다양한 View Controller 배우기

Joonhwan

2013 년 4 월 26 일

차 례

1 Navigation Controller(이하 NC)

NC 는 VC 들을 스택으로 관리하여 계층화된 컨텐츠를 Drill-Down 인터페이스(아이콘이나 텍스트를 클릭하여 마치 꿰뚫고 들어가듯 검색하는 동작)를 제공한다.

NC는

1. NC가 직접 관리하는 View
2. 사용자가 제공한 컨텐츠 VC 에 의해 관리되는 view

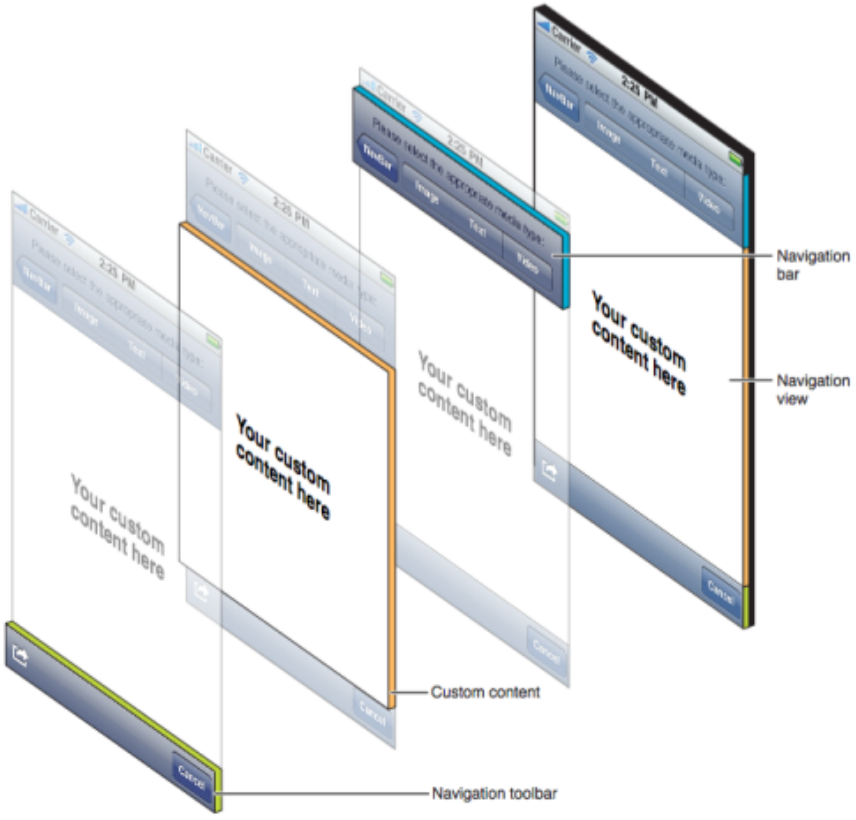
의 2가지로 구성되고, 각 컨텐츠 VC 는 독립적 view 계층을 관리하고, NC 는 이들간의 네비게이션을 관리한다.

1.1 네비게이션 인터페이스의 구성

NC는 컨텐츠 VC의 표시를 관리하는 주된 작업뿐 아니라, 자기 자신의 커스텀view 를 표시를 관할한다.

- 네비게이션 바
- 네비게이션 뷰
- 커스텀 컨텐츠
- 네비게이션 툴바

로 구성. 정말 잘그려진 한장의 그림.



