

FastPdfKit activation procedure

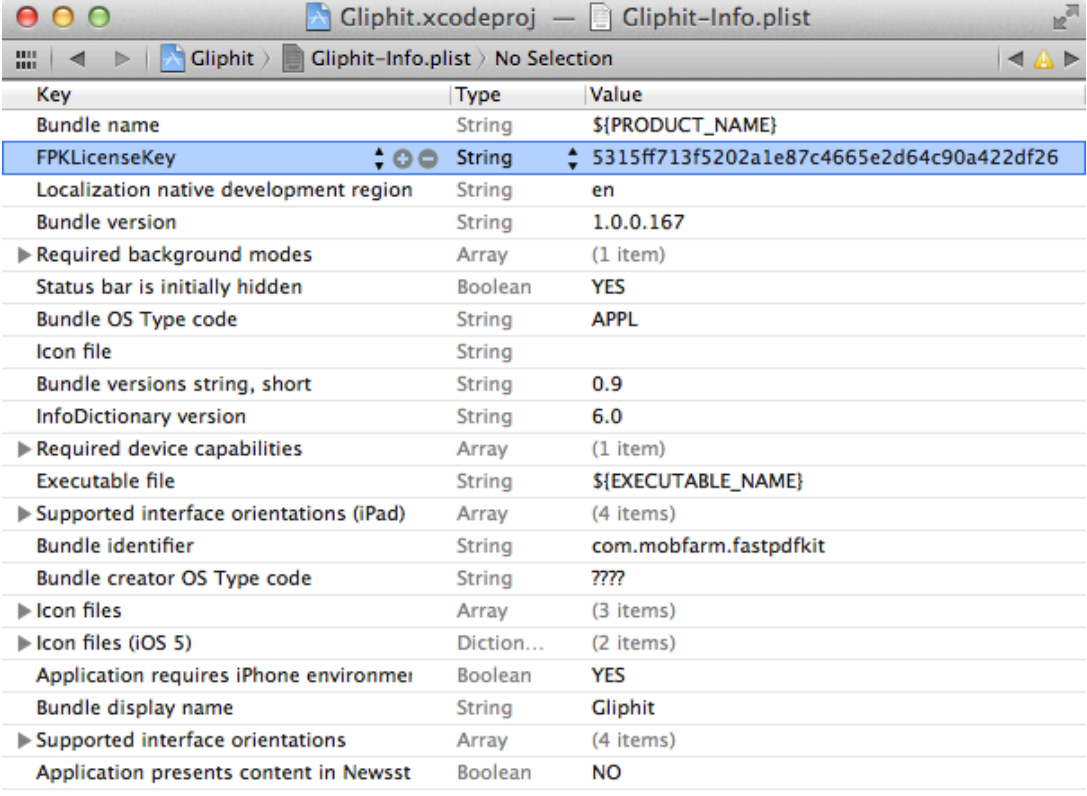
Since FastPdfKit 0.X the activation process was slow and time consuming.

With version 3.0, we've added a completely new procedure, that is immediate and far more easy to follow.

- Purchase a license on the [online store](#) with PayPal, AmEx or Bank Transfer;
- Let us verify the transaction (in realtime if you use PayPal);
- Receive the activation code by email.

Then you just need to add in your application **Info.plist** file a new key called `FPKLicenseKey` of String type. As value put the activation code.

The result will be like the figure below.



| Key | Type | Value |
|---|------------|--|
| Bundle name | String | \$(PRODUCT_NAME) |
| FPKLicenseKey | String | 5315ff713f5202a1e87c4665e2d64c90a422df26 |
| Localization native development region | String | en |
| Bundle version | String | 1.0.0.167 |
| ▶ Required background modes | Array | (1 item) |
| Status bar is initially hidden | Boolean | YES |
| Bundle OS Type code | String | APPL |
| Icon file | String | |
| Bundle versions string, short | String | 0.9 |
| InfoDictionary version | String | 6.0 |
| ▶ Required device capabilities | Array | (1 item) |
| Executable file | String | \$(EXECUTABLE_NAME) |
| ▶ Supported interface orientations (iPad) | Array | (4 items) |
| Bundle identifier | String | com.mobfarm.fastpdfkit |
| Bundle creator OS Type code | String | ??? |
| ▶ Icon files | Array | (3 items) |
| ▶ Icon files (iOS 5) | Diction... | (2 items) |
| Application requires iPhone environmei | Boolean | YES |
| Bundle display name | String | Gliphit |
| ▶ Supported interface orientations | Array | (4 items) |
| Application presents content in Newsst | Boolean | NO |

If you lose the mail message your code will be always available at [#](http://fastpdfkit.com) Are internal url supported?

Yes, FastPdfKit recognizes tap on internal url and will bring you automatically to the destination page.

Availability and price

FastPdfKit is available for **FREE**...

Grab your copy from [github](#).

If you need a version without the logo, use our [online store](#) or [contact us](#), but be sure to place a *private* request.

Can I change the MFDocumentViewController's background

You can just comment the `@private` line on `MFDocumentViewController.h` and on your `DocumentViewController.m` add a line like

```
[pagedScrollView setBackgroundColor:[UIColor redColor]];
```

to change the background color.

Can I customize the interface?

Yes you can customize everything!

Can I use custom annotations to embed something?

Yes you can add custom link annotation with your desired url and then implement the method

```
-(void)documentViewController:(MFDocumentViewController *)dvc didReceiveURLRequest:(  
    NSString *)uri {  
    NSLog("The URI tapped is %@", uri);  
}
```

to perform your custom action.

Can i draw over a pdf page?

Yes providing a `MFOverlayDataSource` with array of drawables and touchables.

Look at the [documentation](#) for more details on methods.

You can also overlay `UIViews` directly over the page that will move with the document inside the scroll view. To do that use the `addOverlayViewDataSource:` method.

Change zoom level for search results

If you are using our sample `SearchViewController` and `MiniSearchView` (or just the

[ReaderViewController](#)) you can set the zoom level for searched words changing the

```
#define ZOOM_LEVEL 4.0
```

parameter in [SearchViewController.m](#) and [MiniSearchView.m](#).

Documentation

[Go to this page for the complete documentation](#)

[Get Xcode documentation with automatic updates](#)

Drawables and Touchables

Drawables and Touchables are two faces of the same medal: a way to draw something onto the pdf page and interact with it. An object implementing the [MFOverlayDrawable](#) protocol needs to define a `-(void)drawInContext:(CGContextRef)ctx` method, while the [MFOverlayTouchable](#) protocol define a `-(BOOL)containsPoint:(CGPoint)point` method. The `drawInContext` method is used to draw the drawable over the page, while the `containsPoint` method is used to check for user interaction. Since touchables and drawables are defined in page space coordinates, both methods should take this into account: if we want to draw an opaque red square in the bottom left of the page, the `drawInContext` method could look like this

```
-(void)drawInContext:(CGContextRef)ctx {  
    ...  
    CGContextSetRGBFillColor(ctx, 1.0, 0.0, 0.0, 1.0);  
    CGContextFillRect(ctx, CGRectMake(0,0,100,100));  
    ...  
}
```

while, if we want to make the touchable active in the same area of the above red square, we should fill the `containsPoint` method with something like this

```
-(BOOL)containsPoint:(CGPoint) point {  
  
    return CGRectContainsPoint(touchableRect,point);  
  
}
```

where the `touchableRect` is elsewhere defined with `CGRectMake(0,0,100,100)`.

