Developing Applications for iOS

Lecture 7:
View Controller Lifecycle and UIKit

Radu Ionescu
raducu.ionescu@gmail.com
Faculty of Mathematics and Computer Science
University of Bucharest

Content

- View Controller Lifecycle

  When your controller hears about things and what you should do about it.

- Image View

  Kind of like “`UILabel`” for images.

- Web View

  Complete browser in a view.

- Scroll View

  Provides a moving “viewport” on a rectangular area that has views (the scroll view’s subviews) in it.

View Controller Lifecycle

View Controllers have a Lifecycle

- A sequence of messages is sent to them as they progress through it.

Why does this matter?

- You very commonly override these methods to do certain work:

```
- (void)viewDidLoad;
- (void)viewWillAppear:(BOOL)animated;
- (void)viewDidAppear:(BOOL)animated;
- (void)viewWillDisappear:(BOOL)animated;
- (void)viewDidDisappear:(BOOL)animated;
- (void)viewDidUnload;
```

and many other methods.

View Controller Lifecycle

We have already talked about the first part of the lifecycle

- Creation
- This is done mostly either via a segue or storyboard's:

  `instantiateViewControllerWithIdentifer:`
- Because of this, we rarely override `UIViewController`'s designated initializer in iOS 5.
- Another option is `awakeFromNib`, but we rarely do that either.
- If you cannot define your views in a storyboard or a nib file, override the `loadView` method to manually instantiate a view hierarchy and assign it to the `view` property.
- There are better methods to initialize in: `viewDidLoad`, `viewWillAppear:`, `viewDidAppear:`.

View Controller Lifecycle

- After instantiation and outlet-setting, `viewDidLoad` is called:

```
- (void)viewDidLoad;
```

- This method is called regardless of whether the view hierarchy was loaded from a nib file or created programmatically in the `loadView` method.
- This is an exceptionally good place to put a lot of setup code.
- But be careful because the geometry of your view (its `bounds`) is not set yet!
- If you need to initialize something based on the geometry of the view, use the next method.

View Controller Lifecycle

- Just before the view appears on screen, you get notified:

```
- (void)viewWillAppear:(BOOL)animated;
```

- When this is called, your `bounds` has been set (via your `frame` by your `superview`).
- Your view will probably only get “loaded” once, but it might appear and disappear a lot. So don’t put something in this method that really wants to be in `viewDidLoad`.
- Otherwise, you might be doing something over and over unnecessarily.
- Use this to optimize performance by waiting until this method (i.e. just before view appears) to kick off an expensive operation (might have to put up a spinning “loading” icon though).
- This method is for geometry-related initialization and lazy execution for performance.

View Controller Lifecycle

- And you get notified when you will disappear off screen too. This is where you put “remember what’s going on” and cleanup code.

```
- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
    // Call super in all the viewWill/Did... methods.

    /* Let’s be nice to the user and remember
     * the scroll position they were at */
    [self rememberScrollPosition];
    // We will have to implement this, of course.

    /* Do some other clean up now that we have
     * been removed from the screen.
    [self saveDataToPermanentStore];
    /* But be careful not to do anything
     * time-consuming here, or app will be sluggish.
     * Do it in the did version or in a thread. */
}
```

View Controller Lifecycle

- There are “did” versions of both of the previous methods.
- You can override this method to perform additional tasks associated with presenting the view:

```
- (void)viewDidAppear:(BOOL)animated;
```

- You can override this method to perform additional tasks associated with dismissing or hiding the view:

```
- (void)viewDidDisappear:(BOOL)animated;
```

- If you override these methods, you must call super at some point in your implementation.

View Controller Lifecycle

Frame changed?

- Here's a good place to layout subviews manually (if struts and springs are not enough):

```
- (void)view{Will,Did}LayoutSubviews;
```

- When a view's `bounds` change, the `view` adjusts the position of its `subviews`. Your View Controller can override these methods to make changes before/after the view lays out its subviews.
- Called any time a view's `frame` changed and its `subviews` were thus re-layed out.
- For example, autorotation.
- You can reset the `frames` of your `subviews` here.

