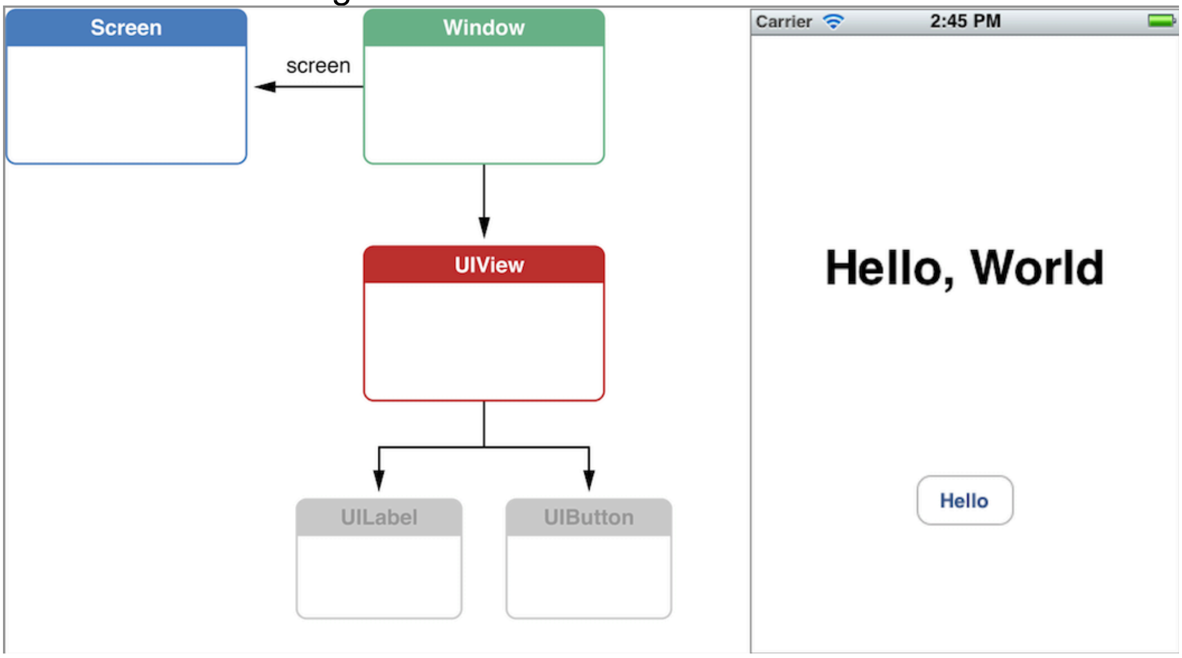
View Controller Basics

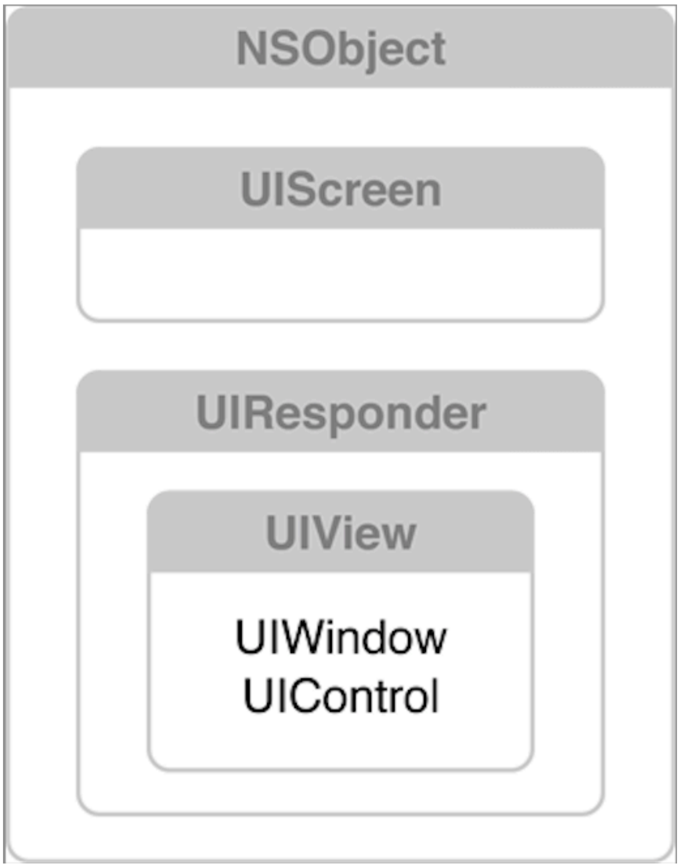
A key part of any view controller's implementation is to manage the views used to display its content. Most view controllers also communicate and coordinate with other view controllers when transitions occur.

Screens, Windows, and Views Create Visual Interfaces

A window with its target screen and content views.