To provide the [MFDocumentViewController](#) with drawable and touchable you just need to implement the [MFDocumentOverlayDataSource](#) in some object, and add it to the [MFDocumentViewController](#) with the `addOverlayDataSource` method (you can remove it with the `removeOverlayDataSource` method). When a page is drawn, each overlay data source is asked to provide an array of drawables and touchables for the page through the invocation of the following methods

```
-(NSArray *)documentViewController:(MFDocumentViewController *)dvc
drawablesForPage:(NSUInteger)page;

-(NSArray *)documentViewController:(MFDocumentViewController *)dvc
touchablesForPage:(NSUInteger)page;
```

It is a good idea to have touchable and drawables ready to be returned at any time, so don't do slow things like parsing xml files to generate them as needed. Overlay data sources will also be notified when an user tap a touchable through the invocation of -
(void)documentViewController:(MFDocumentViewController *)dvc
didReceiveTapOnTouchable:(id<MFOverlayTouchable>)touchable. Each overlay data source should check if the touchable is one of its own and handle the event accordingly

```
@interface MyTouchable : NSObject <MFOverlayTouchable> {

    CGRect touchableRect;
}

@property (nonatomic,readwrite) CGRect touchableRect;

@end

-(void)documentViewController:(MFDocumentViewController *)dvc
didReceiveTapOnTouchable:(id<MFOverlayTouchable>)touchable {

    if([touchables containsObject:touchable]) { // A set of touchables
associated to this overlay data source.

        MyTouchable * myTouchable = (MyTouchable *)touchable;

        NSLog(@"Touched touchable at rect
%@",NSStringFromRect(myTouchable.touchableRect));

    }
}
```

As bottom line, you can also have an object that implements both the [MFOverlayDrawable](#) and [MFOverlayTouchable](#) protocol, so if you want to draw something and make it interactive too, you just have to create a single class.

FastPdfKit 3.0 - What's new?

The just released version 3.0 includes three main features:

- Complete support for **custom multimedia annotations**;
- Redesigned and flexible **rendering engine**;
- **Gorgeous interface** for the [ReaderViewController](#) so you can use it as is.

Plus a bonus one:

- New **activation** method with code for immediate deploy.

Custom multimedia annotations

Now **FastPdfKit** has reached the state of the art: with a pdf and simple annotations you can create a full featured multimedia magazine.

- Create the document with InDesign or use an already authored pdf document;
- Add some custom uri annotations directly over the page with your preferred editor (Preview, InDesign, Adobe Acrobat, etc).
- Add support for that custom uri in your [MFDocumentViewController](#) subclass;
- Enjoy.

The key difference between version 2 and 3 is that now you can add overlay views based on the annotations before the user tap on them.

In **FastPdfKit** you have now three products:

- A **pdf reader** with search and text highlight;
- A customizable **multimedia magazine** viewer;
- A **kiosk** to download the issues.

The circle is closed.

Take a look at the [Gliphit](#) video: the first multimedia pdf created with **FastPdfKit 3.0**.

Redesigned rendering engine

We've spent some months to design and create the new rendering engine that let your display the documents in three way: single page, double page and **overflow**, with the page fitted by its width and aligned on the top. These modes are available with every device, every screen resolution, every view dimension and every orientation: you choose.

The most notable improvement is the ability to navigate through a pdf document like you do with iPad designed multimedia publications like Wired. Thanks to the overflow mode you swipe right to left to change page and up to down to read long articles, just setting the pdf page dimensions appropriately.

New interface for the default reader controller

FastPdfKit key feature is that is **fully customizable**, but sometimes you need to avoid customization to be ready in minutes. So we decided to provide you a new default interface for the [ReaderViewController](#) with gorgeous icons and redesigned controllers. I bet you are going to like it!

Realtime Activation

FastPdfKit now includes an activation method that doesn't need an updated binary, so you can just use the very same static library or framework you've tested and just add a **key** in your application's Info.plist file. Very convenient and immediate, as the code is generated as soon as the purchase is confirmed. It's in **realtime** as soon as the purchase is verified.### Reading

- Side page sliding
- [Single page](#)

- [Double page](#)
- Landscape/Portrait with [automode](#)
- Extreme speed
- Native pdf thumbnails extraction
- [High definition thumbnails](#) generation
- [Page preload](#)
- [Double tap](#) and pinch to zoom
- [Tap on the side](#) to go forward or backward
- [Zoom Lock with animation](#)
- [Control zoom level](#) for each page
- [Quick page change bar with thumbnails](#) and page numbers
- [Bookmarks](#)
- [Outline - Table Of Contents personalizzabili](#)
- [Left to Right and Right to left](#) reading
- [Switch left to right pages](#)
- [Shadown on pages customizable](#)
- [Padding settable](#)
- [Side areas to go forward and backward with custom dimension](#)
- [Directional lock](#)
- [Custom pdf background](#)
- [Precise zoom](#) on annotations and page parts
- [Auto mode](#): single page in portrait and double page in landscape
- [Goto page and zoom on rect](#)

Multimedia and Annotations

- [Custom annotation support](#)
- [Multimedia documents overlayed over the pages](#)
- YouTube videos
- Local videos
- Streaming videos
- Offline sound reproduction
- Music streaming
- Web pages and html5 in overlay
- Custom modal multimedia views
- No limits
- [You can customize protocols](#)
- [Touchable overlay views](#)
- [Draw over pdf layers](#) in page coordinates
- [Overlayed UIViews](#)
- [Embedded local audio Player](#) with [pause/play](#) and volume
- [Embedded remote audio player](#)

- Embedded movie player with fullscreen option
- Embedded fullscreen browser
- [Overlay on Tiled Layer on demand](#)
- [Tap](#) and [double tap](#) on annotations
- [Video](#) and [sound](#) with autoplay
- [Coordinates conversions](#)

Kiosk

- [Sample interactive kiosk](#)
- [xml document to get new issues](#)
- [Background download](#)
- [Ready in 3 minutes](#)
- Download pdf document or zipped archives (.fpk)
- Cover displayed
- Fully customizable

