Жизненный цикл UIViewController

Согласно шаблону проектирования MVC `UIViewController` в этой схеме является *Controller'ом*. Т.е. он обеспечивает взаимосвязь между моделью и *View*. В iOS на *ViewContrller* возлагаются задачи, связанные с контролем жизненного цикла *View*, и его отображением. *View* довольно тяжеловесный объект и использует много памяти, объем которой в мобильном устройстве ограничен, поэтому он грузится только тогда когда действительно нужен, т.е. непосредственно перед отображением, а не при создании *ViewController'a*, и выгружается из памяти сразу после того, как перестает быть нужным.

В момент когда *View* загружается происходит выполнение определенных методов. Далее после загрузки *View* на всем протяжении его жизненного цикла, вплоть до его закрытия в роль вступают те или иные методы, которые обеспечивают возможность управлять и взаимодействовать с *View*.

viewDidLoad

В данном методе можно выполнить большинство настроек для контроллера. Он вызывается ровно один раз, когда контроллер представления загружается в память. Здесь можно инициализировать переменные и объекты, которые должны жить на протяжении всего жизненного цикла вью.

На этом этапе границы вью еще не установлены и мы не можем обращаться к значениям геометрии (высоте, ширине и прочим параметрам)

viewWillApear

Перед появлением вью на экране мы получаем уведомление об этом от метода `viewWillApear`. Этот метод вызывается перед тем как *view* будет добавлена в текущую иерархию, перед отработкой какой бы то ни было анимации. Для чего можно

использовать `viewWillApear`? Для более тонкой настройки subview, которые нужно сделать без анимации. Если обратиться к официальной документации, то там сказано, что в этом методе можно менять стиль статус бара, или стиль всего View. Данный метод может вызваться не один раз в течении жизненного цикла контроллера. Обязательно нужно вызывать `super.viewWillAppear()`

Сразу после этого устанавливаются границы вью и геометрические значения становятся доступны. Расположение всех элементов пользовательского интерфейса задаются параметрами *Autolayout* , а методы `viewWillLayoutSubviews` и `viewDidLayoutSubviews` вызываются чтобы убедится, что расстановка выполнена правильно. Эти два метода вызываются до начала расстановки и сразу же после расстановки.

viewWillLayoutSubviews

Если вы не используете *Auto Layout*, то `viewWillLayoutSubviews` является тем самым местом, где вы можете изменить размеры subview и их положение. Вызов `super.viewWillLayoutSubviews` не требуется.

updateViewConstraints

В остальных случаях, для того, что бы изменить значения констрейнтов (например, при повороте экрана), используется метод `updateViewConstraints`. В конце переопределенного метода нужно обязательно вызывать `super.updateViewConstraints` и именно в конце. Вызов данного метода происходит перед вызовом `viewWillLayoutSubviews`.

viewDidLayoutSubviews

На этом этапе жизненного цикла можно быть уверенным, что *View* корректно выставила положения для всех своих потомков. В этом методе можно сохранять последние состояния каких либо элементов, например положение *Scroll View* или выделение ячейки в *Table View*. По умолчанию данный метод не делает ни чего, так что вызывать *super.viewDidLayoutSubviews()* не нужно

viewDidApear

После расстановки границ, вью появляется на экране и мы получаем уведомление об этом от метода `viewDidApear`. Данный метод является финальным метод в цепочке и может быть вызван не один раз в течении жизненного цикла вью контроллера, например если из контроллера открыть *Modal View Controller*. В этом случае наш контроллер не будет выгружен из памяти, но само вью, тем не менее будет закрыто. После возврата на наш вью контроллер из *Modal View Controller* вью снова появится на экране, вызвав метод `viewWillApear` и `viewDidApear`. Данный метод не лучшее место для хранения, каких либо свойств и объектов, т.к. это вызовет утечку памяти, если вы не отпустите их, когда представление исчезнет. В этом методе можно размещать ресурсоемкие операции, например, связанные с получение данных, оповестив об этом пользователя. В нем можно вызвать `becomeFirstResponder`, чтобы сразу начать ввод в нужное поле.