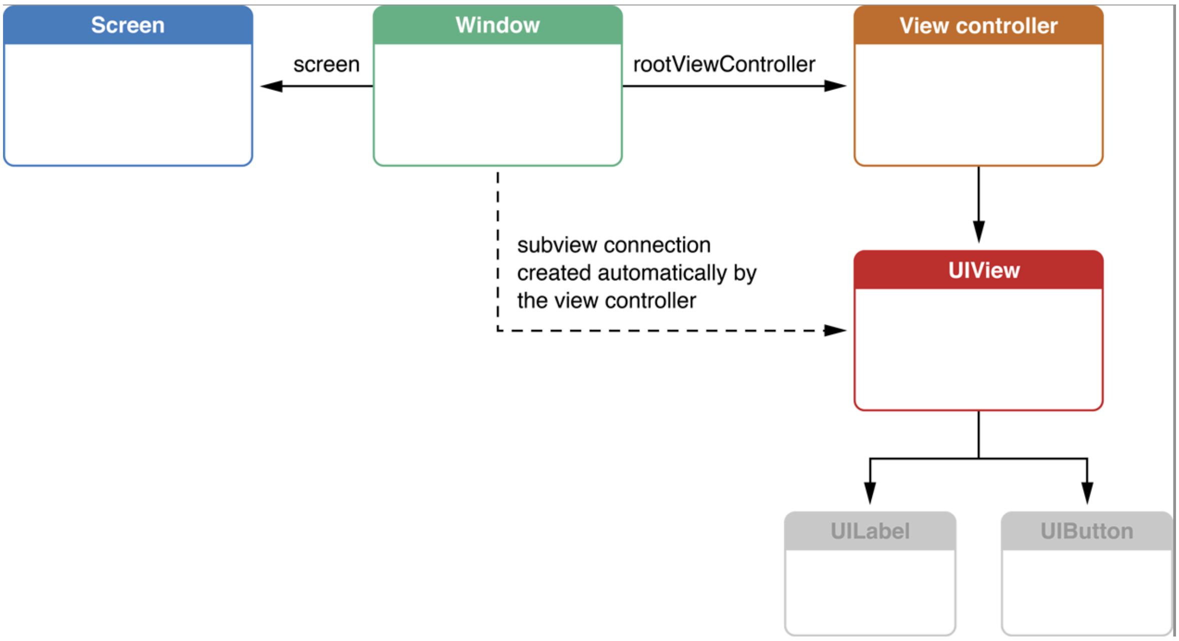
Classes in the view system



View Controllers Manage Views

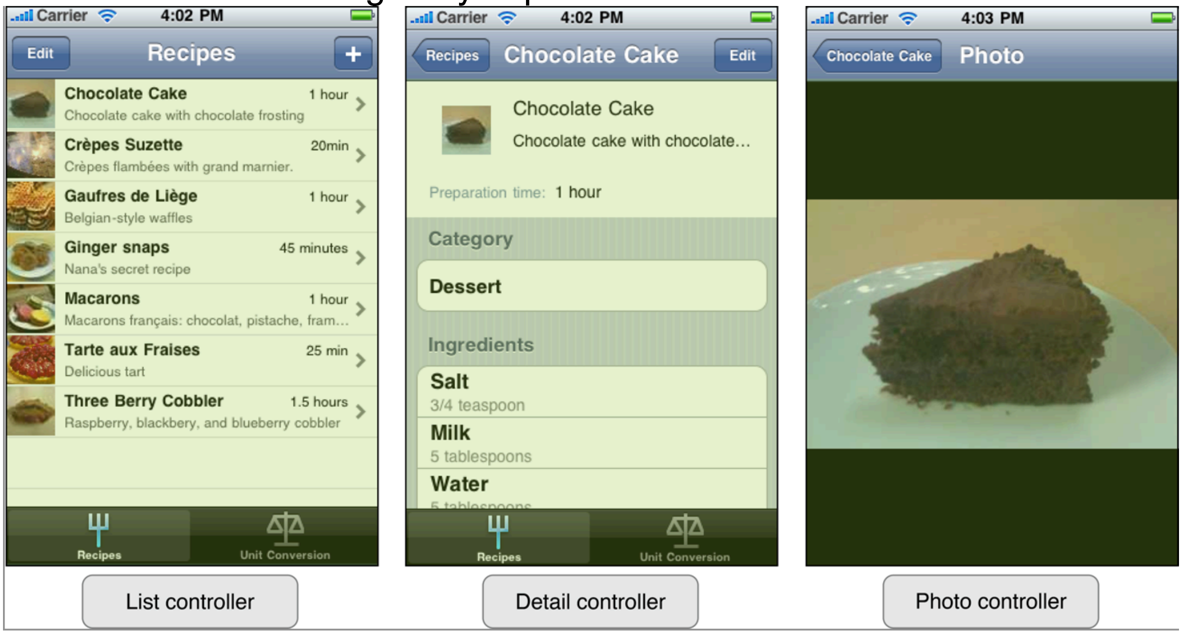
1. Each view controller organizes and controls a view, this view is often the root view of a view hierarchy. View controllers are controller objects in MVC design pattern but a view controller also has specific tasks iOS expects it to perform.

A view controller attached to a window automatically adds its view as a subview of the window