Text

- [Text search and result highlight](#)
- [List search results](#)
- [Look through results and go to the following one](#)
- [Text extraction](#)
- [Link between pages](#)
- [Support for link with urls](#)
- [Support for link to external documents](#)
- Search with [word selection](#) and results skimming
- [Zoom on results](#)
- [Multibyte characters supported](#)
- [Customizable search](#) and [profiling](#)

Miscellaneous

- True PDF
- [Free version](#) for low budget projects with complete features
- Fully multitouch
- Large document support
- [Retina Display](#) optimization
- [Customizable interface](#)
- Support for [encrypted documents](#)
- Works on every iOS version since 3.2
- Universal Binary
- [Destination named urls](#)

- [Get cropbox and rotation](#) for each page
- [Compatibility mode](#) for older devices
- [Three sample projects](#) with highly commented code
- [Ready to use interface](#) for immediate deploy with framework and library
- [Complete documentation](#)
- [View pdf in subview](#) or in standard [views stack](#)
- Every feature optimized for [iPhone](#)
- Every feature optimized for [iPad](#)
- Library and examples source code available on [github](#)
- Works with [Xcode](#) 3.2.6 and 4.x
- Open PDF documents [from file](#) and from [NSDataProvider](#)
- Smart [caching](#) pdf
- Open [multiple pdf documents at once](#)
- Complete [delegate feedbacks](#)
- Support for [UIGestureRecognizer](#)
- Multi-Threading is used to achieve maximum speed
- [Automatic and optimized caching system](#)

Opportunities

- Become a [Reseller](#)
- How [fits your needs](#)
- Free for selected [no-profit organizations](#)
- [100+ applications](#) already developed in [20 countries](#)
- Tens of [Partners](#)

Get latest library

- Pull from [github](#);
- Delete any previously installed application;
- Delete any intermediate build file;
- Perform a clean all from within Xcode;
- Empty the trash;
- Build you application without the logo.

How to add a UIToolbar

The library will always fill the view of the [MFDocumentViewController](#) subclass, that is [DocumentViewController](#) in the sample.

To have a toolbar at the bottom, you can either add you own toolbar as overlay (it will cover

the bottom of the pdf's view), and show/hide it as necessary, or as an alternative, present your [DocumentViewController](#) with a [UINavigationController](#) and use the [UINavigationController](#) default toolbar (it is hidden by default).

The latter will require you to manually set the toolbar items in `viewDidAppear:animated:`, since doing earlier will cause the toolbar to discard them too soon.

How to obtain a license

You are supposed to register at fastpdfkit.com and use **PayPal** to pay for the license, **AmEx** or perform a bank transfer. If you send us a payment receipt we will activate the license immediately.

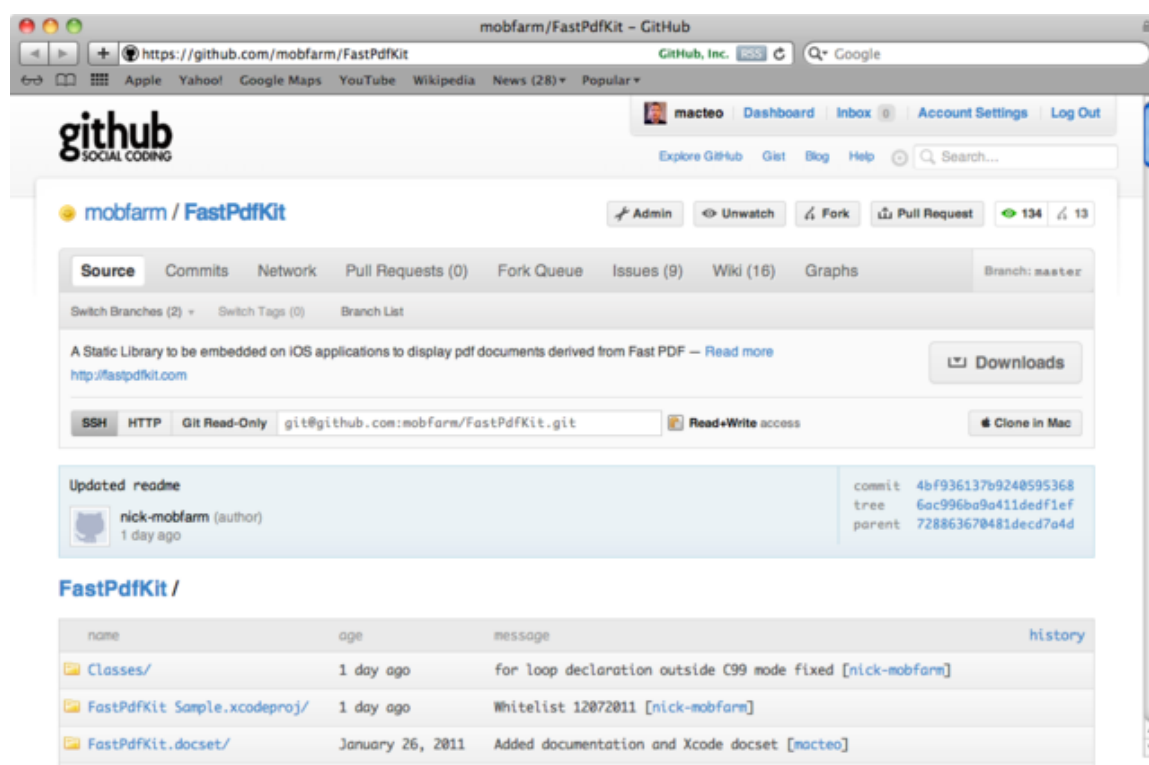
In few hours we will update the github repository with an activated library for your app's bundle-id (like `com.company.application`).

Then you just need to pull and grab the updated `libFastPdfKit.a` file and overwrite the old one on your project. Remember to empty the trash and perform a clean all from Xcode. Then you can safely build the activated app.

How-To clone from github

This tutorial is available as screencast on [YouTube](#). You should take a look!

