Objective-C

Курс лекций и семинаров для студентов,
желающих научиться программировать под iPhone

Осень-Зима 2013

Лекция №6

UIViewConotroller: Life cycle, View management, Containment

Автор: Дмитрий Волков, iPhone Developer, Sibers

Sibers®

Что вы узнаете сегодня:

1. Responsibilities
2. App Hierarchy
3. UIView life cycle
4. Containment
5. Modal Controllers
6. INavigationController

UIViewController

UIViewController — основная боевая единица вашего приложения.

- Тот самый Controller из MVC. Обычно управляет передачей данных из model во view и наоборот.

- В iOS разработке обычно UIViewController = один экран приложения (UI + логика)

UIViewController

UIViewController — базовый класс, описывающий интерфейс контроллера.

- Наследуется от ***UIResponder***, может принимать участие в Responder chain.
- **UIKit** имеет довольно много готовых системных контроллеров для различных системных служб (фото-галерея, список контактов, музыка библиотека и т.д.)
- Так же, присутствуют контроллеры-контейнеры, используемые для иерархического (UINavigationController) или разделенного (UITabbarController) предоставления своих дочерних контроллеров.

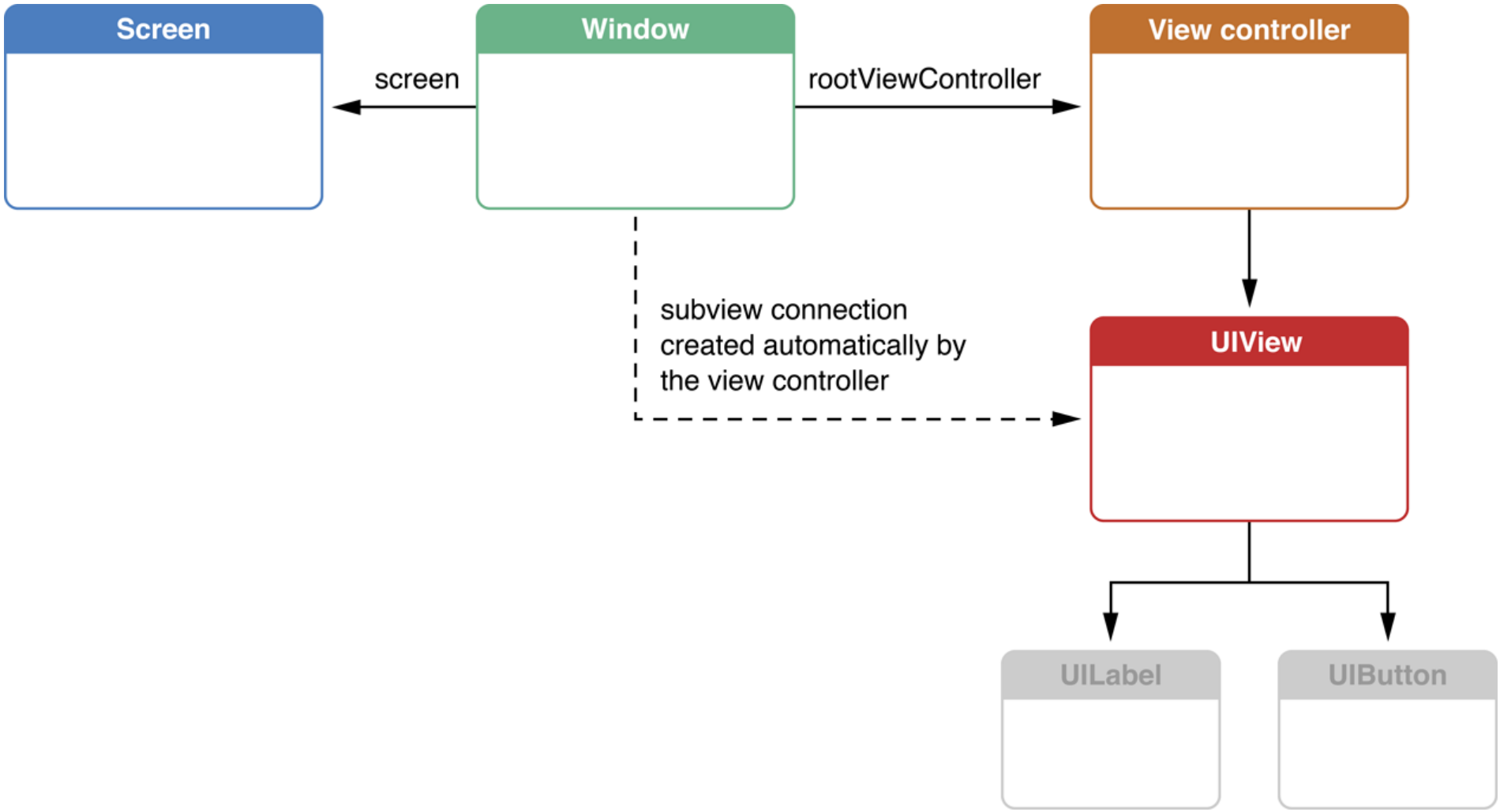
Responsibilities

- Контроллер управляет частью вашего UI, предоставляет UIView, который он контролирует.
- Отвечает за изменении ориентации UI при повороте экрана устройства.
- Участвует в Responder chain, если его UIView никак не обработали события.
- Реагирует на сообщения системы о недостаточном объеме памяти.
- Восстанавливает состояние UI после перезапуска приложения.