2. A view controller is the natural place to coordinate actions of its connected views.

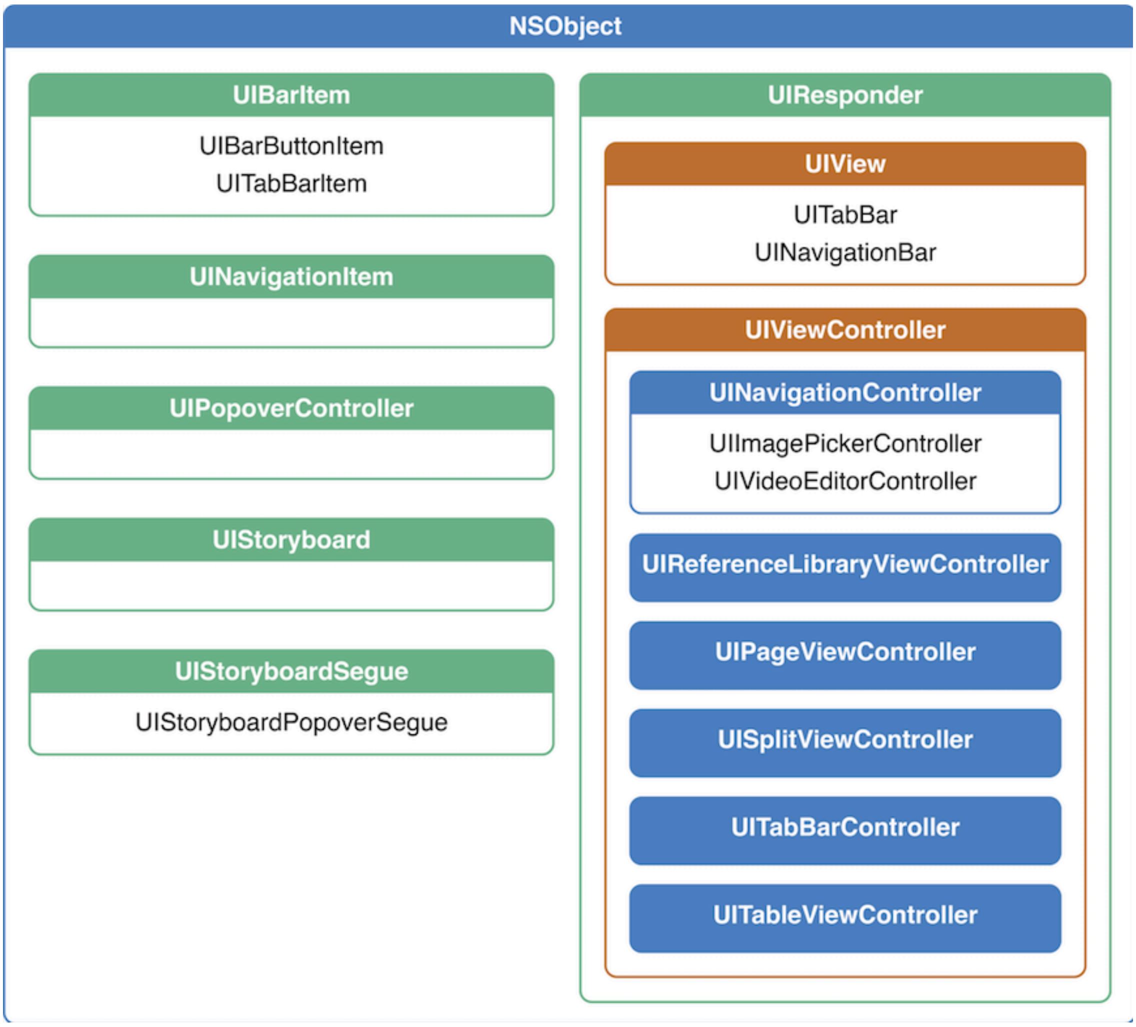
3. Distinct views managed by separate view controllers



This example demonstrates a few factors common to view controllers:
=> Every view is controlled by only one view controller. When a view is assigned to a view controller’s view property, the view controller owns it.
=> Each view controller interacts with a subset of your app’s data.
=> Because each view controller provides only a subset of the user experience, the view controllers must communicate with each other to make this experience seamless. They may also communicate with other controllers, such as data controllers or document objects.

A Taxonomy of View Controllers

1. View controller classes in UIKit

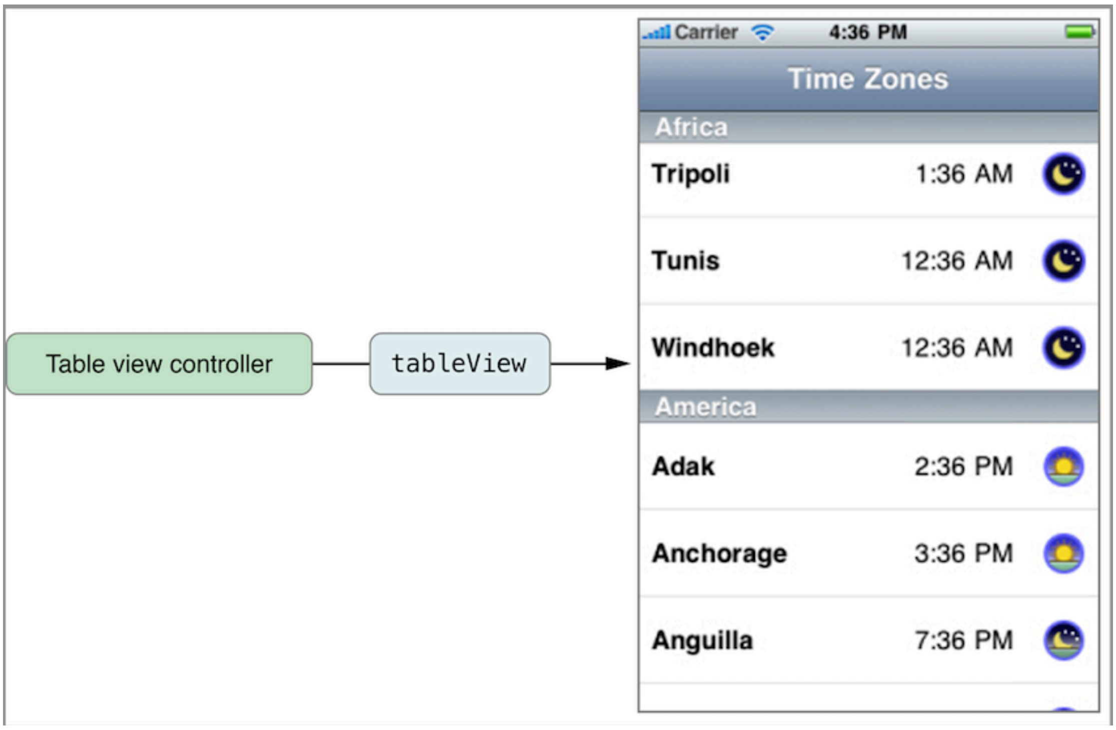


2. The view controller is divided into two categories —— content view controllers and container view controllers.

> Content View Controllers Display Content
1. A content view controller presents content on the screen using a view or a group of a view organized into view hierarchy.