1) Point your browser to github.com/mobfarm/fastpdfkit

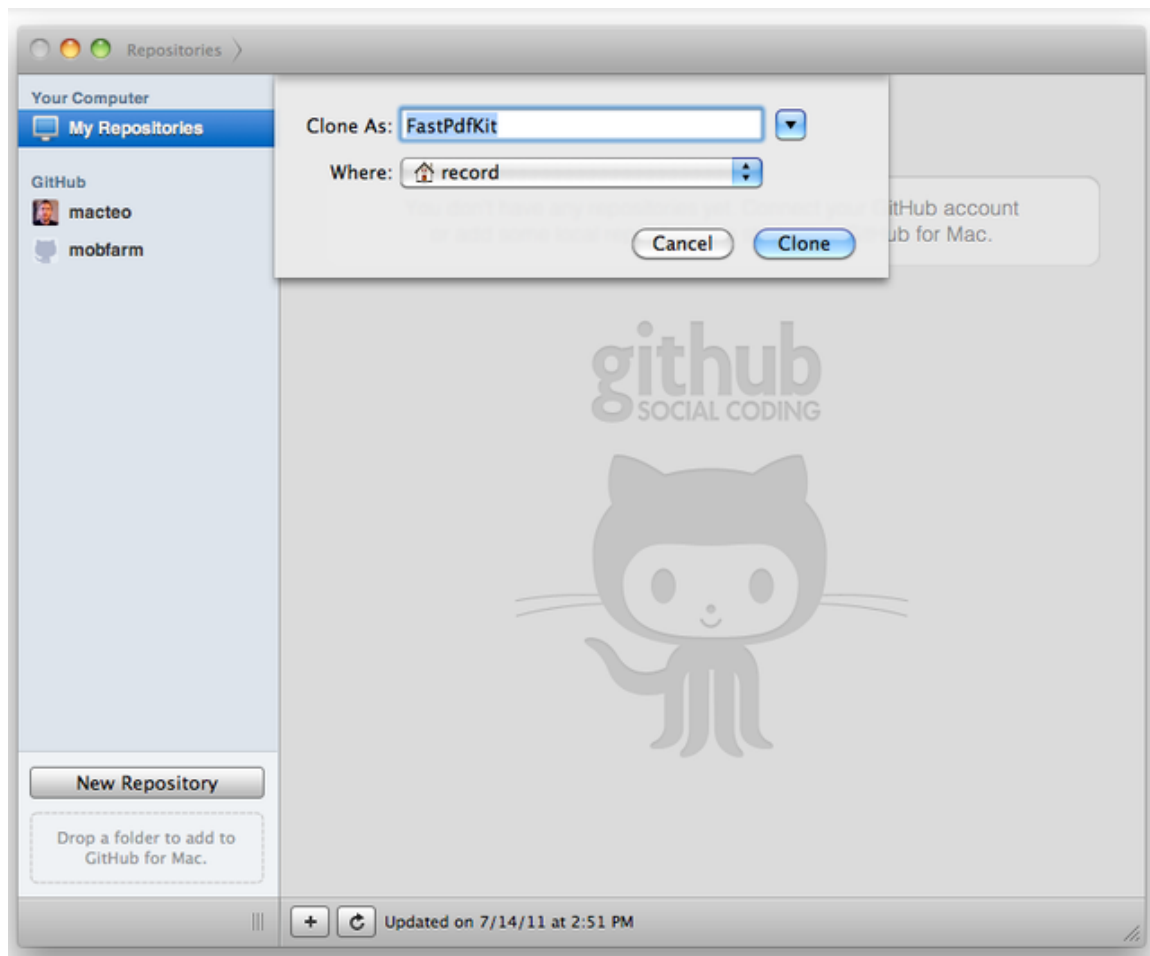


2) Login with your github account (if you don't have one you can create it for free).

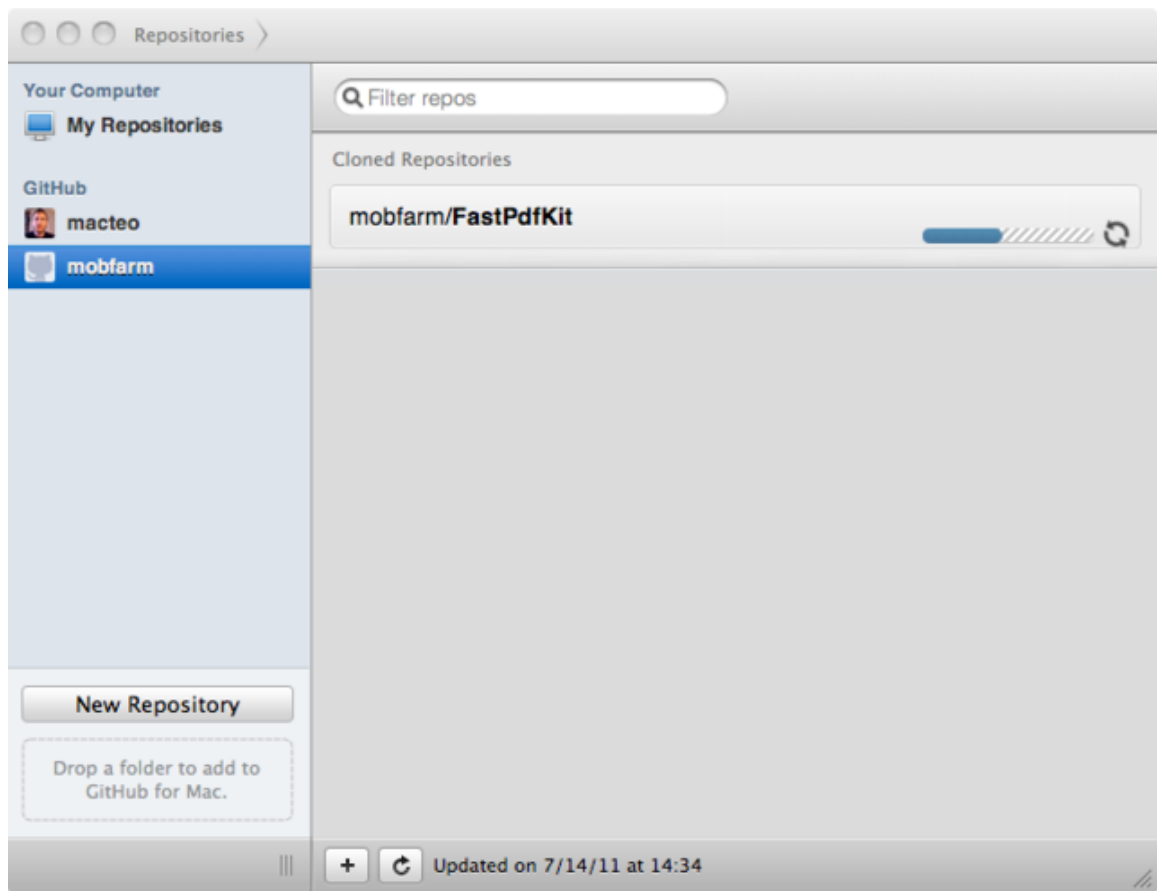
3) Click on **Clone in Mac** button, if you have the github mac app installed it will open,