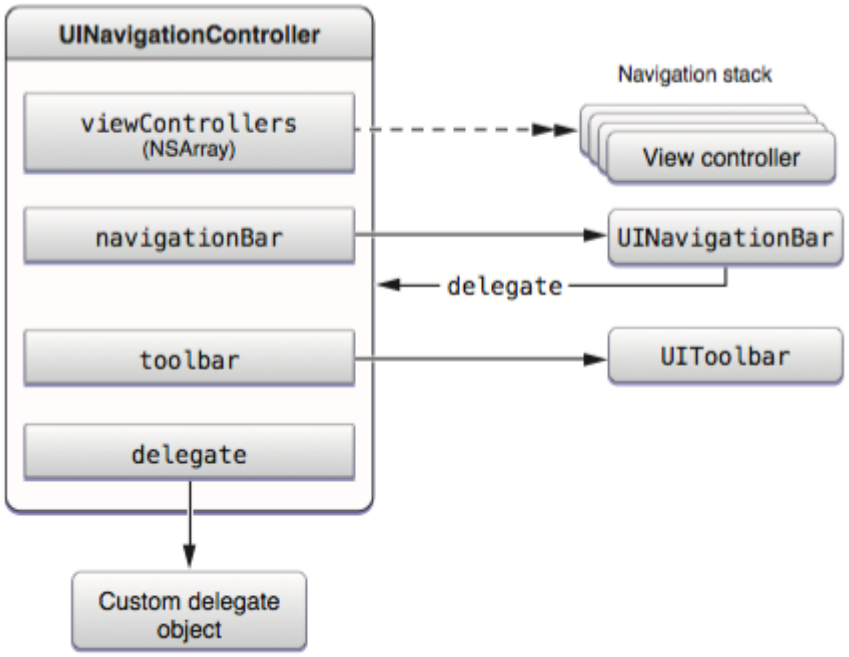
네비게이션 바와 툴바는 사용자화가 가능한 View이긴 하지만, 절대 네비게이션 계층내의 view들을 직접 수정하는 식으로 설정하면 안된다. `UINavigationController`

혹은 `UIViewController` 클래스의 메소드를 통해서만 사용자화 해야 한다. (구체적인 방법은 다음에 나옴)

1.2 네비게이션 인터페이스의 객체들

NC 는 몇개의 객체로 네비게이션 인터페이스를 구현. 이들중 일부는 사용자가 제공해야 하는 것이고 나머지는 NC 스스로 자동생성. 특히 컨텐츠 VC 는 사용자가 제공해야 한다. NC 로 부터 통지를 받기 위해서는 `delegate` 객체를 제공해야 함. NC 자체도 네비게이션 바 와 네비게이션 툴바 같은 view 를 생성하며, NC 에 의해 관리된다.

주요 객체와 NC 와의 관계를 한장의 그림.

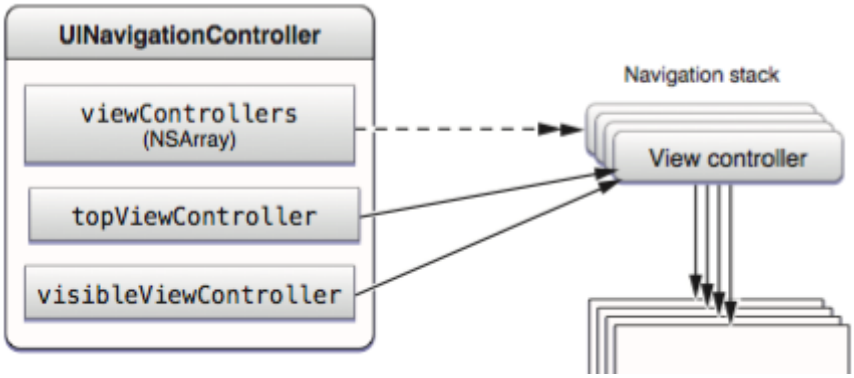


위 그림에서 사용자가 수정할 수 있는 부분은

- delegate
- navigation stack(이하 NS)

이다. stack 에 맨 처음 들어간 VC 는 root VC 가 되며, 절대 stack 에서 pop 되지 않는다. `UINavigationController` 의 메소드를 사용해 stack 에 추가적인 항목이 들어갈 수 있다.

다음 그림은 NC 와 NS 간의 관계를 나타낸 것.



위 그림에서 `topViewController` 와 `visibleViewController` 는 항상 같지 않을 수 있음에 유의(예를 들어, 어떤 VC 를 modal 하게 present 한 경우 `visibleViewController` 는 present 된 VC 를 나타내지만, `topViewController` 는 여전히 presenting 한 VC 를 가리키게 된단다).

NC 의 주요한 임무는 사용자의 행위에 따라, 새로운 컨텐츠 VC 를 스택에 push 혹은 pop 하는 것이다. 각 VC 는 앱의 데이터중 일정부분을 표시할 임무를 띄고 있다. 대개, 사용자가 현재 보이는 view 에서 어떤 항목을 선택한 경우 해당 항목에 대한 상세한 부분을 표시하는 VC 를 설정하고 NS 에 push 한다(예: 사진 어플에서 사용자가 사진 앨범을 선택하면 해당 사진 앨범을 표시하는 VC 를 NS 에 push).