2. Each content view controller is responsible for managing all the views in a single view hierarchy.

>> About Table View Controllers
1. An example that using a table view controller. (UITableViewController)



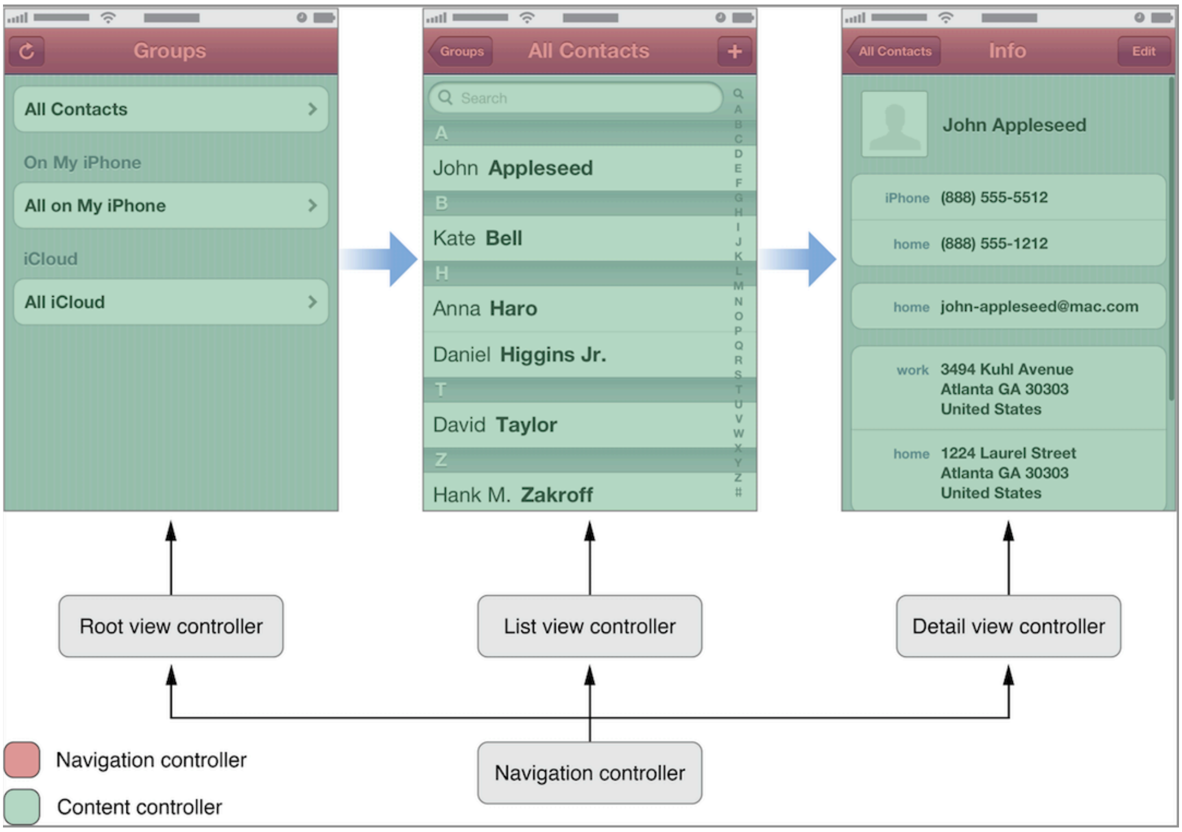
Africa	
Tripoli	1:36 AM
Tunis	12:36 AM
Windhoek	12:36 AM
America	
Adak	2:36 PM
Anchorage	3:36 PM
Anguilla	7:36 PM

> Container View Controllers Arrange Content of Other View Controllers
1. A container view controller contains content owned by other view controllers. These other view controllers are assigned to the container view controller as its children. This combination of controllers establishes a controller hierarchy.

2. Each type of container view controller establishes a user interface that its children operate in.

>> About Navigation Controllers
1. A navigation controller presents data that is organized hierarchically and is an instance of the UINavigationController class.