otherwise you need to download it for free from mac.github.com

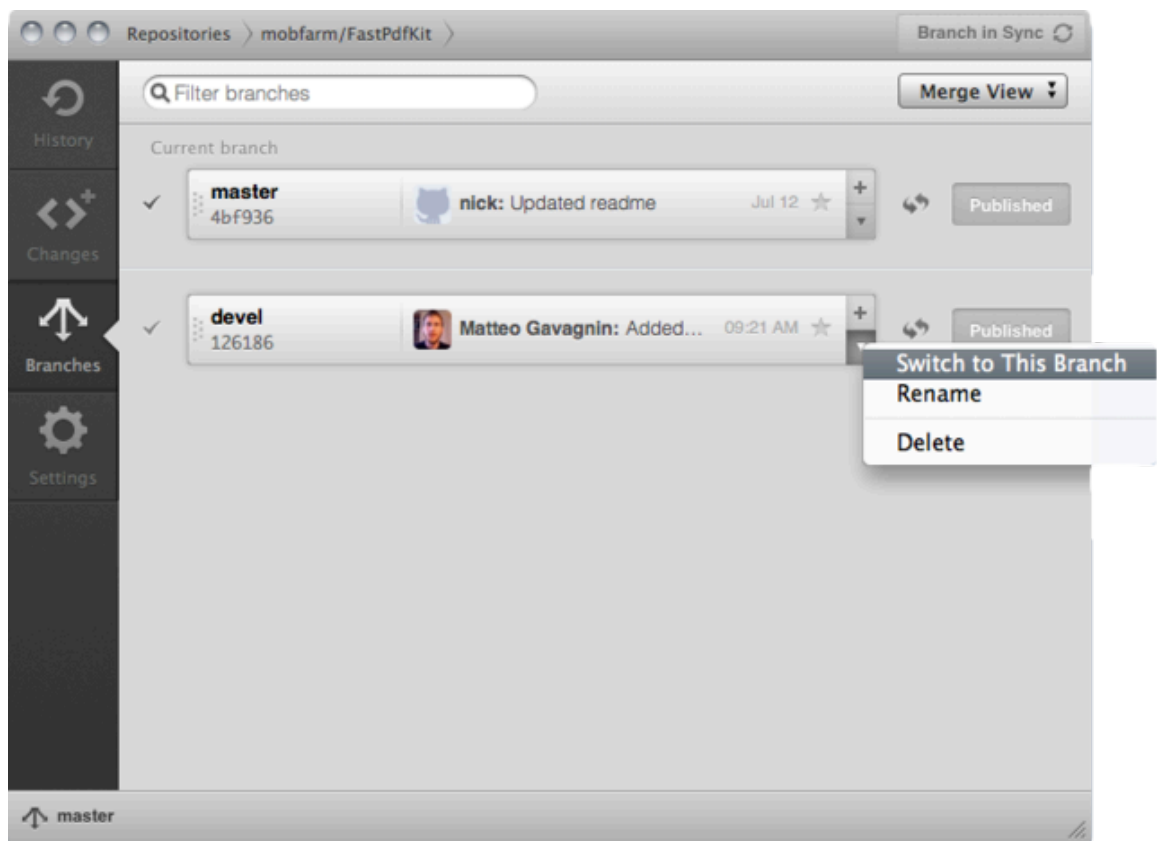
4) Choose a destination on your disk and wait for the cloning process



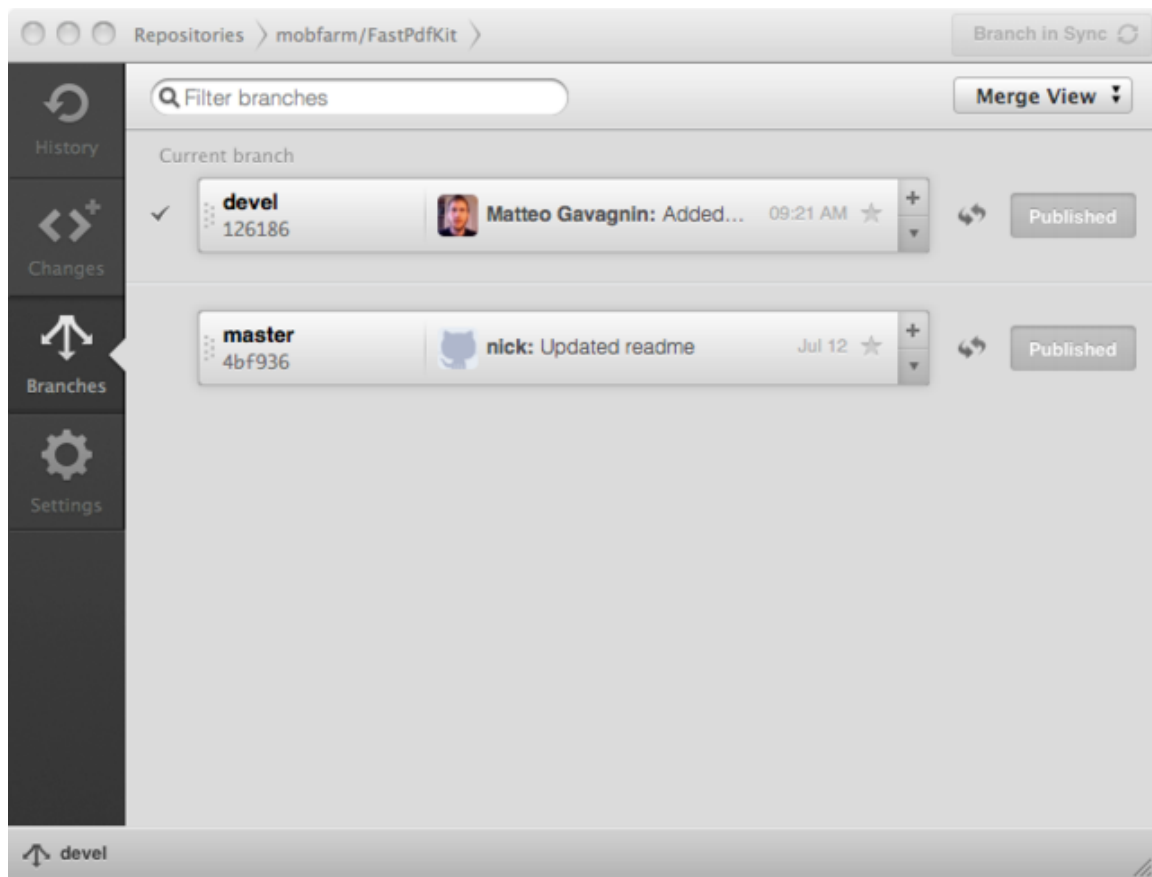
5) Wait for the clone to complete



6) If you need the latest development version switch to the devel branch (if you don't see it press first the *Sync Branch* button), otherwise you're done



7) Finish



How-To use FastPdfKit on existing projects (old version 1.X)