대개의 경우, NS 에서 VC 를 코드상에서 pop 할 필요는 없다. 대신, NC 가 제공하는 네비게이션바의 "back(뒤로가기)" 버튼이 눌릴 때, stack 의 가장 위에 놓인 VC 가 자동적으로 pop 된다.

1.3 네비게이션 인터페이스(이하 NI) 생성하기

NI 를 생성할 때, 이를 어떻게 쓸 것인지를 결정해야 한다.

- NC 가 앱의 전체 실행과정에서 상당한 부분을 차지하는 경우

1. Window 의 root VC 로 설치한다.
2. 탭바 인터페이스에 탭의 VC 로 설치한다.
3. 스플릿 뷰 인터페이스에서 2 개의 root VC 중 하나로 설치한다 (iPad 만).

- NC 를 짧은 순간만 사용하는 경우

1. 다른 VC 로 부터 modal 하게 present
2. Popover 로 부터 표시 (iPad 만)

NI 용 컨텐츠 VC 정의하기.

모든 NI 는 root level 에 해당하는 데이터 level 1 개는 가진다. 이 level 은 사용자의 인터페이스 시작점이라 할 수 있다. 예를들어, 사진 어플은 자신의 데이터 계층에서 root level 로 포토앨범을 가진다. 포토 앨범을 선택하면 해당 앨범의 사진들이 표시되고, 그 사진들 중 하나를 선택하면 사진 한장을 크게 표시하게 된다.

NI 를 구현하려면, 데이터 계층의 각 단계에서 어떤 데이터를 표시할 지 결정해야 한다. 각 단계마다, 해당 단계의 데이터를 관리/표시할 컨텐츠 VC 가 제공되어야 한다. 만일 여러 단계에 걸쳐 동일한 화면표시가 이루어 진다면, 동일한 VC 클래스를 여러개 만들어, 각 객체마다 자신만의 데이터 집합을

관리하도록 설정해 주어야 한다. 예를들어, 사진 어플은 3 개의 독립적인 표시 유형을 가진다(위 그림참조). 이 경우에는 각 단계마다 모든 다른 방식의 표시방식이 필요하다.

각 컨텐츠 VC 는, 맨 끝단계를 제외하고는, 사용자에게 데이터 계층내 다음 단계로 옮겨갈 수 있는 방법을 제공해야 한다. 항목의 목록을 표시하는 VC 는 주어진 각 테이블 셀을 탭하면 다음 단계의 데이터를 표시할 수 있다.

스토리보드를 사용해 NI 생성

Xcode 에서 Master-Detail 어플리케이션 템플릿을 사용하면 NC 를 스토리보드상에서 사용하고 첫 장면으로 설정한 프로젝트가 생성된다.

스토리보드상에 직접 NC 를 만들려면 다음과 같이 한다.

1. NC 를 라이브러리에서 드래그
2. IB 에서 NC 와 VC 를 만들고, 이들간 관계를 설정한다. 이 관계는 새로이 생성된 VC 를 NC의 root VC로 인식한다.
3. 어트리뷰트 인스펙터에 Initial VC 옵션을 선택하여 생성한 NC 를 첫번째 VC 로 표시한다.

NI 를 코드로 생성하기.

코드상 적절한 위치에서 NC 를 직접 생성할 수도 있다. 예를 들면, NC 가 앱 Window 의 root view 를 제공한다면, NC 를 App Delegate의 `applicationDidiFinishLaunching:` 메소드에서 생성할 수 있다.

1. NI 를 위한 root VC 를 생성

   이 객체는 NS에서 최상위 VC가 된다. 네비게이션 바는, view 가 표시될 때 뒤로가기 버튼을 표시하지 않으며, NS 에서 pop 될 수 없다.