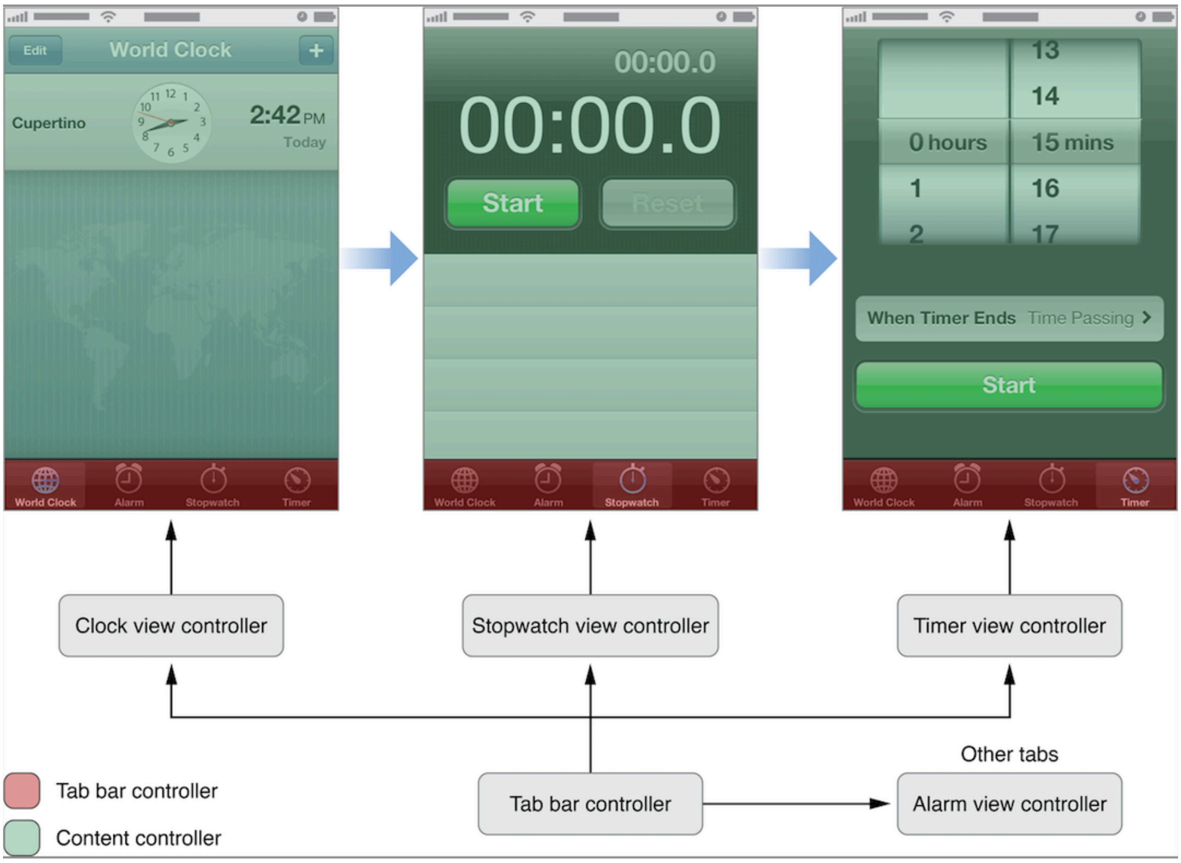
2. Navigating hierarchical data



>> About Tab Bar Controllers

1. A tab bar controller is a container view controller that you use to divide your app into two or more distinct modes of operation.

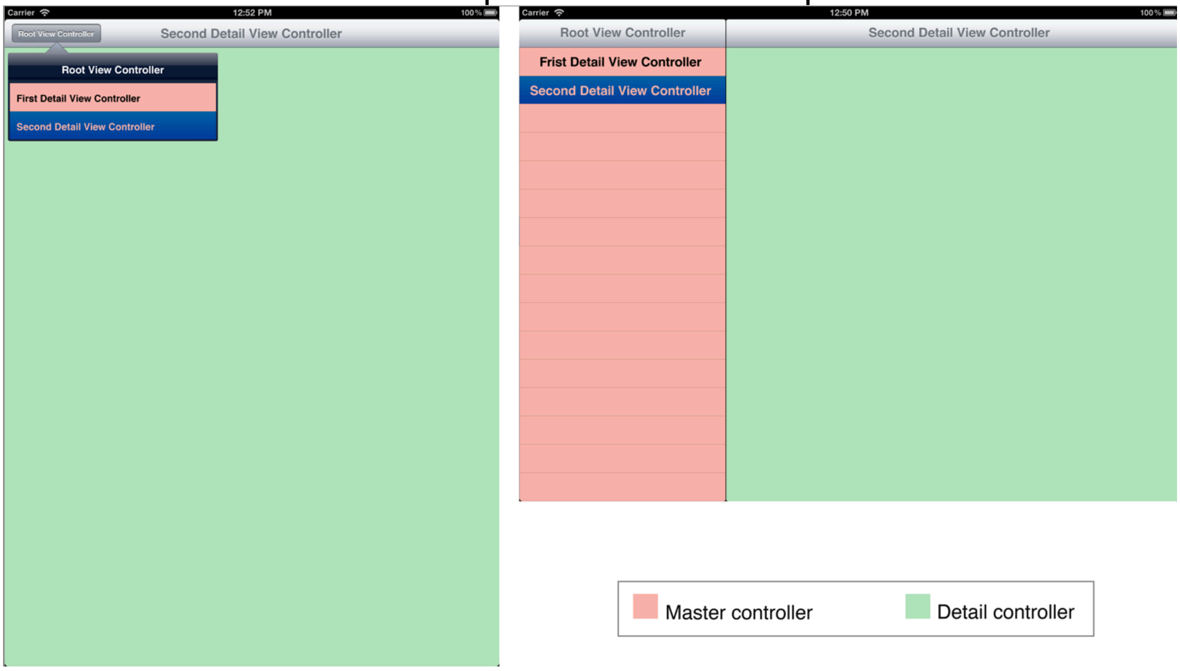
2. The figure below shows several modes of the Clock app along with the relationships between the corresponding view controllers.