Add required files to existing project

- Download and extract the last [sample project](#);
- Open your existing app XCode project, open Project menu and choose Add to Project...
⌘⇧A, then locate FastPdfKit folder inside the downloaded package and click Add (this folder include the file [MFDocumentViewController.h](#), [libFastPDFKit.a](#) and many other file), be sure to check Copy items into destination group's folder (if needed);
- Right click on the Framework group and select Add and then Existing Framework..., then choose [QuartzCore.framework](#) from the list and press Add;
- With the same method add **all** these frameworks: [UIKit.framework](#), [CoreText.framework](#), [libz.1.2.3.dylib](#), [AVFoundation.framework](#), [MediaPlayer.framework](#), [CFNetwork.framework](#), [AudioToolbox.framework](#), [Foundation.framework](#) and [CoreGraphics.framework](#).
- At the end of this steps youu must have [this situation](#)
- Start coding
- Choose or add a new controller (we will call it [LauncherController](#)) to manage pdf documents and in the .h file add and add lines 3 and 7 to the controller

```
// LauncherController.h
#import <UIKit/UIKit.h>
@class MFDocumentManager;
@interface LauncherController : UIViewController {

}
-(IBAction)actionOpenPlainDocument:(id)sender;
@end
```

- Add this code before the @implementation line in the .m file

```
#import "MFDocumentManager.h"
#import "MFDocumentViewController.h"
#define DOC_PLAIN @"FastPdfKit-1.0RC1"
```

- Implement at least this method in the .m file

```
// LauncherController.m
-(IBAction)actionOpenPlainDocument:(id)sender {
    NSString *documentPath = [[NSBundle mainBundle]pathForResource:DOC_PLAIN
ofType:@"pdf"];
    NSURL *documentUrl = [NSURL fileURLWithPath:documentPath];
    MFDocumentManager *aDocManager = [[MFDocumentManager
alloc]initWithFileUrl:documentUrl];
    MFDocumentViewController *aDocViewController = [[MFDocumentViewController
alloc]initWithDocumentManager:aDocManager];
    [self presentViewController:aDocViewController animated:YES];
    [aDocViewController release];
}
```

- Now call the above action to open the pdf. You can find the code above with comments in the [BasicLauncherController](#) class.
- Within FastPdfKit folder there are many other sample controllers (in the Controllers group) where you can find methods (heavily commented) to manage every feature.

In development features

- Accessibility support;
- UIPageViewController display mode;
- Vertical scrolling.

Want to suggest a new feature? Start a new discussion in the [Suggestions](#) category.

Links between documents

FastPdfKit supports links between pdf documents and when you click one of them you get one of these two callbacks:

```
documentViewController:didReceiveRequestToGoToPage:ofFile:
```

```
documentViewController:didReceiveRequestToGoToDestinationNamed:ofFile:
```

With the first one you get directly the page number of the destination document, in the second case you need to call the `MFDocumentManager`'s `pageNumberForDestinationNamed:` method to get the number.

The Simple target provides some useful implementation of these callbacks. In the `MenuViewController` class there are two new methods to get the params from the `DocumentViewController`, that has the `MenuViewControllers` as delegate (set it even in the `actionOpenPlainDocument:` method).

`MenuViewController.m`

```
-(IBAction)actionOpenPlainDocument:(id)sender {
    // We are using NSBundle to lookup the file for us, but if you store the pdf
    // somewhere else than the application
    // bundle, you should use the NSFileManager instead or be able to provide the
    // right file path for the file.

    NSString *filePath = [[NSBundle mainBundle]pathForResource:DOC_PLAIN
ofType:@"pdf"];
    NSURL *documentUrl = [NSURL fileURLWithPath:filePath];

    //
    // Now that we have the URL, we can allocate an instance of the
    MFDocumentManager class (a wrapper) and use
    // it to initialize an MFDocumentViewController subclass
    MFDocumentManager *aDocManager = [[MFDocumentManager
alloc]initWithFileUrl:documentUrl];

    DocumentViewController *aDocViewController = [[DocumentViewController
alloc]initWithDocumentManager:aDocManager];
    [aDocViewController setDocumentId:DOC_PLAIN]; // We use the filename as an
ID. You can use whatever you like, like the id entry in a database or the hash of
the document.

    // This delegate has been added just to manage the links between pdfs, skip it
    // if you just need standard visualization
    [aDocViewController setDelegate:self];

    // In this example we use a navigation controller to present the document view
    // controller but you can present it
    // as a modal viewcontroller or just show a single PDF right from the
    // beginning
    [self presentViewController:aDocViewController animated:YES];

    [[self navigationController]pushViewController:aDocViewController
animated:YES];

    [aDocViewController release];
    [aDocManager release];
}

/* This method should be called from the DocumentViewController when you get a link
to another document */
```

```

-(void)setLinkedDocument:(NSString *)documentName withPage:
(NSUInteger)destinationPage orDestinationName:(NSString *)destinationName{
    NSArray *params = [NSArray arrayWithObjects:documentName,destinationName,
    [NSNumber numberWithInt:destinationPage], nil];
    [self performSelector:@selector(openDocumentWithParams:) withObject:params
    afterDelay:0.5];
}

/* This method opens a linked document after a delay to let you pop the controller
*/

-(void)openDocumentWithParams:(NSArray *)params{

    // Depending on the link format you need to manage the destination path
    accordingly to your own application path
    // In this example we are assuming that every document is placed in your
    application bundle at the same level
    NSString *filePath = [[NSBundle mainBundle]pathForResource:[params
    objectAtIndex:0] lastPathComponent] stringByDeletingPathExtension] ofType:@"pdf"];
    NSURL *documentUrl = [NSURL fileURLWithPath:filePath];

    MFDocumentManager *aDocManager = [[MFDocumentManager
    alloc] initWithFileUrl:documentUrl];

    DocumentViewController *aDocViewController = [[DocumentViewController
    alloc] initWithDocumentManager:aDocManager];
    [aDocViewController setDocumentId:[params objectAtIndex:0]];

    int page;
    if([params objectAtIndex:2] intValue] != -1){
        page = [params objectAtIndex:2] intValue];
    } else {
        // We need to parse the pdf to get the correct page
        page = [aDocManager pageNumberForDestinationNamed:[params
    objectAtIndex:1]];
    }

    [aDocViewController setPage:page];
    [aDocViewController setDocumentDelegate:aDocViewController];
    [aDocViewController setDelegate:self];

    [[self navigationController]pushViewController:aDocViewController
    animated:YES];

    [aDocViewController release];
    [aDocManager release];
}

```

In the `DocumentViewController.m` you just need to pass the parameters to the root controller and dismiss the current document.


```

- (void)documentViewController:(MFDocumentViewController *)dvc
didReceiveRequestToGoToDestinationNamed:(NSString *)destinationName ofFile:
(NSString *)fileName{

    // We set the parameters to the MenuViewController that will open the other
    document as soon as this one is dismissed

    [(MenuViewController *)delegate setLinkedDocument:fileName withPage:-1
orDestinationName:destinationName];

    // We need to dismiss the document
    [self actionDismiss:nil];
}
- (void)documentViewController:(MFDocumentViewController *)dvc
didReceiveRequestToGoToPage:(NSUInteger)pageNumber ofFile:(NSString *)fileName{

    // We set the parameters to the MenuViewController that will open the other
    document as soon as this one is dismissed

    [(MenuViewController *)delegate setLinkedDocument:fileName withPage:pageNumber
orDestinationName:@""];

    // We need to dismiss the document
    [self actionDismiss:nil];
}

```

Multimedia Protocols and uri

In the 2.0 version (devel branch currently) we've added multimedia support in overlay.

