Resource Management in View Controllers

Initializing a View Controller

When a view controller is first instantiated, it creates or loads objects it needs through its lifetime.

> Initializing a View Controller Loaded from a Storyboard

> Initializing View Controllers Programmatically

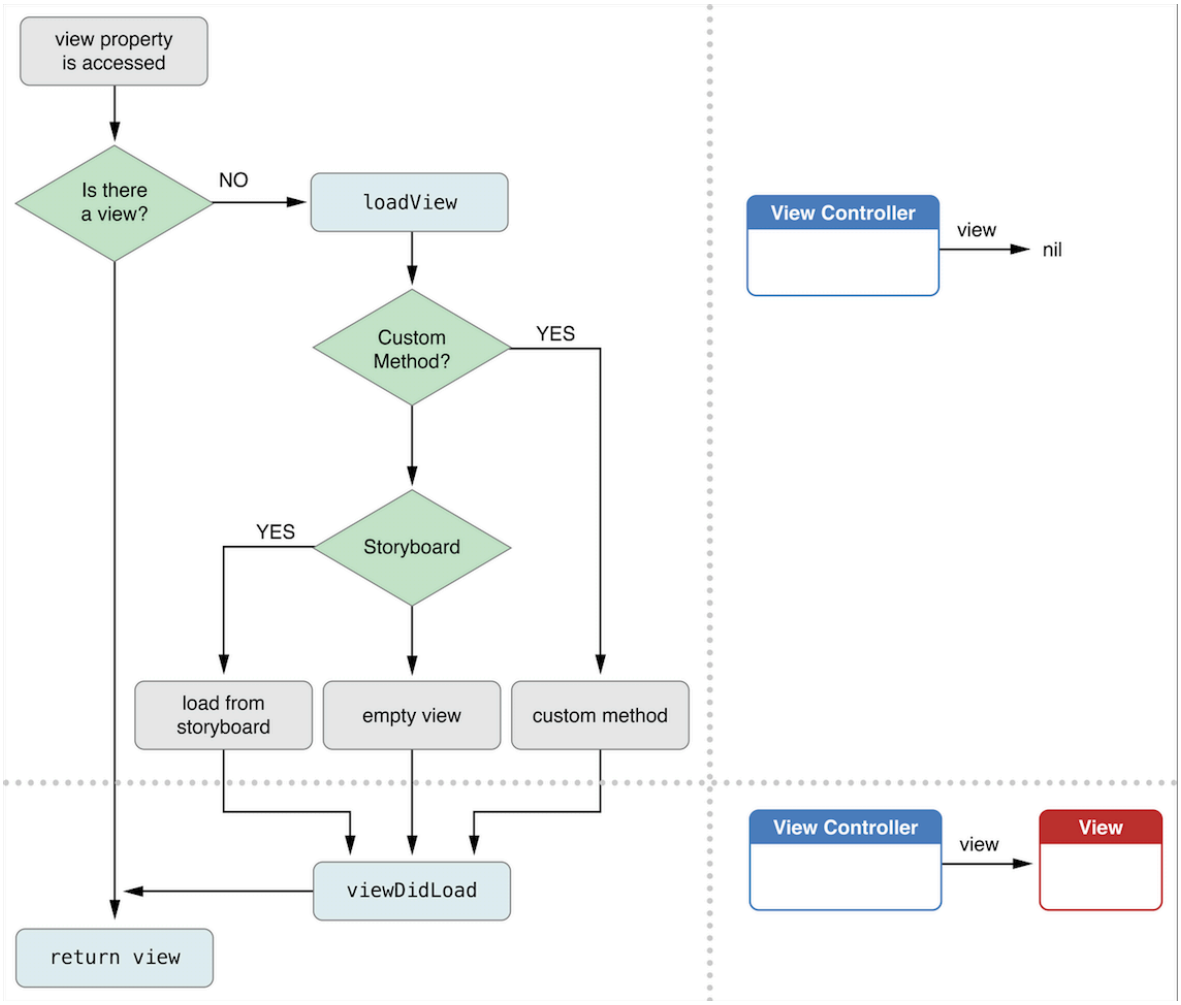
A View Controller Instantiates Its View Hierarchy When Its View is Accessed

1. Whenever some part of your app asks the view controller for its view object, and that object is not currently in memory, the view controller loads the view hierarchy and stores it in its view property for future reference. The steps that occurs during the load cycle are:

=> The view controller call its loadView method.

=> The view controller call its viewDidLoad method, which enables your subclass to perform any additional load-time tasks.