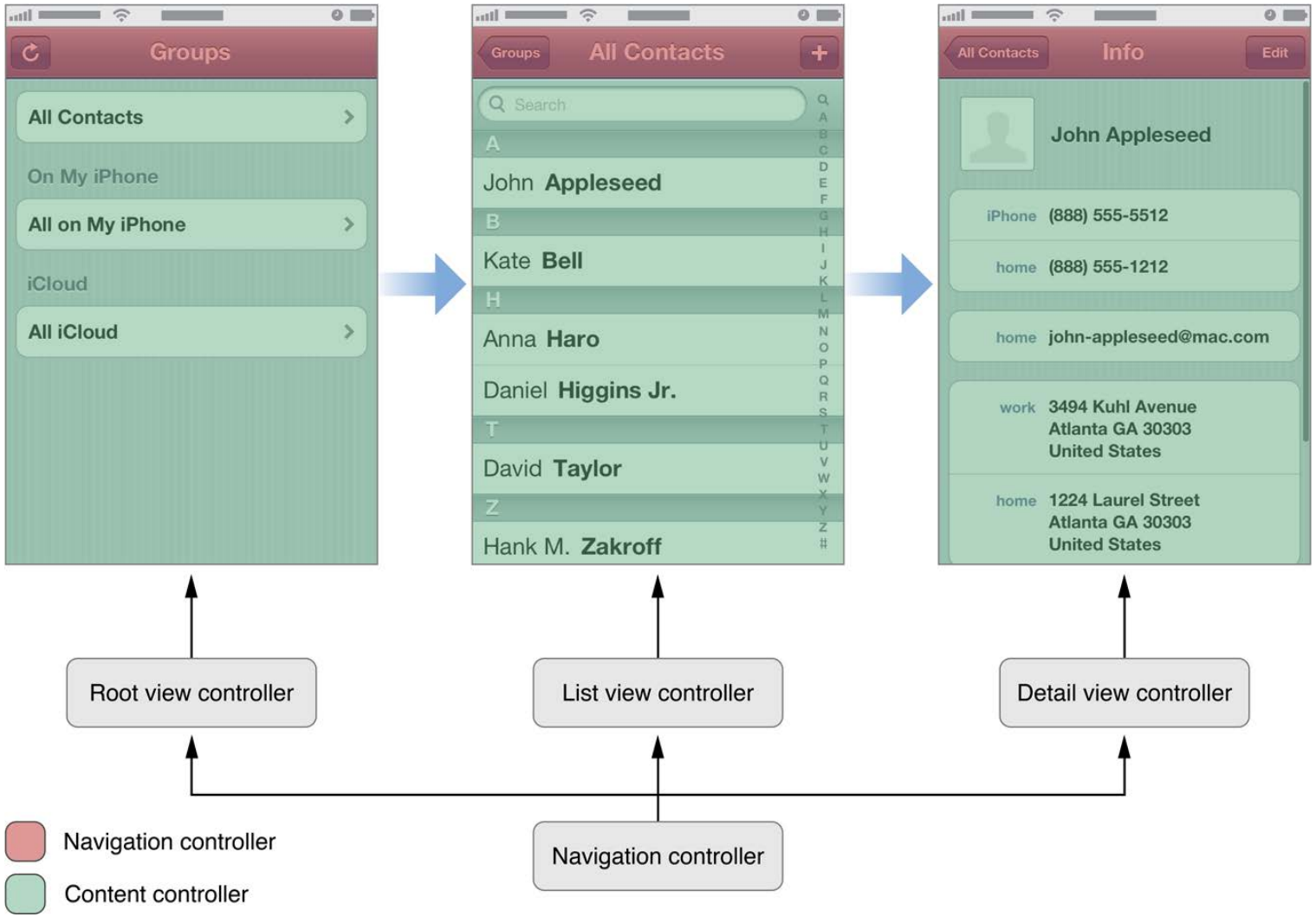
Responsibilities

- Контроллер отвечает на события UI: нажатие кнопок, изменение текстовых полей.
- Контроллер решает, какие именно действия производить в ответ на взаимодействие пользователя с UI — обновить модель, сыграть анимацию, выполнить запрос по сети и т. д.
- Ввиду того, что на мобильных устройствах размер экрана ограничен, UI разбивается на отдельные логические части-контроллеры.
- Каждый контроллер может так же показывать другие контроллеры, предоставляющие отдельную логику.