2. NC 를 생성하고, `initWithRootViewController:` 메소드를 사용해 초기화 한다.

3. NC 를 사용자 윈도우의 root VC 로 설정한다(또는 사용자의 인터페이스에서 present 한다).

아래는 `applicationDidiFinishLaunching:` 메소드에서 NC 를 만들고 앱의 main wnidow 에 대한 root VC 로 설정하는 코드다. `navigationController` 와 `window` 변수는 App Delegate 클래스의 멤버변수이고, `MyRootViewController` 클래스는 커스텀 VC 클래스이다. 이 예제에서 Window 가 표시될 때, NI 는 root VC 에 대한 view 를 present 한다.

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

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

네비게이션 뷰에 전체화면 레이아웃을 적용하기.

NI 는 네비게이션바 아래와 툴바 혹은 탭바의 위쪽 사이 공간에 커스텀 컨텐츠 VC 를 표시한다. 하지만, 전체 화면 레이아웃으로도 표시되게 할 수 있다. 전체화면 레이아웃에서 컨텐츠 뷰는 네비게이션바, 상태바, 툴바등과 적절히 겹쳐진 형태로 표시된다. 가시 영역을 최대화 할 수 있는 것이다.

NC 는 화면 전체 혹은 대부분을 채우도록 크기조정되어야 하는지 결정할 때, 다음과 같은 몇가지 조건들을 고려한다.

- 깔린 윈도우(혹은 부모 뷰)가 전체 화면을 채우는가?
- 네비게이션 바가 반투명으로 설정되었는가?
- 네비게이션 툴바가 (있다면) 반투명으로 설정되었는가?

- 밑에 깔린 VC의 `wantsFullScreenLayout` 속성이 YES 인가?

커스텀 뷰의 최종 크기는 이러한 조건들을 고려하여 결정된다. 위의 조건들의 나열 순서는 고려 순서이다. 윈도우 크기는 첫번째 제한 요소이다. 앱의 메인 윈도우(또는 modal하게 present된 VC)가 스크린을 다 덮지 않으면, 윈도우내의 View들도 화면을 채울 수 없다. 마찬가지로, 네비게이션바와 툴바가 불투명하게 표시되면 VC 가 자신의 View 를 전체화면 레이아웃을 써서 표시되게 하려고 해도 할 수가 없다. VC 는 불투명한 네비게이션바 밑에 컨텐츠를 표시하지 않게 된다.

NI 를 생성하여 커스텀 컨텐츠를 거의 대부분의 화면에 채우려면 다음과 같은 단계를 처리해야 한다.

1. 커스텀 뷰의 프레임을 전체 스크린 크기로 설정한다.

   이 경우, 사용자 View 의 autoresizing 어트리뷰트도 함께 설정해야 한다. 이 속성은 view 의 크기를 조정하려 할 때, 자신의 컨텐츠도 그에 맞추어 크기조정되도록 해 준다. 또 다른 방법으로는, view 의 크기가 변경될 때, 그 view의 `setNeedsLayout` 메소드를 호출하여 subview 들의 위치가 재조정되도록 하는 방법도 있다.

2. 네비게이션바 영역에도 표시되게 하려면, NC 의 `translucent` 속성을 `YES` 로 설정한다.

3. 선택적인 툴바를 활성화하고 그 영역에 커스텀 컨텐츠가 표시되게 하려, 툴바의 `translucent` 속성을 `YES` 로 설정한다.

4. 상태바 영역에 컨텐츠를 표시하려면, VC 의 `wantsFullScreenLayout` 속성을 `YES` 로 설정한다.

NI 를 present 할 때, 네비게이션 view 를 추가하는 윈도우 또는 view는 반드시 적당한 크기를 가져야 한다. 만일 앱이 NC 를 메인 인터페이스로 사용한다면, 메인 윈도우는 물리적 스크린 크기에 맞추어져야 한다. 그렇게 하려면, `UIScreen` 클래스의 `bounds` 속성값을 가져와서 view 의 크기를 설정해야 한다(`applicationFrame` 속성이 아니라!). 사실, NI 는 NC가 자신의 view

들을 상태바를 고려해서 크기를 자동으로 조정하기 때문에, 어떤 경우라도 윈도우를 아예 전체화면 크기로 생성하는 편이 낫다.

NC 를 modal 하게 present 한다면, NC 가 표시하는 컨텐츠는, prensent 하는 VC 쪽에 의해 컨텐츠 표시 영역이 정해진다. 만일 VC 가 상태바영역까지 표시되지 않도록 설정되었다면, 그 VC 가 modal 하게 present 하는 VC 도 마찬가지로 설정된다. present 되는 view 는 자신의 부모 view 에 의해서 항상 표시방법이 영향을 받게 된다.