>> About Split View Controllers

1. A split view controller divides the screen into multiple parts, each of which can be updated separately.

2. A master-detail interface in portrait and landscape modes

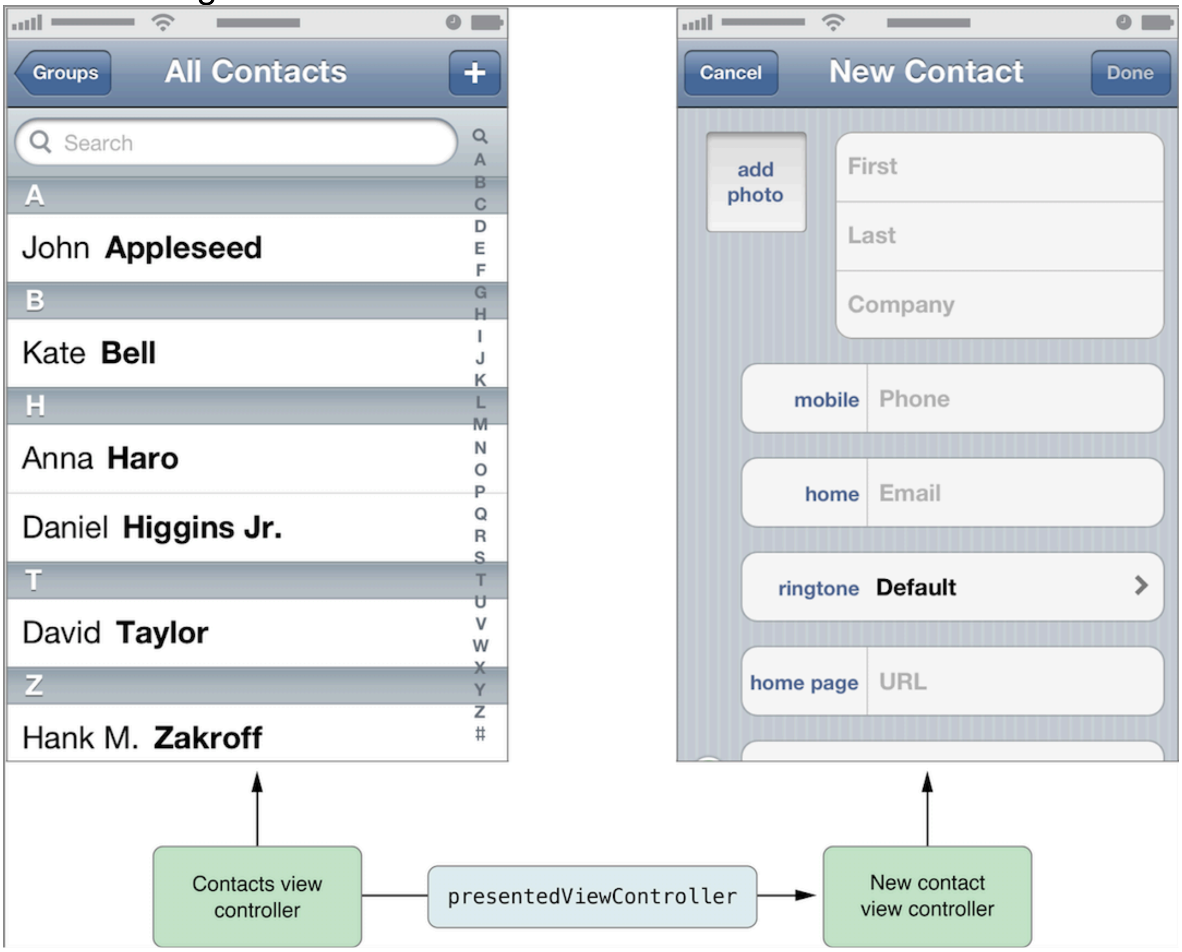

Carrier
12:52 PM
100%
Root View Controller
Second Detail View Controller
Root View Controller
First Detail View Controller
Second Detail View Controller
Carrier
12:50 PM
100%
Root View Controller
Second Detail View Controller
Frist Detail View Controller
Second Detail View Controller
Master controller
Detail controller

>> About Popover Controllers
A popover controller is not actually a container; it does not inherent from UIViewController at all. But, in practice, a popover controller is similar to a container, so you apply the same programming principles when you use them.

>> About Page View Controllers
A page view controller is a container view controller used to implement a page layout.

A View Controller's Content Can Be Displayed in Many Ways
1. For a view controller's contents to be visible to the user, it must be associated with a window. There are many ways you can do this in your app:
=> Make the view controller a window's root view controller.
=> Make the view controller a child of a container.
=> Show the view controller in a popover control.
=> Present it from another view controller.