2. Loading a view into memory



> Loading a View Controller's View from a Storyboard

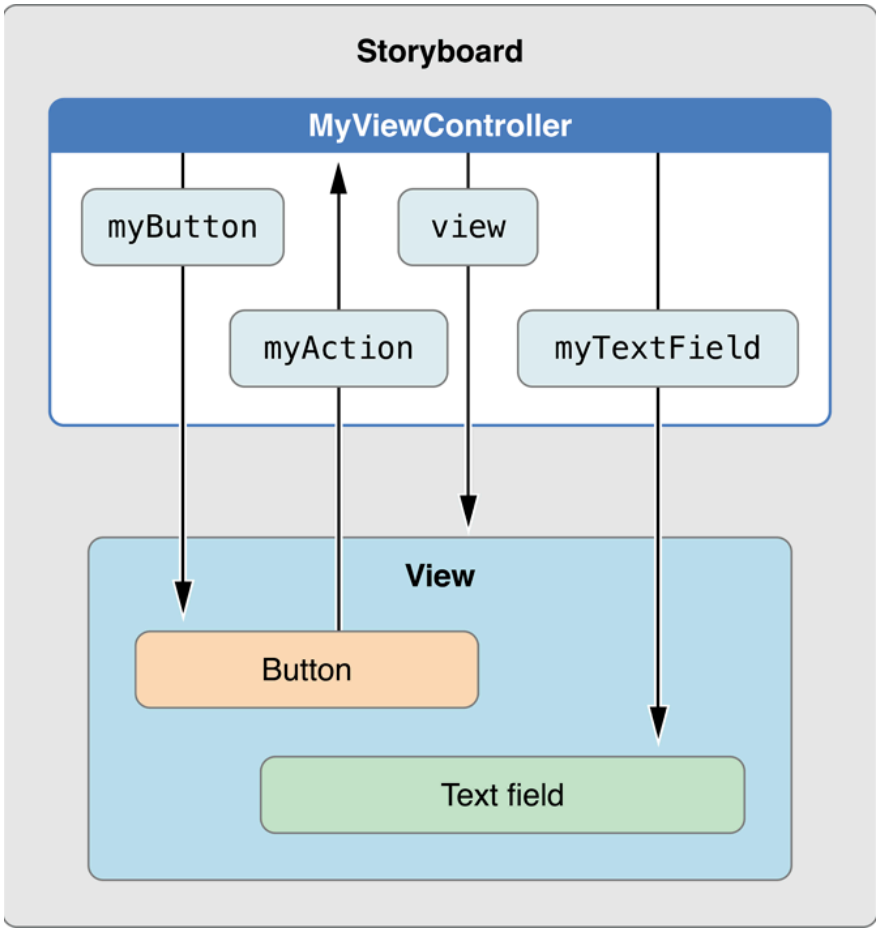
>> Creating the View in Interface Builder
Configuring the View Display Attributes in Interface Builder
Configuring Actions and Outlets for Your View Controller

1. Custom view controller class declaration

```
@interface MyViewController()
@property (nonatomic) IBOutlet id myButton;
@property (nonatomic) IBOutlet id myTextField;

- (IBAction)myAction:(id)sender;
@end
```

2. Connections in the storyboard



> Creating a View Programmatically
1. If you prefer to create a view programmatically, you do so by overriding your view controller's loadView method:
=> Create a root view object.
=> Create additional subviews and add them to the root view.
For each view, you should
a. Create and initialize the view.
b. Add the view to a parent view using the addSubview: method.
EXAMPLE

```
- (void)loadView
{
    CGRect applicationFrame = [[UIScreen mainScreen]
applicationFrame];
    UIView *contentView = [[UIView alloc]
initWithFrame:applicationFrame];
    contentView.backgroundColor = [UIColor blackColor];
    self.view = contentView;

    levelView = [[LevelView alloc] initWithFrame:applicationFrame
viewController:self];
```

```
    [self.view addSubview:levelView];
}
```