전체 화면 레이아웃 지원을 위한 인터페이스 설정법에 대한 상세한 내용은 *View Controller Programming Guide for iOS* 의 "Creating Content View Controller" 편을 참조.

NS 수정하기.

NS 에 저장된 객체들은 사용자에 의해 만들어진다. NC 객체가 초기화될 때, 데이터 계층의 root 컨텐츠를 표시하기 위한 컨텐츠 VC 를 만들어야 한다. 그런 다음, 사용자의 상호작용에 의해, 혹은 코드상으로 VC 를 추가 하거나 제거하게 된다. NC 는 NS 의 컨텐츠를 관려하기 위한 몇가지 옵션을 제공한다. 이 옵션들은 앱에서 발생할 수 있는 다양한 시나리오에 대응하기 위한 것들이다. 아래 표 참조.

시나리오	설명
다음 수준의 계층적 데이터 를 표시한다.	사용자가 최상위 VC 가 표시하는 항목을 선택하면, seg
계층을 한 수준 되돌아 올라간다	사용자가 최상위 VC 가 표시하는 항목을 선택하면 segu
NS 를 이전상태로 복구한다	앱이 실행될 때, `setViewControllers:animated` 메소드 앱을 이전상태로 돌리기 위해서는 필요한 VC 를 다시 이 메소드를 사용하면 데이터 계층내에서의 임의 위치
Root VC 로 되돌아 간다	NI 의 최상위 단계로 되돌아가려면, `popToRootViewCon`
계층구조에서 임의갯수의 단계를 되돌린다.	한번에 한 수준이상의 전 단계로 되돌아가려면, `popToV`

VC 의 push/pop 동작이 애니메이션될 때, NC 가 가장 적당한 애니메이션을 자동 생성한다. 예를 들어, `popToViewController:animated` 메소드를 사

용해, 여러 VC 를 스택에서 제거하면, 그 다음 보여질 최상위 VC 에 대해서만 애니메이션 효과를 주게된다. 다른 임시적인 VC 들은 애니메이션 효과 없이 그냥 dismiss 시킨다. 애니메이션이 끝나기 전 까지는 다른 VC의 push 혹은 pop 동작을 수행할 수 없다.

NS에 대한 변경사항 모니터링

VC를 push 또는 pop 할 때는 항상, VC가 해당 VC에게 메시지들을 보내게 되어 있다. NC 는 스택에 변경이 생기면 자신의 `delegate` 에게도 메시지를 보낸다. 다음 그림은 push 또는 pop 동작중 발생하는 일련의 이벤트들의 처리 과정과 각 단계에서 사용자 커스텀 객체들에 보내지는 대응 메시지들을 나타낸다. 아래에서 "New View Controller"는, 스택의 최상단 VC가 되려고 하는 VC 를 나타낸다.