View Controller Lifecycle

Specific notification that rotation will/did happen

```
- (void)willRotateToInterfaceOrientation:
                        (UIInterfaceOrientation)anOrientation
                            duration:(NSTimeInterval)seconds;

- (void)willAnimateRotationToInterfaceOrientation:
                  (UIInterfaceOrientation)toInterfaceOrientation
                            duration:(NSTimeInterval)seconds;

- (void)didRotateFromInterfaceOrientation:
                        (UIInterfaceOrientation)anOrientation;

@property UIInterfaceOrientation interfaceOrientation;
```

- The `interfaceOrientation` property will have the current (old or new) orientation when each of the above is called.
- Example use: stop doing anything expensive (e.g. an animation maybe?) in will and resume it in did.

View Controller Lifecycle

In low-memory situations, `viewDidUnload` gets called

- Right after your `UIViewController`'s `self.view` is set to `nil` (hopefully freeing up its heap usage). This can only happen if your MVC is not on-screen, of course!
- This rarely happens, but well-designed code will anticipate it.
- Even though your outlet pointers are `weak` and will probably get set to `nil` automatically, it is "good practice" to set them to `nil` here so that we know this for sure.
- For example, CalculatorViewController's `viewDidUnload` should probably look like this:

```
- (void)viewDidUnload
{
    self.display = nil;
    ...
}
```

This line was added by CTRL-dragging from Interface Builder to create an outlet. The same for other outlets.

- If the `UIViewController` goes back on-screen, `viewDidLoad` will be called again.

View Controller Initialization

Creating a `UIViewController` from a .xib file

- This is the old, iOS 4 way. Not covered in this class.
- You create a .xib file with the same name as your `UIViewController` subclass. Then use `alloc`/`init` to create it.
- Designated initializer (only if you need to override it; use `init` otherwise):

```
- (id)initWithNibName:(NSString *)nibName
                bundle:(NSBundle *)bundle;
```

View Controller Initialization

Creating a `UIViewController`'s UI in code (no .xib, no storyboard)

- Override the method `- (void)loadView` and set `self.view`.
- Do **NOT** implement `loadView` if you use a storyboard/.xib to create the `UIViewController`.
- Do **NOT** set `self.view` anywhere else besides in `loadView`.
- Do **NOT** implement `loadView` without setting `self.view` (i.e. you must set `self.view` here).
- You should **NEVER** call this method directly. The View Controller calls this method when its `view` property is requested but is currently `nil`.

View Controller Initialization

- Avoid `awakeFromNib` if possible.
- It is an acceptable place to initialize stuff for a `UIViewController` from a storyboard/.xib.
- But try to put stuff in `viewDidLoad`, `viewWillAppear:` or the segue preparation code (`prepareForSegue:sender:`) instead.

UIView's frame

Who's responsible for setting a UIView's frame?

- The object that puts the UIView in a view hierarchy.
- In Interface Builder, you set all view's frames graphically.
- You do this by dragging on the little handles (or from Size Inspector).

What about the frame passed to initWithFrame:?

- If you're putting it into a view hierarchy right away, pick the appropriate frame.
- If you are not, then it doesn't really matter what frame you choose (but avoid CGRectZero).
- The code that eventually does put you in a view hierarchy will have to set the frame.

UIView's frame

Setting frames in viewDidLoad

- Recall that your final bounds are not set in viewDidLoad.
- If you create views in code in viewDidLoad, pick sensible frames based on the view's bounds then.
- But be sure to set struts/springs (UIView's autoresizingMask property).
- You specify the value of this mask by combining the constants described in UIViewAutoresizing using the C bitwise OR operator.
- Think of adding something in viewDidLoad as the same as laying it out in Interface Builder.
- In both cases, you have to anticipate that the top-level view's bounds will be changed.

UIImageView

A UIView subclass which displays a UIImage

- We covered how to create a UIImage in the lecture on Views.