App hierarchy



Controllers hierarchy



UIView life cycle

- Базовый класс UIViewController предоставляет методы, оповещающие о текущем состоянии UIView, которым управляет контроллер.
- При переопределении большинства этих методов следует вызвать метод родительского класса.
- Методы оповещают о том, загрузился ли UIView, видим ли он на экране, попытках поворота устройства.

UIView life cycle

Базовый класс ***UIViewController*** объявляет свойство

@property(nonatomic, retain) UIView* view;

Это тот самый объект класса UIView (либо наследник этого класса), которым управляет контроллер.

ВАЖНО: view не создается до тех пор, пока не будет каким-либо образом добавлен на экран, либо вы лично не обратитесь к этому свойству: ***viewcontroller.view***

Для того, чтобы проверить, загружен ли view, но при этом не создавать его, UIViewController объявляет метод:

- (BOOL) isViewLoaded

UIView life cycle

Для создания контроллеров используется инициализатор:

- (id) initWithNibName:(NSString*) nibName nibBundleName:(NSString*) nibBundle

Здесь вы можете указать, какой именно .xib файл и из какого бандла следует загрузить данному контроллеру.

Если вы явно не укажете имя, контроллер попытается найти файл ***<ControllerClass>.xib*** и загрузить его.

Если такой файл не найден, view контроллера будет обычным пустым UIView.

Метод ***init*** класса **UIViewController** вызывает initWithNibName:nibBundle с пустыми параметрами.

UIView life cycle

Если вы создавали контроллер без .xib файла:

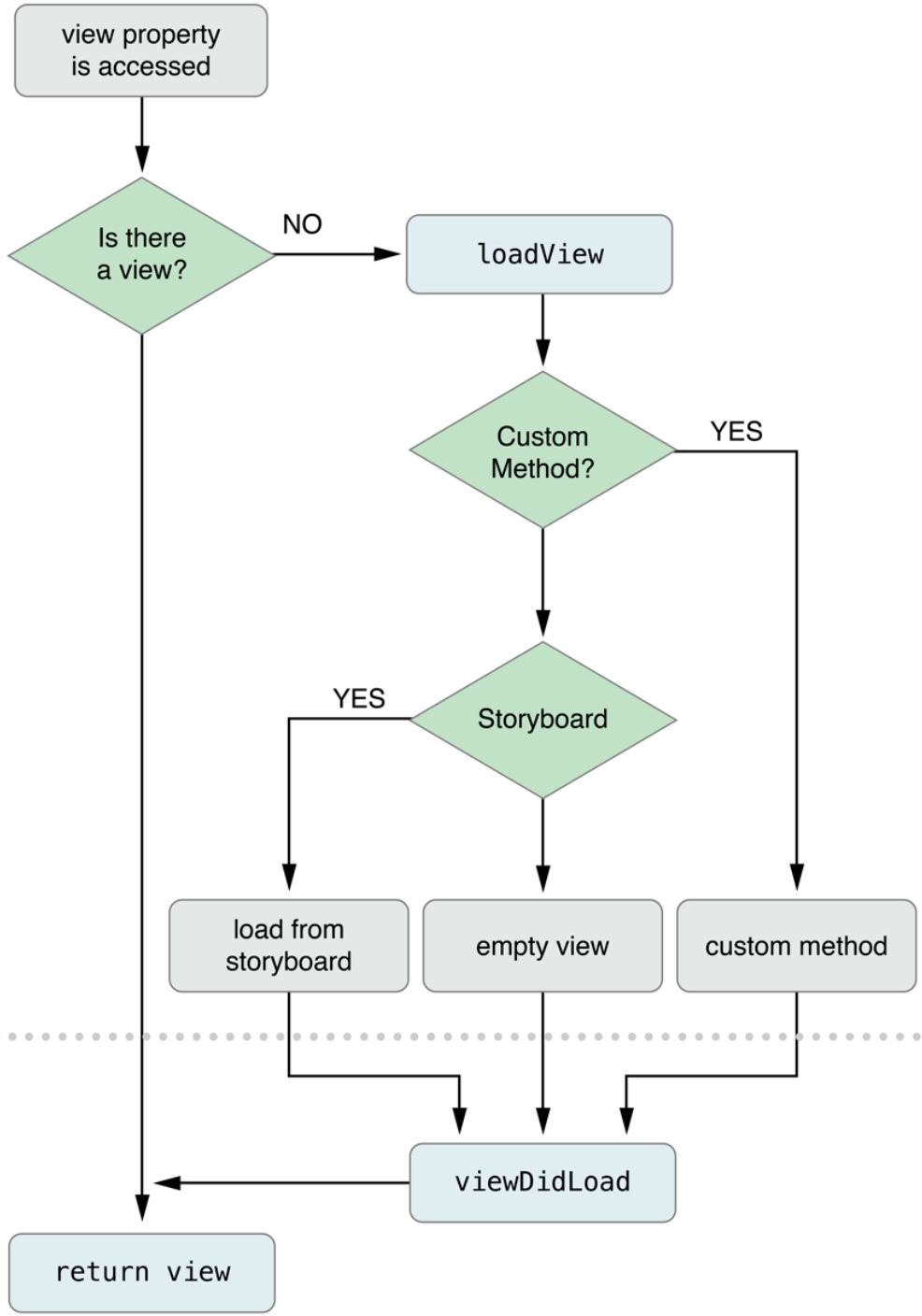
Переопределяете метод – ***(void) loadView*** и в нем создаете иерархию своих UI-компонентов.