To add these overlay you need to add some annotations to the pdf with custom url destinations.

For each media overlay you can add remote or local files and they can be displayed in overlay on the page, or as modal view. You can also define your own uri and then design your custom views.

For local documents you need to specify relative paths to the pdf document.

For example if you put a local html page called *index.html* in a folder named *page* at the same level of the pdf and need to display it in overlay: *fpkh://page/index.html*.

Protocols:

| Type | Local | Remote |
|---------------|---------|---------|
| Overlay Movie | fpkv:// | fpky:// |
| Modal Movie | fpke:// | fpkz:// |
| Overlay HTML | fpkh:// | fpkw:// |
| Modal HTML | fpki:// | http:// |

| | | |
|---------------|---------|---------|
| Overlay Audio | fpka:// | fpkb:// |
|---------------|---------|---------|

Overlay: Drawables and Touchables

FastPdfKit supports custom overlays over pdf pages. Starting from version 2 some of them are rendered automatically like movies and web pages due to support for custom [uri annotations](#).

Here we give you some suggestions on how-to add support for your own overlays. First of all there are three different overlays: [MFOverlayDrawable](#) (that don't have any user interaction), [MFOverlayTouchable](#) (that give you a feedback when users tap on them) and views (real [UIView](#)). In this article we are going to describe the first two.

First of all you need an objects that implements the [MFDocumentOverlayDataSource](#) protocol: particularly [drawablesForPage:](#) and [touchablesForPage:](#) methods, returning an array filled with [MFOverlayDrawable](#)s and [MFOverlayTouchable](#)s.

Obviously you need to create your own [MFOverlayDrawable](#) and [MFOverlayTouchable](#) subclasses to provide the right overlays. It's very simple as you need to add your [drawInContext:](#) method.

You can call [reloadOverlay](#) on your [MFDocumentViewController](#) to force the redraw or disable any overlay with the [overlayEnabled](#) property. If you need that the overlays is maintained sharp even when zoomed, simply set to YES the [useTiledOverlayView](#)

A sample of this flow is included in the [devel branch](#) and the *Simple* target.

In the [MenuViewController](#) we instantiate the [OverlayManager](#) that is a sample object that implements the [MFDocumentOverlayDataSource](#) protocol in the [actionOpenPlainDocument](#) method.

```
OverlayManager *ovManager = [[OverlayManager alloc] init];
[aDocViewController addOverlayDataSource:ovManager];
[ovManager release];
```

The [OverlayManager](#) is composed of just one method as we need only one [MFOverlayDrawable](#) in the first page.

```
- (NSArray *)documentViewController:(MFDocumentViewController *)dvc
drawablesForPage:(NSUInteger)page{
    NSArray *array;
    if(page == 1){
        array = [NSArray arrayWithObject:[Drawable alloc] init];
    } else
        array = [[NSArray alloc] init];
    return array;
}
```

In the [Drawable](#) class we add an image in the lower left corner and two lines of text near it.

```
-(void)drawInContext:(CGContextRef)context{
    UIImage *image = [UIImage imageNamed:@"Icon.png"];
    CGRect rect = CGRectMake(20, 80, 61, 61);

    CGContextSetGrayFillColor(context, 0.0, 1.0);
    CGContextSelectFont(context, "Courier", 12, kCGEncodingMacRoman);
    CGContextSetTextDrawingMode(context, kCGTextFill);
    CGContextSetShouldAntialias(context, true);
    char* text = "This is an overlay, check classes";
    CGContextShowTextAtPoint(context, 101, 113, text, strlen(text));
    text = "OverlayManager and Drawable";
    CGContextShowTextAtPoint(context, 101, 100, text, strlen(text));
    CGContextDrawImage(context, rect, image.CGImage);
}
```

Enjoy your overlays!

As you probably know you can support custom annotations, directly adding url annotations on your pdf document. Then implement the `documentViewController:didReceiveURLRequest:` or one of the other delegate methods to display custom overlays or views directly in page coordinates.

Overview and Features

Overview

FastPdfKit is a Static Library that can be embedded in your iOS projects to support pdf documents. It has been developed to be easy to implement for developers, fast and responsive for users, and compatible with every iOS device.

Features

- Fast PDF rendering with side sliding;
- Fully customizable interface;
- Internal link support;
- Search with highlighted results;
- Text extraction;
- Legacy or Speedy mode;
- Embedded PDF thumbnails support;
- Page thumbnail creation;
- Page preloading;
- Large document support;
- Single, double or auto page modes;
- Autorotation;
- Double tap and pinch to zoom;
- Landscape and Portrait support;

- Tap on a side to go forward or backward;
 - Zoom Lock;
 - Auto Zoom;
 - Brightness control;
 - Slider to change page;
 - Bookmarks;
 - Support for password protected documents;
 - Outline – TOC;
 - Full screen view;
 - Partial screen view;
 - Retina display support;
 - Support every iOS version starting from 3.1;
 - Compatible with every iPad, iPhone and iPod touch.
-

PDF as Subview

In the master branch we've you can find a simple project called *ResizingSample* that adds the [MFDocumentViewController](#)'s view as a subview of another controller's view.

Messages like `-viewWillAppear` and `viewDidAppear` have to be forwarded manually to the [MFDocumentViewController](#) when it is used outside of its intended scope (as element of a navigation stack or as a modal view controller).

Just press the *Move!* button to see what happens. The [MoveViewController](#) class is commented in the critical parts.

To place your order you can choose between three payment methods:

PayPal

With your credit card or other funding sources using our [online store](#).

Bank Transfer

Be sure to contact us and request an order number or and invoice before proceeding with the bank transfer. If you send us your order receipt we will be able to activate the license in few hours instead of few days.

- **IBAN** IT87Y0572862180234570631265
- **SWIFT** BPVIT22
- **Owner** MobFarm

AmEx

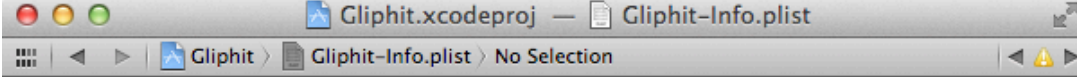
Just insert the card number and owner information in the form in [online store](#).

How-To get the activated license.

As soon as you buy the license we will activate it from monday to friday the same day at 6 PM GMT+1.

Then you just need to add in your application **Info.plist** file a new key called `FPKLicenseKey` of String type. As value put the activation code.