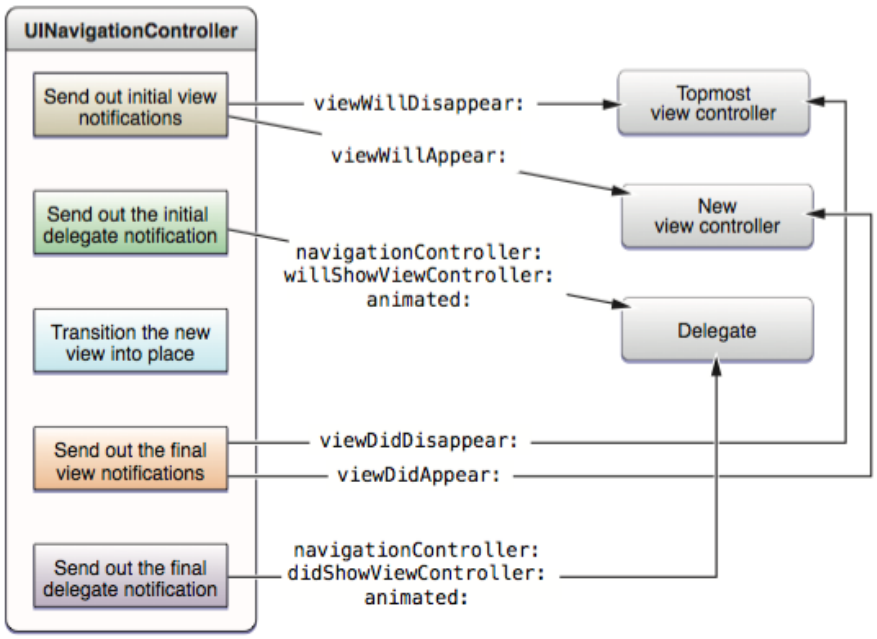


그림 1: 스택의 변경이 생기는 동안 전달되는 메시지

VC의 delegate에 있는 메소드를 사용해서 컨텐츠 VC들간의 동작을 조

율할 수 있다(예를 들어, 이들간의 공유되는 상태를 갱신한다던지...). 여러 VC들을 한번에 push 또는 pop 하면, 화면에 표시되던 VC와 새로이 표시될 VC가 이 메소드들의 호출을 받는다. 그 사이의 중간 VC 들은 메소드 호출을 받지 않는다. 단, 콜백함수내에서 연결동작(예를들면, `viewWillAppear:` 가 `pushViewController:animated:` 를 호출하도록 구현한 경우)이 일어나는 경우에는 중간 VC라도 위와 같은 호출이 일어날 수 있겠다.

`UIViewController` 의 `isMovingToParentViewController` 와 `isMovingFromParentViewController` 메소드를 사용하여 VC가 push 또는 pop 의 결과로 화면에 표시되거나 감춰지는 지를 알 수 도 있다.

네비게이션바 모양 바꾸기

네비게이션바는 NI 의 제어를 관리하는 view. NC 객체에 의해 관리될 때는 특별한 역할을 수행. 일관된 인터페이스를 유지하고, NI 를 구축할때 필요한 코딩을 줄이기 위해, 각 NC 객체는 스스로 자신만의 네비게이션바를 생성하고 그 내용을 관리하는 작업의 대부분을 수행한다. 필요하면, NC 는 다른 객체(컨텐츠VC 같은 객체)와 상호작용하여 이 과정을 원활히 해 준다.

주의 독립된 view로 네비게이션바를 만들고 원하는 방식대로 사용할 수도 있다. `UINavigationBar` 클래스의 메소드와 속성에 대해 상세한 내용은 *UINavitaionBar Class Reference* 를 참조.

1. 네비게이션 아이템 객체 관리하기 네비게이션바의 구조는 NC의 구조와 여러면에서 유사하다. NC처럼 네비게이션바는 다른 객체에 의해 제공되는 컨텐츠의 컨테이너역할을 한다. 네비게이션 바의 경우, 컨텐츠는 하나 이상의 `UINavigationItem` 객체에 의해 제공되는데, 이는 네비게이션 아이템 스택(navigation item stack)이라는 스택 자료구조를 사용해 저장된다. 각 네비게이션 아이템은 view들과 컨텐츠로 구성된 완전한 집합을 제공하여 네비게이션 바에 표시될 수 있다.

   다음 그림은 런타임시 네비게이션바와 관련된 주요 객체들을 표시한다. 네비게이션바의 소유자(NC일 수도 있고, 사용자의 코드일 수도 있다)는 이들 주요 객체들을 스택상에, 필요에 따라 push 하거나 pop 한다.

네비게이션 동작을 올바로 수행하기 위해 네비게이션바는 스택내의 객체들을 가리키는 포인터들을 관리한다. 네비게이션바의 컨텐츠 대부분은, 최상위 네비게이션 아이템으로 부터 가져오지만, 뒤로가기 버튼이 이전항목에 대한 올바른 제목을 가질 수 있도록 이정 항목에 대한 포인터 역시 관리된다.