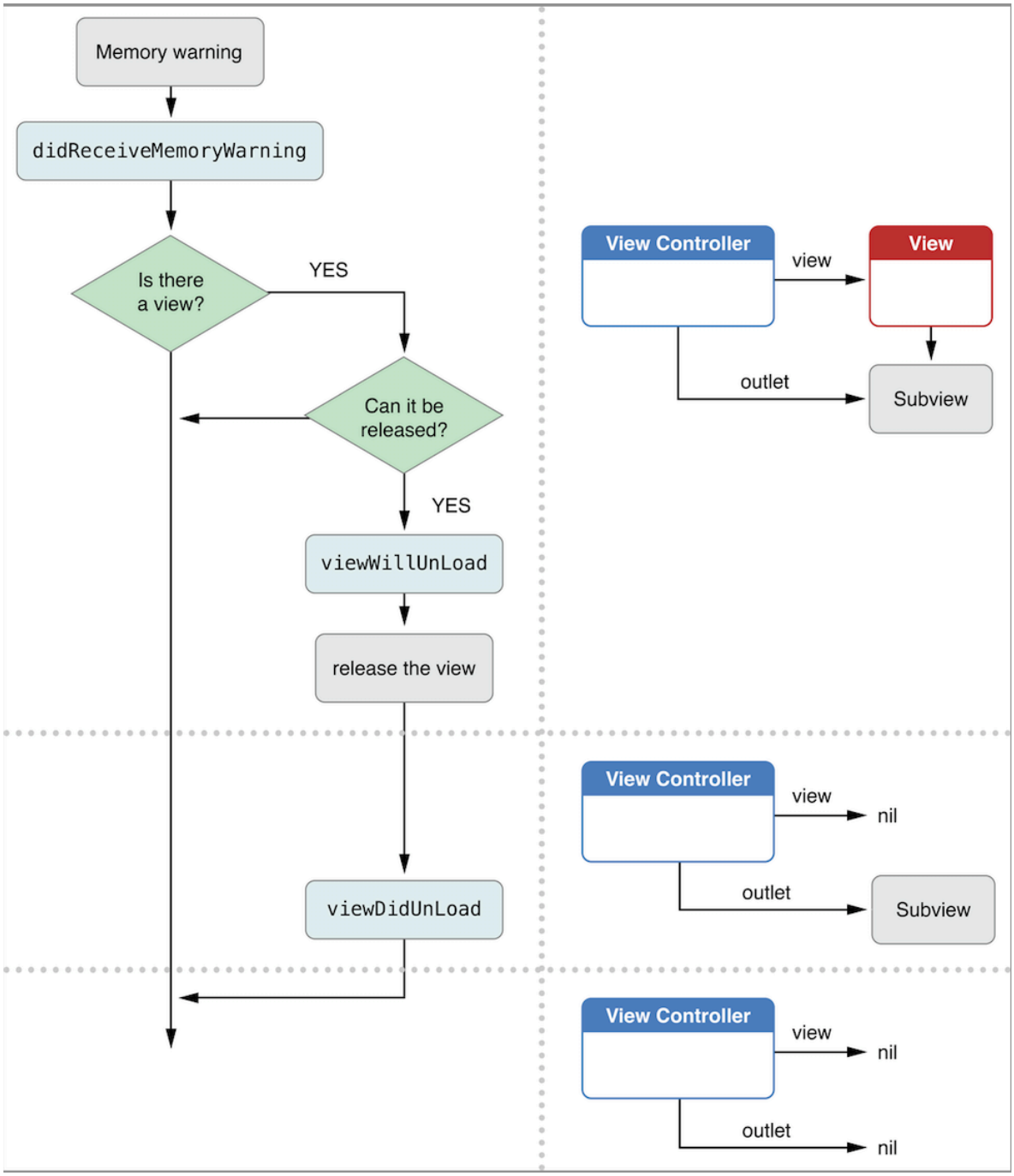
Managing Memory Efficiently
Places to allocate and deallocate memory

Task	Methods	Discussion
Allocating critical data structures required by your view controller	Initialization methods	Your custom initialization method (whether it is named `init` or something else) is always responsible for putting your view controller object in a known good state. This includes allocating whatever data structures are needed to ensure proper operation.
Creating your view objects	`loadView`	Overriding the `loadView` method is required only if you intend to create your views programmatically. If you are using storyboards, the views are loaded automatically from the storyboard file.
Creating custom objects	Custom properties and methods	Although you are free to use other designs, consider using a pattern similar the `loadView` method. Create a property that holds the object and a matched method to initialize it. When the property is read and its value is `nil`, call the associated load method.
Allocating or loading data to be displayed in your view	`viewDidLoad`	Data objects are typically provided by configuring your view controller's properties. Any additional data objects your view controller wants to create should be done by overriding the `viewDidLoad` method. By the time this method is called, your view objects are guaranteed to exist and to be in a known good state.

Responding to low-memory notifications	`didReceiveMemoryWarning`	Use this method to deallocate all noncritical objects associated with your view controller. On iOS 6, you can also use this method to release references to view objects.
Releasing critical data structures required by your view controller	`dealloc`	Override this method only to perform any last-minute cleanup of your view controller class. Objects stored in instance variables and properties are automatically released; you do not need to release them explicitly.

> On iOS 6 and Later, a View Controller Unloads Its Own Views When Desired

> On iOS 5 and Earlier, the System May Unload Views When Memory Is Low
Unloading a view from memory


Memory warning
didReceiveMemoryWarning
Is there a view?
YES
Can it be released?
YES
viewWillUnLoad
release the view
viewDidUnLoad
View Controller
view
View
outlet
Subview
View Controller
view
nil
outlet
Subview
View Controller
view
nil
outlet
nil