Требует вызова `super.viewDidAppear()`

viewWillTransitionToSize

Обработка поворота экрана обрабатывается автоматически, и задается в настройках приложения. Однако если нужно самостоятельно обработать анимацию при повороте, то можно можно воспользоваться методом `viewWillTransitionToSize`

didReceiveMemoryWarning

В случае нехватки памяти срабатывает метод `didReceiveMemoryWarning` и в нем можно попробовать обнулить объекты, которые не используются.

viewWillDisapear

Когда вью уходит с экрана, то перед этим срабатывает метод `viewWillDisapear`. Здесь обычно коммитят изменения, убирают `first responder` статус, ставят на паузу выполняемые действия, отменяют ориентацию/стиль статус бара если ее меняли в

`viewWillAppear` и т.д. Можно почистить данные, обнулить кеш и все в таком духе. Метод требует вызов `super.viewWillDisappear`

viewDidDisapear

Оповещение о том, что *View* было удалено из иерархии. Область применения схожа с `viewWillDisappear`, здесь так же удаляют ненужные данные, ставят на паузу плеер. Нужно просто понимать, что этот метод вызовется после анимации удаления *View*, а предыдущий перед. Также нужно вызывать `super.viewDidDisappear`

awakeFromNib

Отдельно нужно сказать про метод — `awakeFromNib`. На самом деле он не является частью *ViewController Lifecycle*, но нужно понимать что он вызывается сразу после инициализации, перед подготовкой перехода у контроллера, который задан с помощью *storyboard*.

loadView

Так же стоит упомянуть по метод `loadView`, который используется при ручной инициализации View. Если не использовать данный метод, то контроллер делает это автоматически.

Жизненный цикл приложения

Итак сейчас мы рассмотрели основные методы жизненного цикла вью контроллера. Теперь давайте рассмотрим методы, которые срабатывают на определенной стадии жизненного цикла самого приложения.

Любое приложение имеет два режима работы: Фоновый режим и Режим работы на переднем плане. При этом каждое приложение может иметь несколько состояний:

- Не запущено
- Не активно
- Активно
- Фоновый режим работы
- Остановлено

Переход из одного состояния в другое сопровождается вызовом определенных методов класса `AppDelegate`.

didFinishLaunchingWithOptions

Это самый первый метод, который срабатывает после загрузки приложения и в нем делают первичную настройку параметров самого приложения:

- Определяется интерфейс в зависимости от типа устройства на котором запущено приложение
- Определяется стартовый вью контроллер
- Инициализация пуш уведомлений
- Загрузка первичных данных из базы

Так же тут можно изменять глобальный интерфейс всего приложения. Например, можно изменить цвет навигейшин бара и его заголовка не для отдельно взятого экрана, а во всем приложении целиком.

applicationWillResignActive

Вызов данного метода происходит перед переходом приложения в фоновый режим. В фоновый режим приложение переходит при сворачивании, при входящем звонке, при переходе на другое приложение.

- Останавливаем любимые активные задачи
- Ставим на паузу игры и проигрывание видео

applicationDidEnterBackground

Вызов данного метода происходит после перехода приложения в фоновый режим. Метод используется для освобождения общих ресурсов, аннулирования таймеров и сохранения сведений о состоянии приложения для последующего его восстановления. Тут следует отключить обновления пользовательского интерфейса, а так же необходимо избегать использования OpenGL.

Для приложений, способных поддерживать работу в фоновом режими, этот метод вызывается вместо метода `applicationWillTerminate`, который в свою очередь вызывается при завершении работы приложения.

applicationWillEnterForeground

Вызов данного метода происходит при возврате приложения из фонового режима. Используется для отмены изменений, выполненных при входе в фоновый режим.

applicationDidBecomeActive

Перемещает приложение из неактивного состояния в активное. Переход происходит при запуске приложения, а так же при возврате приложения из фонового режима. Приложения также могут вернуться в активное состояние, если пользователь игнорирует прерывание (например, входящий телефонный звонок или SMS-сообщение), которое временно отправило приложение в неактивное состояние.

Этот метод следует использовать для перезапуска всех задач, которые были приостановлены (или еще не запущены), пока приложение было неактивно. Если приложение ранее находилось в фоновом режиме, его также можно использовать для обновления пользовательского интерфейса приложения.

applicationWillTerminate

Вызывается перед тем, как приложение будет выгружено из памяти.

Этот метод следует использовать для очистки приложения и высвобождения общих ресурсов, сохранение пользовательских данных и аннулирование таймеров. Для приложений, поддерживающих фоновое выполнение, этот метод обычно не вызывается при выходе пользователя из приложения, поскольку в этом случае приложение просто перемещается в фоновый режим. Однако этот метод может быть вызван в ситуациях, когда приложение работает в фоновом режиме (не приостановлено), и система должна прекратить его по какой-либо причине.