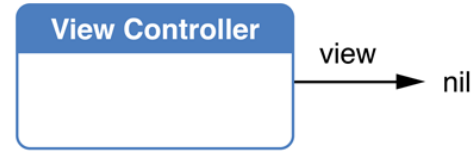
После того, как view будет создан, вызывается метод:

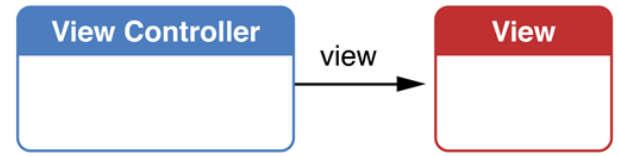
- (void) viewDidLoad — в нем можно дополнительно настроить свой UIView.

ВАЖНО: методы ***init / initWithNibName:nibBundle:*** НЕ СОЗДАЮТ view.

В версиях iOS до 6.0 система могла выгрузить view контроллера при недостатке памяти вызывался метод – ***(void) viewDidUnload***


view property is accessed
Is there a view?
NO
loadView
Custom Method?
YES
Storyboard
YES
load from storyboard
empty view
custom method
viewDidLoad
return view


View Controller
view
nil


View Controller
view
View

Sibers®

UIView life cycle

```
AwesomeController* controller = [[AwesomeController alloc]
initWithNibName:@”AwesomeController” nibBundle:nil];

//или короче

AwesomeController* controller = [[AwesomeController alloc] init];

//или еще короче

AwesomeController* controller = [AwesomeController new];
```

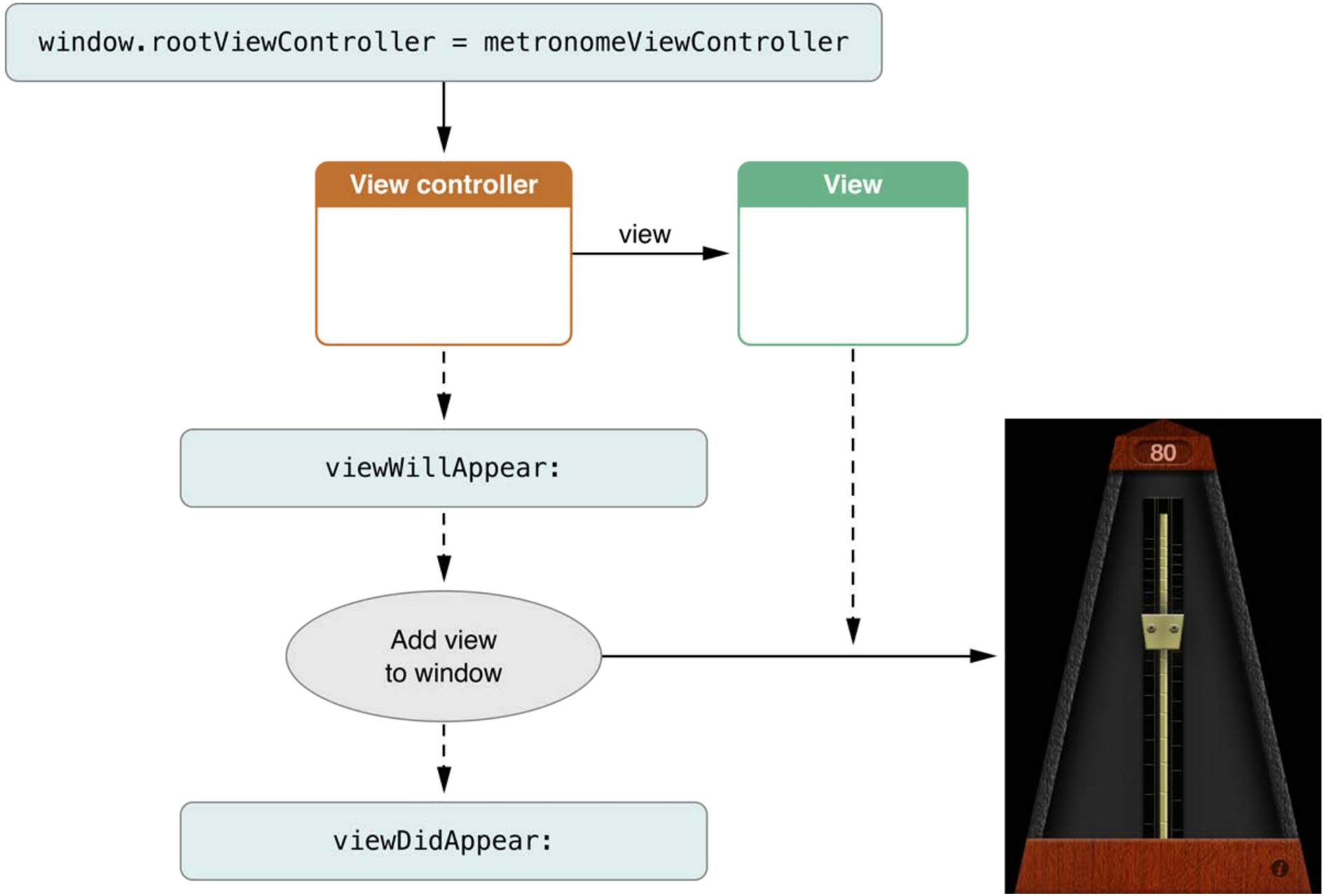
UIView life cycle

```
- (BOOL)application:(UIApplication*) application
didFinishLaunchingWithOptions:(NSDictionary*) launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen]
bounds]];
    MainViewController *mainViewController = [MainViewController new];
    self.window.rootViewController = mainViewController;
     [self.window makeKeyAndVisible];
    return YES;
}
```