2. Presenting a view controller.

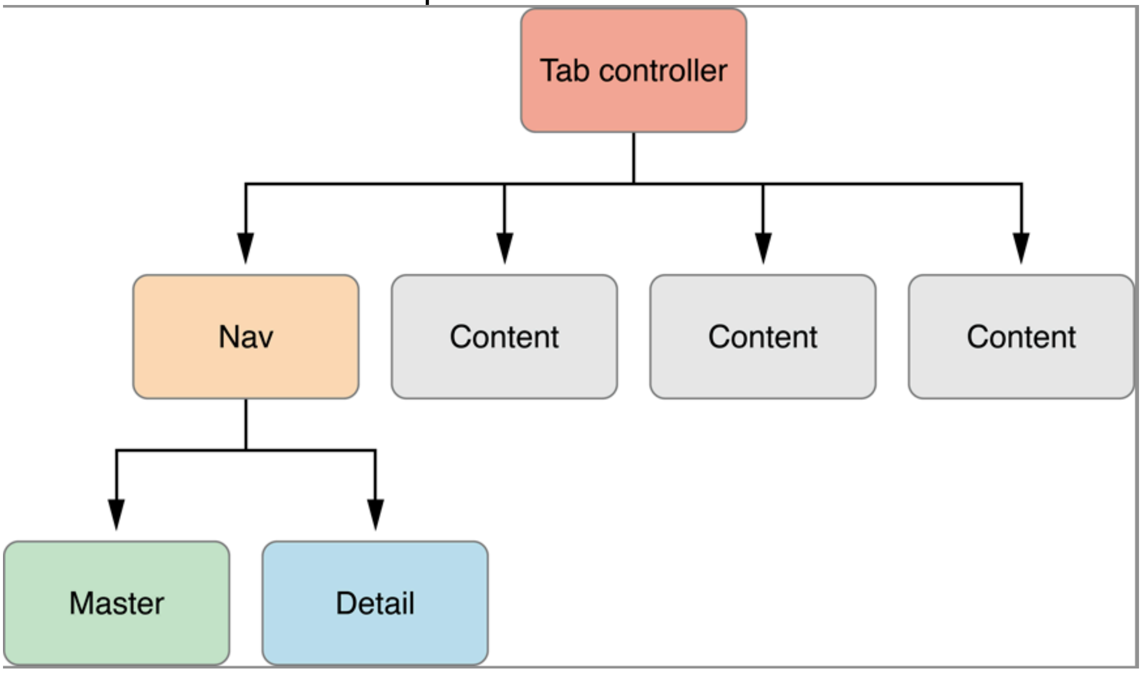


A presented view controller is not a specified type of view controller.

View Controllers Work Together to Create an App’s Interface

> Parent-Child Relationships Represent Containment

1. Parent-Child relationships

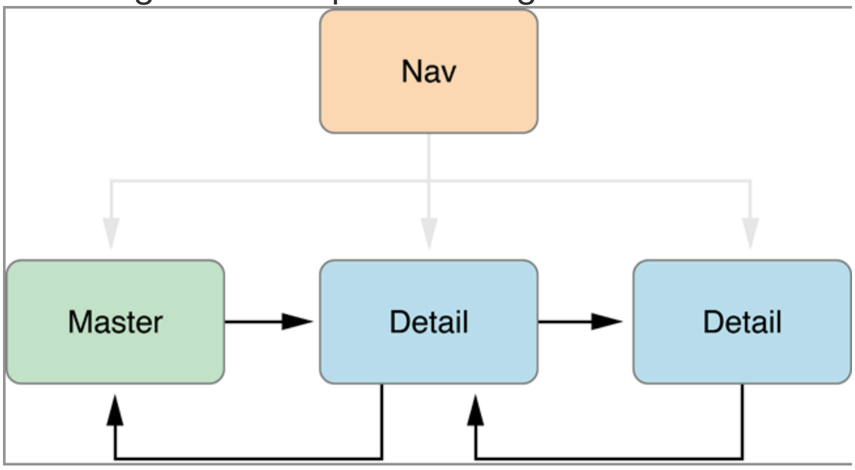


2. The area each view controller fills is determined by its parents. The root view controller’s area is determined by the window.

3. This combination of views and view controllers also establishes the responder chain for events handled by your app.

> Sibling Relationships Represent Peers Inside a Container

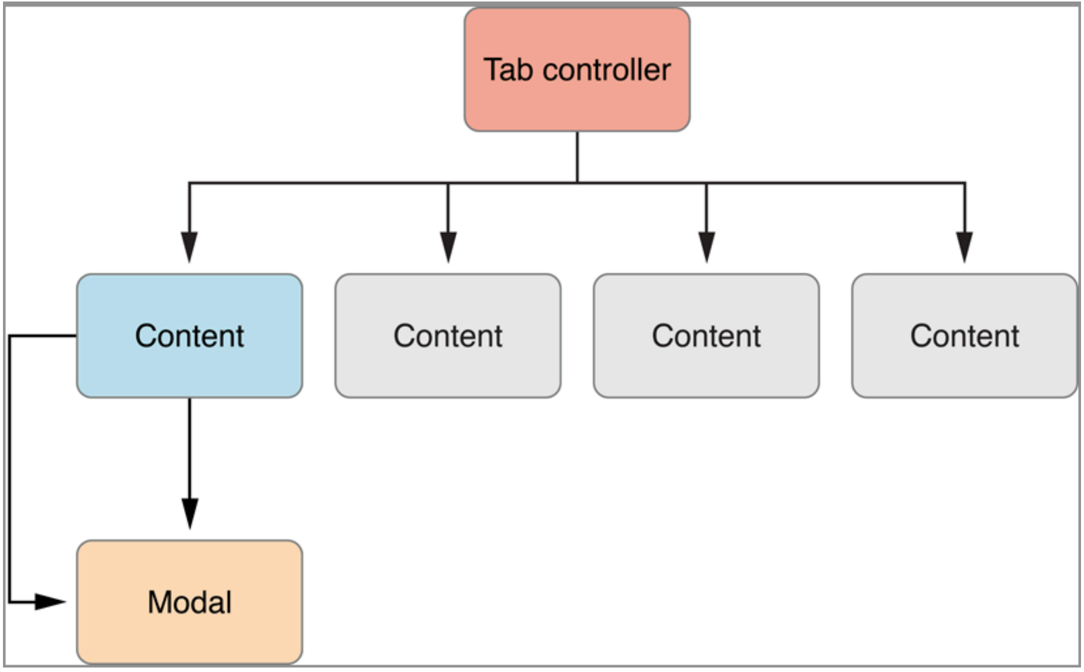
1. Sibling relationships in a navigation controller