How to set the UIImageView's UIImage

Use this initializer:

```
- (id)initWithImage:(UIImage *)image;
```

- It will set its frame's size to match image's size. Note that the designated initializer is still initWithFrame: (inherited from the UIView superclass).
- You can also set this property (but it will not adjust the frame size):

```
@property(nonatomic, strong) UIImage *image;
```

UIImageView

Remember UIView's contentMode property?

- It can be set to UIViewContentMode{Redraw,Top,Left,...}.
- UIViewContentModeScaleToFill is the default.
- Determines where the image appears in the UIImageView's bounds and whether it is scaled.

Highlighted image

```
@property(nonatomic,strong) UIImage *highlightedImage;
@property(nonatomic,getter=isHighlighted) BOOL highlighted;
```

- Note that you can also rename setters or getters of a @property. The naming convention is not to include verbs in a property (this is why the getter is renamed).
- To get the highlighted property of myImageView you would use:

```
myImageView.isHighlighted
```

UIImageView

Sequence of images forming an animation

- For animating more images, set the `animationImages NSArray @property`.
- `UIImageView` class provides controls to set the duration and frequency of the animation:

```
@property(nonatomic) NSTimeInterval animationDuration;
@property(nonatomic) NSInteger animationRepeatCount;
```

The default value is 0, which specifies to repeat the animation indefinitely.

- You can also start and stop the animation:

```
- (void)startAnimating;
- (void)stopAnimating;
- (BOOL)isAnimating;
```

The `startAnimating` method always starts the animation from the first image in the list.

UIWebView

A full internet browser in a `UIView`

- Can use it not only to take your users to websites, but to display PDFs, for example.
- Built on WebKit, an open source HTML rendering framework (started by Apple).
- Supports JavaScript, but limited to 10 seconds or 10 megabytes of memory allocation:

```
- (NSString *)stringByEvaluatingJavaScriptFromString:
                                        (NSString *)script;
```

When scripts are running, the user is not able to interact with the webpage (this is why the limitation is imposed).

- Example:

```
NSString *title = [webView
stringByEvaluatingJavaScriptFromString:@"document.title"];
```

UIWebView

- Property to control whether page is scaled to fit the `UIView`:

```
@property(nonatomic) BOOL scalesPagesToFit;
```

  If `YES`, then page will be squished or stretched to fit the width of the `UIView`.

  If `NO`, the page will be its natural size and the user cannot zoom inside it. The default value is `NO`.

- Property to get the scroll view it uses

```
@property(nonatomic, readonly, strong)
                                UIScrollView *scrollView;
```

  Can now set properties in the scroll view to control the scrolling behavior of the web view.

- If you allow the user to move back and forward through the webpage history, then you can use the `goBack` and `goForward` methods as actions for buttons.

UIWebView

- Three ways to load up HTML:

```
- (void)loadRequest:(NSURLRequest *)request;

- (void)loadHTMLString:(NSString *)string
               baseURL:(NSURL *)baseURL;

- (void)loadData:(NSData *)data
        MIMEType:(NSString *)MIMEtype
textEncodingName:(NSString *)encodingName
         baseURL:(NSURL *)baseURL;
```

- We'll talk about `NSURLRequest` in a moment.
- Base URL is the "environment" to load resources out of (i.e., it's the base URL for relative URL's in the data or HTML string).
- MIME type (Multimedia Internet Mail Extension) says how to interpret the passed-in data.

  Standard way to denote file types (like PDF). Think "e-mail attachments" (that's where the name MIME comes from).

UIWebView

NSURLRequest

```
+ (NSURLRequest *)requestWithURL:(NSURL *)url;
+ (NSURLRequest *)requestWithURL:(NSURL *)url
                     cachePolicy:(NSURLRequestCachePolicy)policy
                 timeoutInterval:(NSTimeInterval)timeoutInterval;
```

NSURL

- Basically like an `NSString`, but enforced to be “well-formed”.
- Examples: file://... or http://...
- In fact, it is the recommended way to specify a file name in the iOS API.

```
+ (NSURL *)URLWithString:(NSString *)urlString;
+ (NSURL *)fileURLWithPath:(NSString *)path
                isDirectory:(BOOL)isDirectory;
```

NSURLRequestCachePolicy

- Ignore local cache; ignore caches on the internet; use expired caches; use cache only (don’t go out onto the internet); use cache only if validated.