UIView visibility cycle

UIViewController определяет методы, извещающие о текущем состоянии UIView:

- ***(void) viewWillAppear / viewDidAppear*** — вызываются перед / после появлением view на экране.
- ***(void) viewWillDisappear / viewDidDisappear*** — вызываются перед / после исчезновения view на экране
- ***(void) viewWillLayoutSubviews / viewDidLayoutSubviews*** — вызывается перед / после того, как view изменил свой frame / bounds и разложил по местам свои subview.


window.rootViewController = metronomeViewController
View controller
view
View
viewWillAppear:
Add view
to window
viewDidAppear:
80

UIView orientation cycle

UIViewController определяет следующие методы, вызываемые при изменении ориентации экрана (6.0+)

- (BOOL) shouldAutorotate — вызывается при изменении ориентации и определяет, поддерживает ли поворот интерфейса контроллер.

- (UIInterfaceOrientationMask) supportedInterfaceOrientations — вызывается для определения, в каких ориентациях может находиться ваш UI

Методы, оповещающие о прогрессе изменения ориентации:

-(void) willRotateToInterfaceOrientation:(UIInterfaceOrientation) toInterfaceOrientation duration:(NSTimeInterval)duration

- (void) didRotateFromInterfaceOrientation:(UIInterfaceOrientation) fromInterfaceOrientation

willRotateToInterfaceOrientation:duration:

rotate the view

viewWillLayoutSubviews

willAnimateRotationToInterfaceOrientation:duration:

rotation is complete

didRotateFromInterfaceOrientation:

Containment

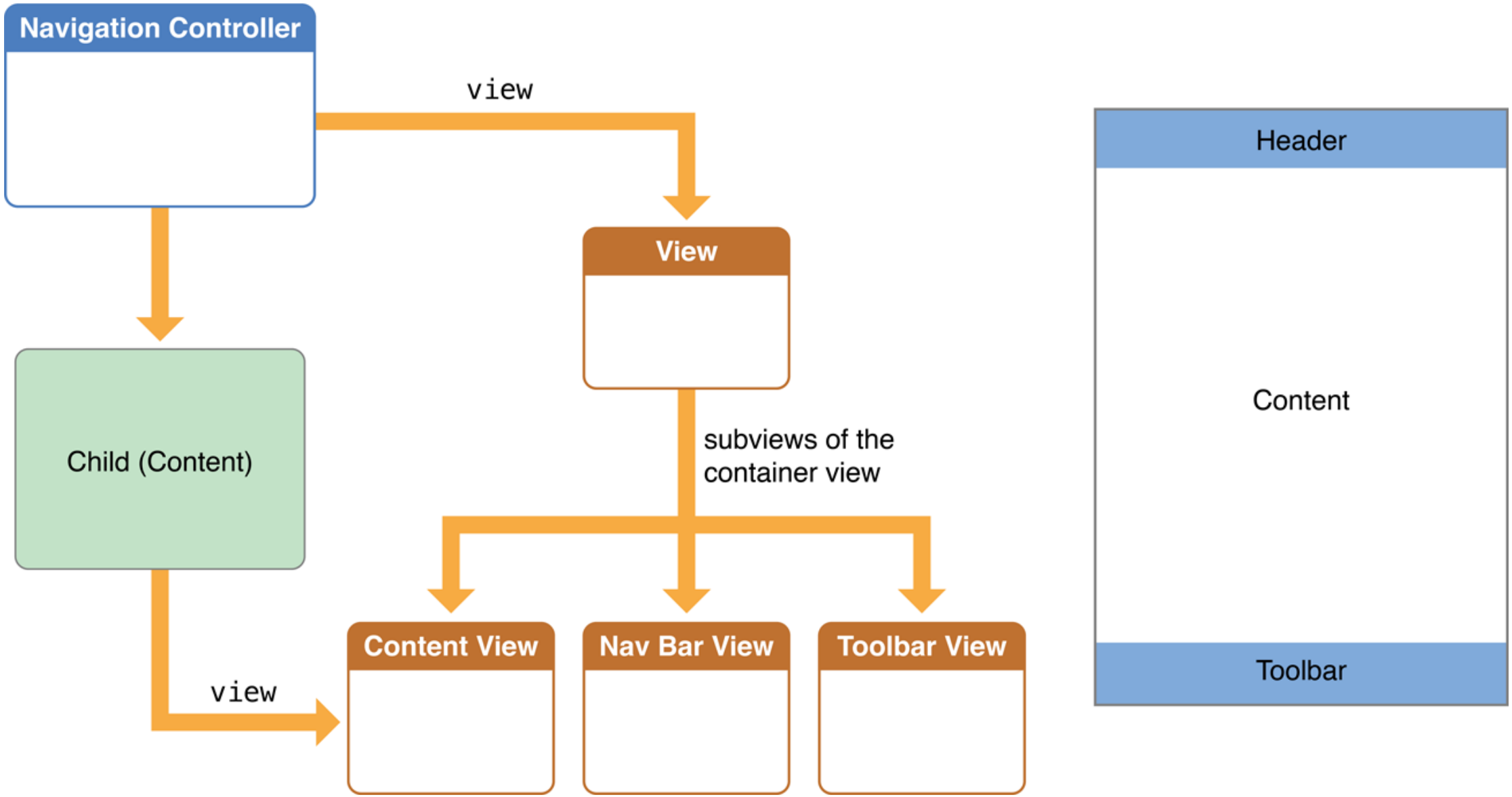
- Начиная с iOS 5 введена возможность построения иерархий контроллеров
- Вы можете самостоятельно выстраивать логику поведения и отображения вашим родительским контроллеров дочерних контроллеров в зависимости от действий пользователя.

Используются следующие методы:

- ***addChildViewController:(UIViewController*) controller***
- ***removeFromParentViewController***
- ***willMoveToParentViewController:(UIViewController*) controller***
- ***didMoveToParentViewController:(UIViewController*) controller***

Containment

```
- (void) displayContentController: (UIViewController*) content;
{
   [self addChildViewController:content];
   content.view.frame = [self frameForContentController];
   [self.view addSubview:self.currentClientView];
   [content didMoveToParentViewController:self];
}

- (void) hideContentController: (UIViewController*) content
{
   [content willMoveToParentViewController:nil];
   [content.view removeFromSuperview];
   [content removeFromParentViewController];
}
```


Navigation Controller
view
View
Child (Content)
subviews of the
container view
view
Content View
Nav Bar View
Toolbar View
Header
Content
Toolbar

Containment

Передачу сообщений о том, видим ли контроллер на экране сейчас, а также передача сообщений о смене ориентации контролируется следующими методами:

- ***(BOOL)shouldAutomaticallyForwardRotationMethods***
- ***(BOOL)shouldAutomaticallyForwardAppearanceMethods***

- Реализация этих методов в UIViewController возвращает YES
- Если вы возвращаете NO из этих методов, вы сами должны вызывать соответствующие методы у дочерних контроллеров.