> Presentation Represents a Transient Display of Another Interface

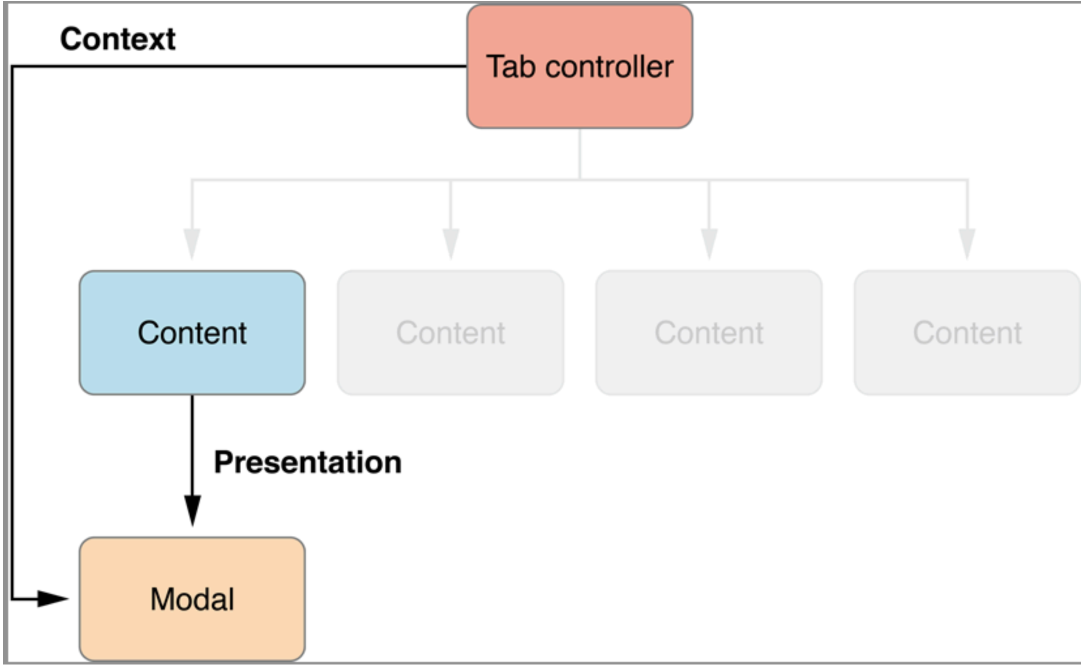
1. A view controller presents another view controller when it wants that view controller to perform a task.

2. Modal presentation by a content view



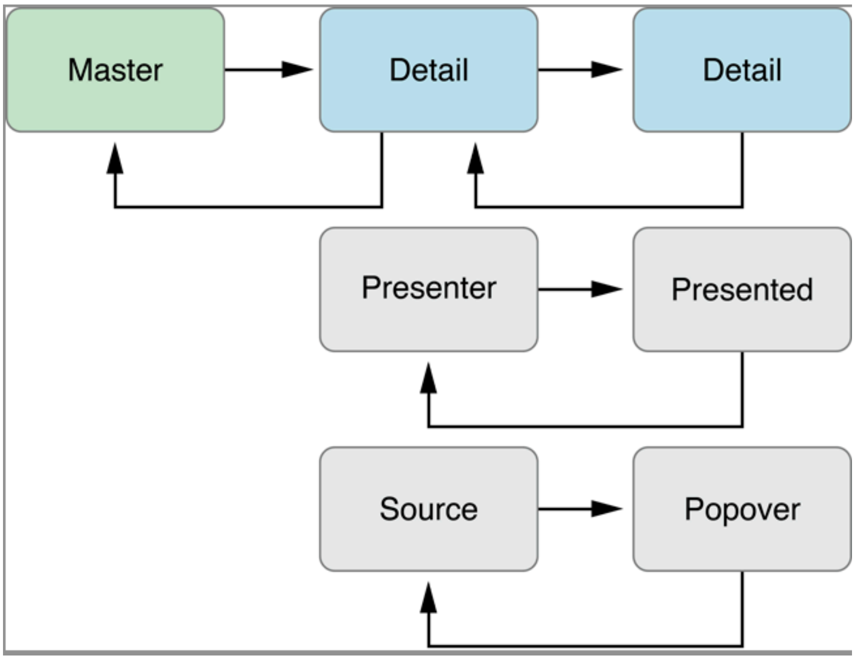
3. When a view controller is presented, the portion of the screen it covers is defined by a presentation context provided by another view controller.

4. The actual presentation is performed by the root view controller.



> Control Flow Represents Overall Coordination Between Content Controllers

1. The first view controller, known as the source view controller directs as second view controller, the destination view controller. If the destination view controller presents data to the user, the source view controller usually provides that data.
Communication between source and destination view controllers.



2. The destination view controller provides properties used to configure its data and presentation. If the destination view controller needs to communicate with view controllers preceding it in the chain, it uses delegate.

Storyboards Help You Design Your User Interface

1. When you implement your app using storyboards, you use interface builder to organize your app's view controllers and any associated views.

2. A **scene** represents an onscreen content area that is managed by a view controller.

3. You create **relationships** between scenes in the same storyboard. Relationships are expressed visually in a storyboard as a connection arrow from one scene to another. Two important kind of relationships:
=> **Containment** represents a parent-child relationship between two scenes.
=> A **segue** represents a visual transition from one scene to another.

4. Different kinds of segues provide the common transitions needed between two different view controllers:
=> A push segue pushes the destination view controller onto a navigation controller’s stack.
=> A modal segue presents the destination view controller.
=> A popover segue displays the destination view controller in a popover.
=> A custom segue allows you to design your own transition to display the destination view controller.