyboard for the detail view and set its class to the detail view.
* Design the rest of the view dynamically. Content -> Static Cells
* Add Segue to connect master view row to detail view.
* Code related change in prepareForSegue to prepare data.

1. Property list for data

Plist stands for property list and has been around for quite a while with OS X.

NSString \*path = [[NSBundle mainBundle] pathForResource: @”AlbumArray” ofType: @”plist”];

NSArray \*defaultAlbumPlist = [NSArray arrayWithContentsOfFileL path];

For(NSDictionary \*albumInfo in defaultAlbumPlist) {}

1. Core data

As plist can only hold string data, we need a more complicated way for loading/saving data.

* The ability to load and save your objects.
* Different ways to store your data
* Memory management with undo and redo.

There are some terminology used in core data:

* Entity: something you want to store in database. It contains attributes, also relationships between entities in some complex case.

Core Data describes entities with a **Managed Object Model**

* Inside of Core Data is a stack of three critical pieces: the **Managed Object Context**, the **Persistent Store Coordinator**, and the **Persistent Object Store**
  + **Managed Object Context**: keeps track of all the entities.
  + **Persistent Store Coordinator**: keep track of the **Persistent Object Stores** . The stores actually know how to read and write the data.
* Xcode can help you to create ObjectiveC class from the **Managed Object Model**

1. Fetching Data using Core Data

* configureCell is where the data is loaded into the view.
* The Fetched Results Controller is code that comes with Core Data to help get result from the database.

1. Ss
2. S
3. S
4. S
5. Ss
6. S
7. Ss
8. S