- (void)beginAppearanceTransition:(BOOL)isAppearing animated:(BOOL)animated

- (void)endAppearanceTransition

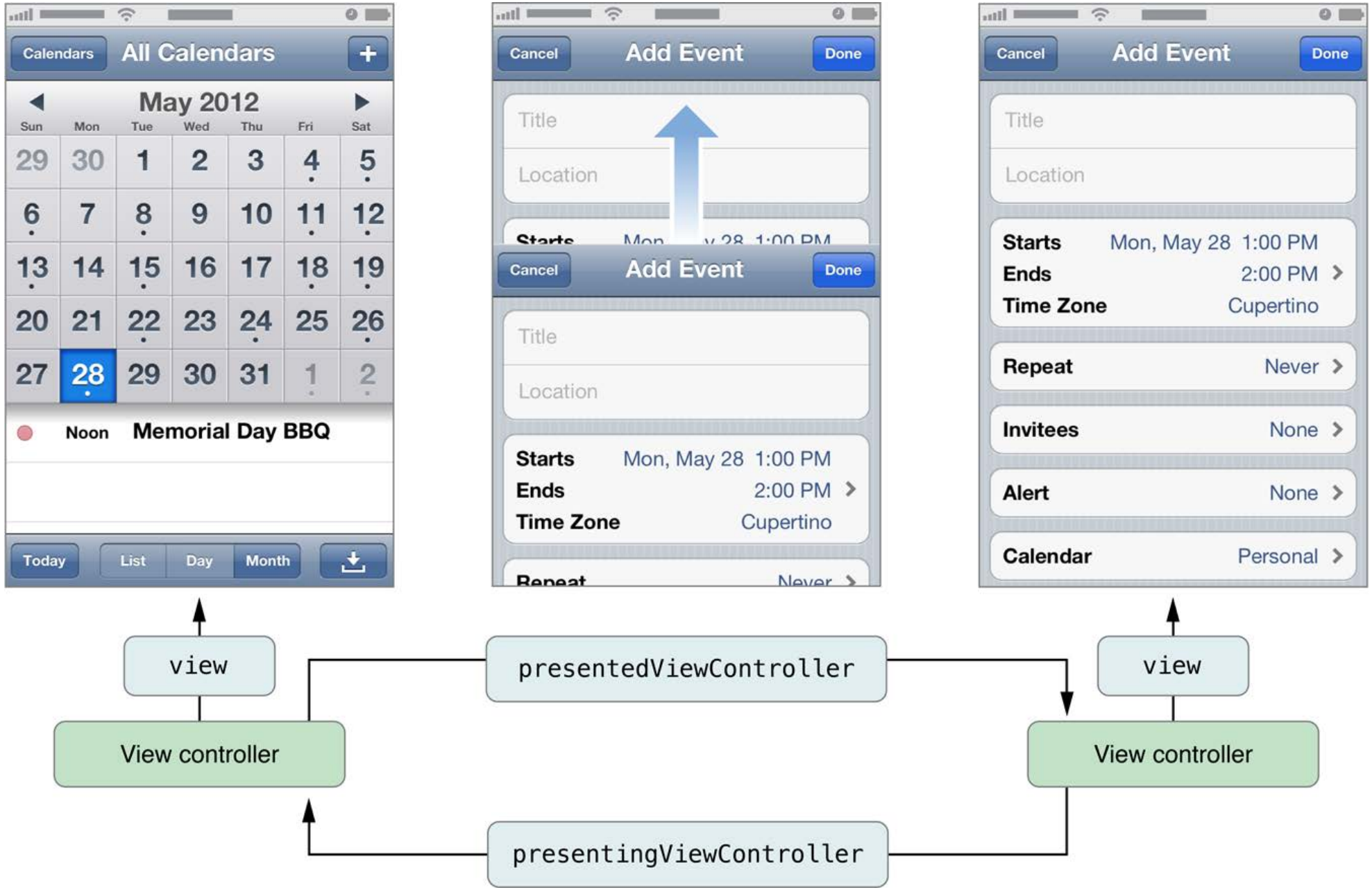
Modal Controllers

- Для получения важных данных от пользователя (например, логин и пароль) иногда требуется прервать вашу цепь контроллеров и показать особый «модальный» контроллер.
- Модальные контроллеры должны предоставлять небольшой набор данных, которые должен ввести пользователь. Они не должны быть продолжением логики вашего приложения.

- Модальные контроллеры отображаются при помощи следующих методов:

  - (void) presentViewController:(UIViewController*) controller animated:(BOOL) animated

  - (void) dismissViewControllerAnimated:(UIViewController*) controller


Calendars
All Calendars
+
May 2012
Sun Mon Tue Wed Thu Fri Sat
29 30 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31 1 2
Noon Memorial Day BBQ
Today
List
Day
Month
Cancel
Add Event
Done
Title
Location
Starts Mon, May 28 1:00 PM
Ends 2:00 PM
Time Zone Cupertino
Repeat Never
Invitees None
Alert None
Calendar Personal
view
presentedViewController
View controller
presentingViewController