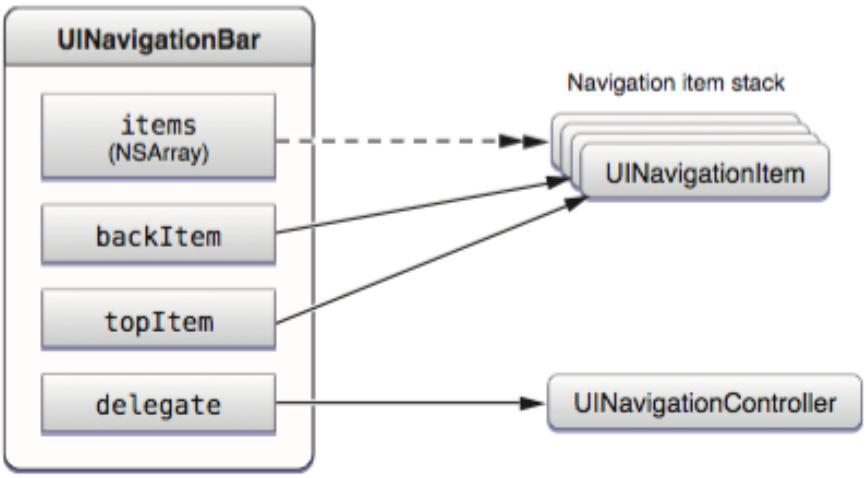


그림 2: 네비게이션바와 관련된 객체들

중요 NC 와 함께 네비게이션바를 사용하는 경우, 네비게이션바의 `delegate` 는 항상 NC 가 된다. 이걸 바꾸면 예외가 발생한다.

NI 에서, NS 에 저장된 각 컨텐츠 VC 는, 자신의 `navigationItem` 속성의 값에 네비게이션 아이템의 값을 가지고 있다. NS 와 네비게이션 아이템 스택은 항상 함께 유지된다. 즉, NS 상에서 각 컨텐츠 VC 의 위치는, 항상 네비게이션 아이템 스택내에 동일한 위치에 자신의 네비게이션 아이템이 위치한다.

네비게이션 바는, 좌측, 중앙, 그리고 오른쪽의 3 가지 주요한 위치를 제공한다. 다음 표는, 이 위치들을 설정하는데 사용되는 `UINavigationItem` 클래스의 속성들을 나열하고 있다. NC 에 대해 네비게이션 아이템을 설정할 때는, 어떤 위치에 커스텀 컨트롤이 다른 컨트롤에 의해 무시되는

경우도 있다는 점에 유의한다. 각 위치는 커스텀 객체가 어떻게 사용되는지에 대한 정보를 기술한다.

표 1: 네비게이션바에서의 아이템의 위치

위치	속성	설명
좌측	`backBarButtonItem` `leftBarButtonItem`	네비게이션 인터페이스에서 네비게이션 컨트롤러는
중앙	`titleView`	네비게이션 인터페이스에서, 네비게이션 컨트롤러는
우측	`rightBarButtonItem`	이 위치는 디폴트로는 비어있는 상태이다. 이 위치

다음그림은 네비게이션바의 컨텐츠가 NI 를 어떻게 구성하는지를 보여준다. 현재 VC 와 연관된 네비게이션 아이템은 네비게이션바 중앙과 오른쪽에 온다. 이전 VC 와 관련된 네비게이션 아이템은 왼쪽 위치에 있다. 왼쪽과 오른쪽 아이템이 `UIBarButtonItem` 객체를 사용해야 하기는 하지만, 아래 그림과 같이 버튼 안쪽에 view 로 다른 객체를 감싸안을 수도 있다(근데, Xcode 문서를 보면, `UIBarButtonItem` 은 `contentView` 속성이 아니라, `customView` 속성을 가지고 있는 것 같은데 ?).