UIWebView

UIWebViewDelegate

- You can set the `delegate` property to an object conforming to the `UIWebViewDelegate` protocol if you want to track the loading of web content.
- Methods in the `UIWebViewDelegate` are:

```
- (BOOL)webView:(UIWebView *)webView
  shouldStartLoadWithRequest:(NSURLRequest *)request
  navigationType:(UIWebViewNavigationType)navigationType;

- (void)webViewDidStartLoad:(UIWebView *)webView;

- (void)webViewDidFinishLoad:(UIWebView *)webView;

- (void)webView:(UIWebView *)webView
  didFailLoadWithError:(NSError *)error;
```

- The navigation type gives you a hint about the user action that generated the request:

```
UIWebViewNavigationType{LinkClicked,FormSubmitted,
  BackForward,Reload,FormResubmitted,Other}
```

UIScrollView

How do you create one?

- Just like any other `UIView`. Drag out in a storyboard or use `alloc/initWithFrame:`.
- Or select a `UIView` in your storyboard and choose “Embed In > Scroll View” from Editor menu.
- Or add your “too big” `UIView` using `addSubview:` like this:

```
UIImage *image = [UIImage imageNamed:@“bigImage.jpg”];
UIImageView *imageView = [[UIImageView alloc]
                          initWithImage:image];
// now imageView.frame.size is equal to image.size
[scrollView addSubview:imageView];
```

- Add more subviews if you want.

UIScrollView

- All of the `frames` of the `subviews` will be in the `UIScrollView`'s content area's coordinate system.
- (0,0) is the upper left corner of the scroll view.
- Width and height are given by `contentSize.width` and `contentSize.height`.
- Don't forget to set the `contentSize`!
- Common mistake is to do the above 3 lines of code (or embed in Interface Builder) and forget to say:

```
scrollView.contentSize = imageView.bounds.size;
```

UIScrollView

Scrolling programmatically

```
- (void)scrollRectToVisible:(CGRect)aRect
                   animated:(BOOL)animated;
```

Other things you can control in a scroll view

- Control whether scrolling is enabled through the `scrollEnabled` property.
- Lock scroll direction to user's first "move" by setting the `directionalLockEnabled` property.
- The style of the scroll indicators are set via the `indicatorStyle` property. (call `flashScrollIndicators` when your scroll view appears).
- Whether the actual content is "inset" from the scroll view's content area (`contentInset` property).
- Note that `UIScrollView` is the superclass of several UIKit classes including `UITableView` and `UITextView`.

UIScrollView

Zooming

- All `UIViews` have a property (`transform`) which is an affine transform (translate, scale, rotate). Scroll view simply modifies this transform when you zoom.
- Zooming is also going to affect the scroll view's `contentSize` and `contentOffset`.
- Will not work without minimum/maximum zoom scale being set

```
scrollView.minimumZoomScale = 0.5; //half its normal size
scrollView.maximumZoomScale = 2.0; //twice its normal size
```

- Will not work without delegate method to specify view to zoom:

```
-(UIView *)viewForZoomingInScrollView:(UIScrollView *)sender;
```

If your scroll view only has one subview, you return it here.

More than one subview? It's up to you then.

UIScrollView

Zooming programmatically

```
@property (nonatomic) float zoomScale;

- (void)setZoomScale:(float)scale
            animated:(BOOL)animated;

- (void)zoomToRect:(CGRect)zoomRect
          animated:(BOOL)animated;
```

Lots and lots of delegate methods!

- The scroll view will keep you up to date with what's going on.
- Example: delegate method will notify you when zooming ends

```
- (void)scrollViewDidEndZooming:(UIScrollView *)sender
                       withView:(UIView*)zoomView
                        atScale:(CGFloat)scale;
```

- If you redraw your view at the new `scale`, be sure to reset the affine transform back to identity.

Next Time

iDevice Capabilities:

- Core Location: GPS + Compass
- Accelerometer
- Map Kit