Modal Controllers

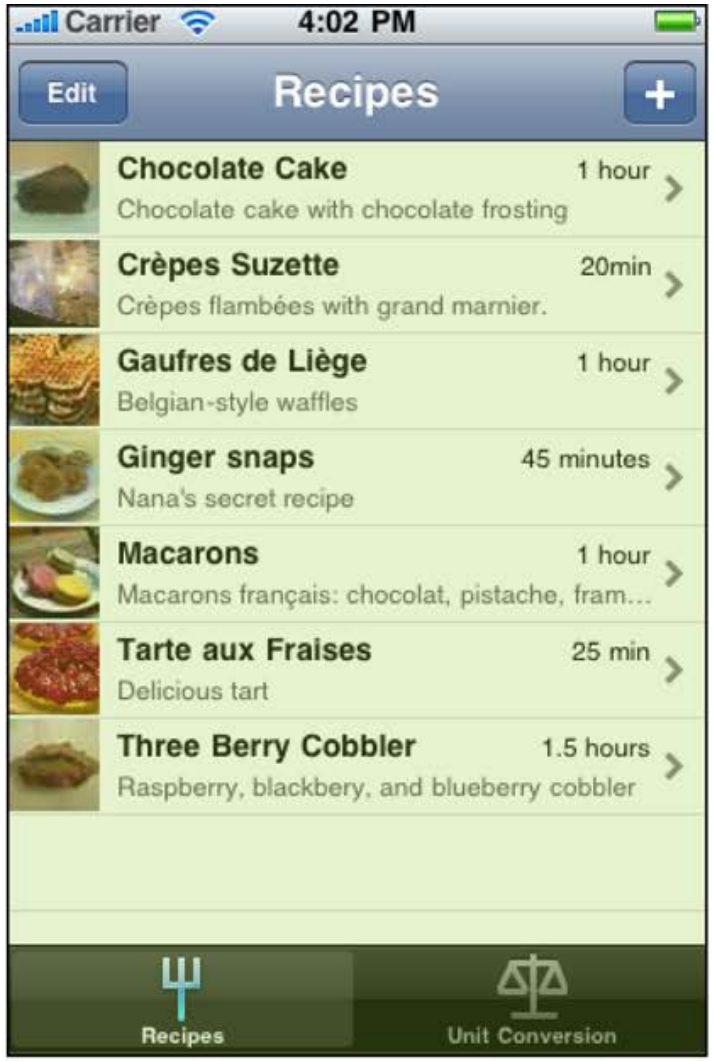
```
- (void)add:(id)sender
 {
      RecipeAddViewController *addController = [[RecipeAddViewController
alloc]
                    init];

      [self presentViewController: addController animated:YES completion:
nil];
}
```

INavigationController

UINavigationController — системный контроллер, управляющий стэком контроллеров, используется для предоставления иерархической информации.

Кроме отображения контента контроллеров, которые он содержит, **UINavigationController** так же добавляет в иерархию свои views:

- ***UINavigationBar*** — отображает кнопку «назад», и, опционально, дополнительные UI-компоненты и название экрана.
- ***UIToolbar*** — опциональный набор дополнительных UI.



List controller

Detail controller

Photo controller

UINavigationController

Для переходов между контроллерами и отображения требуемого в данных момент контента используются методы:

- ***- (void) pushViewController:(UIViewController*) controller animated:(BOOL) animated*** — добавляет контроллер на вершину стэка, делая его видимым
- ***- (void) popViewControllerAnimated:(BOOL) animated*** — убирает контроллер с вершины стэка, делая видимым предыдущий контроллер
- ***- (void) setViewControllers:(NSArray*) controllers*** — очищает стэк контроллеров и заменяет его контроллерами из массива.

Кнопка «Back» добавляется автоматически и при нажатии вызывает popViewControllerAnimated:

UINavigationController

```
- (void)applicationDidFinishLaunching:(UIApplication *)application
{
    UIViewController *myViewController = [[MyViewController alloc] init];
    navigationController = [[UINavigationController alloc]
                            initWithRootViewController:myViewController];

    window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    window.rootViewController = navigationController;
    [window makeKeyAndVisible];
}
```

UINavigationController

```
- (void) showRecipe:(Recipe*) recipe
{
    RecipeViewController* vc = [[RecipeViewController alloc]
initWithRecipe:recipe];
    [self.navigationController pushViewController:vc animated:YES];
}

- (void) hideRecipeController
{
    [self.navigationController popViewControllerAnimated:YES];
}
```

Заключение

- UIViewController — базовая логическая единица iOS приложений.
- UIViewController управляет своим UIView, а также извещается о различных системных событиях.
- iOS позволяет создавать свои контроллеры-контейнеры.
- Система предоставляет удобные стандартные контроллеры для отображдения пользовательского контента (UINavigationController, UITabbarController)

Спасибо за внимание!

Жду ваших вопросов.