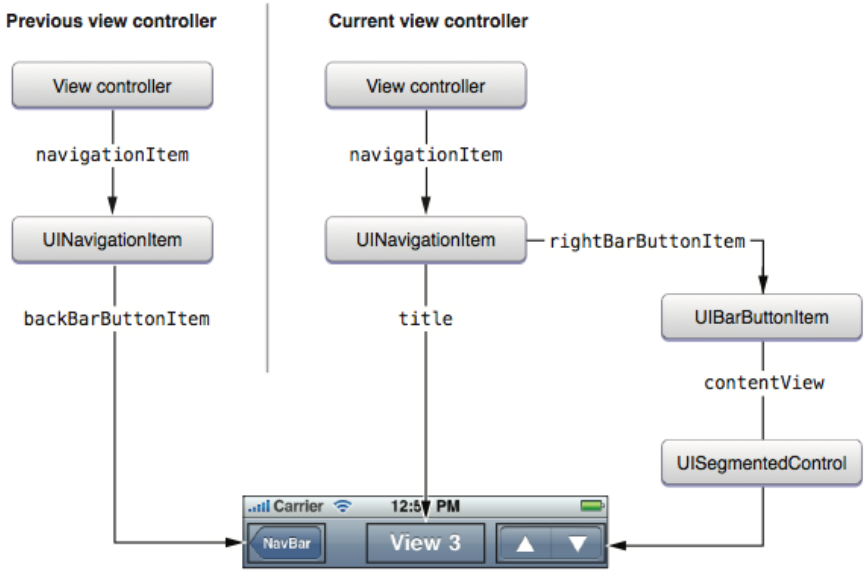


그림 3: 네비게이션바 구조

2. 네비게이션바 표시하기와 감추기 네비게이션바를 NC 에서 사용할 경우, 항상 `UINavigationController` 의 `setNavigationBarHidden:animated:` 메소드를 사용해서 네비게이션바를 보여주거나 감출 수 있다. **주의** `UINavigationController` 의 `hidden` 속성을 바로 건드려서 네비게이션바를 감추면 안된다. 한편, 네비게이션바를 보여주거나 감추는 것 외에 NC 는 좀 더 정교한 동작을 제공하는데, `viewWillAppear:` 메소드에서 네비게이션바를 보이거나 감추게 되면, NC 는 새로운 VC 가 표시되는 때에 맞추어 네비게이션바를 보이거나 감춘다.

   이전의 화면으로 돌아가려면 뒤로가기 버튼이 있는 네비게이션바가 필요하기 때문에, 사용자에게 그렇게 할 수 있는 다른 방법을 제공하기 전에는 네비게이션바를 감추면 안될 것이다. 다른 방법으로 가장 많이 사용되는 것으로는, 터치 이벤트를 가로채어 네비게이션바의 표시여부를 결정하는 방법이 있다. 예를 들어, 사진 어플의 경우 하나의 이미지가 전체 화면으로 보여져야 할 때가 있는데, 그런 경우 이 방법을 쓸 수 있을 것이다. 또는 쓸어넘기기(swipe) 제스쳐를 검출하여 현재 VC를 스택에서 pop 하는 식으로 구성할 수 도 있을 것이다. 하지만, 이 방법은 네비게이션바를 보였다/안보였다 하는 방식보다는 사용자가 조작법이 익숙치 않을 수도 있다.

3. 네비게이션바 객체를 직접 수정하기 NI 상에서, `UINavigationBar` 객체를 소유하고 관리할 책임이 NC에 있다. 네비게이션바 객체를 (다른 객체로) 바꾸거나, bound, frame[1] 또는 알파값을 직접 수정하는 것은 허용되지 않는다. 하지만, 수정할 수 있는 속성들이 몇가지 있다.

   - `barStyle` 속성
   - `translucent` 속성
   - `tinColor` 속성

[1] UIView의 bound는 자기자신의 좌표계. frame은 superview의 좌표계로 표시되는 자신의 사각형 영역이다.

다음 그림은 barStyle 과 translucent 속성 변화에 따라 네비게이션바의 모양이 어떻게 바뀌는지를 나타낸다. 반투명 스타일에서는 밑에 배치된 VC 가 스크롤뷰 일 경우, 컨텐츠가 스크롤되어 밖으로 나가는 것을 허용하기 위해 네비게이션바 스스로 자신의 모양을 자동적으로 조절한다는 점을 기억하자.

그림 4: 네비게이션바 스타일

네비게이션바 전체를 보이거나 감추려면, 직접 객체를 다루기 보다는 NC 의 setNavigationBarHidden:animated 메소드를 사용하는 것이 바람직하다. 여기에 대해서는 "네비게이션바 표시하기와 감추기" 부분을 참조한다.

a) 커스텀 버튼 및 커스텀 뷰를 네비게이션 아이템으로 사용하기 특정 VC 를 위해 네비게이션바의 모양을 바꾸려면, 연관된 UINavigationItem 객체의 속성을 수정하면 된다. 특정 VC 에 대한 네비게이션 아이템을 얻으려면, 해당 UIViewController 의 navigationItem 속성값을 참조하면 된다.