The result will be like the figure below.



| Key | Type | Value |
|---|------------|--|
| Bundle name | String | \$(PRODUCT_NAME) |
| FPKLicenseKey | String | 5315ff713f5202a1e87c4665e2d64c90a422df26 |
| Localization native development region | String | en |
| Bundle version | String | 1.0.0.167 |
| Required background modes | Array | (1 item) |
| Status bar is initially hidden | Boolean | YES |
| Bundle OS Type code | String | APPL |
| Icon file | String | |
| Bundle versions string, short | String | 0.9 |
| InfoDictionary version | String | 6.0 |
| Required device capabilities | Array | (1 item) |
| Executable file | String | \$(EXECUTABLE_NAME) |
| Supported interface orientations (iPad) | Array | (4 items) |
| Bundle identifier | String | com.mobfarm.fastpdfkit |
| Bundle creator OS Type code | String | ??? |
| Icon files | Array | (3 items) |
| Icon files (iOS 5) | Diction... | (2 items) |
| Application requires iPhone environmei | Boolean | YES |
| Bundle display name | String | Gliphit |
| Supported interface orientations | Array | (4 items) |
| Application presents content in Newsst | Boolean | NO |

If you lose the mail message your code will be always available at [fastpdfkit.com/# Release Notes](http://fastpdfkit.com/#ReleaseNotes)

Update 3.0 (November 4th, 2011)

- Internal changes of the view hierarchy to provide more flexibility and overlay transitions.
- Revamped `ReaderViewController` user interface.
- Better thumbnail scroll view and generation.
- Added support to video overlay control parameters in the uri.
- Search now works for terms of unitary length.
- Added a method to retrieve annotations from the document and provide them as overlays.
- Slightly changes to `MFDocumentViewControllerDelegate` callbacks.
- New activation method with key `FPKLicenseKey` from *Info.plist*: no need to pull to get the activated version.

Update 2.1.3 (September 27th, 2011)

Disabled the 2.1.1 font cache due to an implementation oversight. Will be re-introduced as soon as it will behave as expected.

Update 2.1.2 (September 21th, 2011)

- Cleaned up text search and extraction a little bit. More fixes incoming.

Update 2.1.1 (September 15th, 2011)

- Added a font cache system for text extraction and search, which should give sensible improvement in search speed especially on documents with a large amount of different fonts.

Update 2.1.0 (September 14th, 2011)

- Added Overflow page mode to previous Single and Double. Basically, the pdf page will now fill the screen along its width, overflowing under the bottom of the screen if necessary.
- Added `autoMode` property. It will tell the `MFDocumentViewController` to what mode switch when in landscape if `automodeOnRotation` is YES. Default is `MFAutoModeDouble`, other options are `MFAutoModeSingle` and `MFAutoModeOverflow`.

Update 2.0.3 (August 10th, 2011)

- Fixed a bug in the transformation returned on double page mode for page with an angle not equal to 0.
- Added guard to iOS 4.x only methods.

Update 2.0.2 (August 09th, 2011)

- Added support to link annotation with Remote Go-To actions.
- Updated manual with latest methods.
- Added methods to convert points and rect to and from different coordinate systems. Take a look at the `MFDocumentViewController` for details.
- Documented the method to get the cropbox and rotation angle for each document page.
- Finally fixed the bad behavior of the detail (tiled) view on retina device.
- Fixed a bug involving rendering of the preview pages at low res on retina display introduced a few updates ago.
- The `-didGoToPage` callback is now called once when a page is changed on user scroll input.

Update 2.0.1 (July 21th, 2011)

- Bleeding of the pdf cover images fixed.
- The embedded UIWebView is now embedded a bit better.

Update 2.0.0-devel (July 12th, 2011)

- Multimedia support
- Reorganized project
- Many other improvements

Update: 1.0.19 (Sep 5th, 2011)

- Added a temp fix to address search and extraction of text whose fonts metrics is saved only in the embedded font descriptor. An actual fix would require to develop a font descriptor parser.

Update: 1.0.18 (Sep 2nd, 2011)

- Hardcoded the CID font files inside the library. No need to keep them in the bundle. They will be physically removed from the repo at later date.

Update: 1.0.17 (Aug 30th, 2011)

- Small fix to space handling in both text search and extraction.

Update: 1.0.16 (Aug 10th, 2011)

- Added check to avoid a few 4.0 methods getting called on earlier iOS versions.
- Fixed wrong transformation on double page mode for page with angle different from 0.

Update: 1.0.15 (Aug 09th, 2011)

- Added support to link annotation with Remote Go-To actions.
- Updated manual with latest methods.

Update: 1.0.14 (Aug 05th, 2011)

- Added methods to convert points and rect to and from different coordinate systems. Take a look at the [MFDocumentViewController](#) for details.
- Documented the method to get the cropbox and rotation angle for each document page.

Update: 1.0.13 (Jul 26th, 2011)

- Finally fixed the bad behavior of the detail (tiled) view on retina device.
- Fixed a bug involving rendering of the preview pages at low res on retina display introduced a few updates ago.
- The [-didGoToPage](#) callback is now called once when a page is changed on user scroll input.

Update: 1.0.12 (Jul 21th, 2011)

- Fixed the bleeding of the pdf, usually on the front and back covers, introduced with the previous fixes.

Update: 1.0.11 (Jul 18th, 2011)

- Fixed a bug where disabling the shadow, the page resulted with a transparent backing
- Fixed a rare occurrence of a crash while searching due to array boundaries miscalculation

Update: 1.0.10 (Jul 15th, 2011)

- Added a sample application (FastPdfKit Resizing Sample) to illustrate how to handle the

situation where the `MFDocumentViewController`'s view is added as a subview of another view. The `MFDocumentViewController` has not been created to used that way, but there's a simple workaround to getting things work in most cases.

Update: 1.0.9 (Jun 22th, 2011)

- Added directional lock to the page scroll view.
- Changed popover behavior in the `DocumentViewControllerKiosk`. This should fix crash when reopening a document when dismissed with an open popover.
- Fixed a few more leaks.

1.0.8 (Jun 10th, 2011)

- Thumbnails are laid out correctly upon rotation.

1.0.7 (Jun 07th, 2011)

- Fixed a nasty bug in the parser of TrueType font. CMap parser redone on the ground up to be formatting agnostic. Most of the search/extraction related crash should be fixed now. Remember to try the `test_` versions of search and extraction function.

1.0.6 (May 27th, 2011)

- Added an optional tiled version of the overlay view. If you want sharp drawables when zoomed in, set `MFDocumentViewController`'s `useTiledOverlayView` to YES. Keep in mind that tiled layer rendering is slower, and memory usage is higher.
- Dropped search view controller and mini search view local copy of search manager delegate's results. They now directly access the data source results. Crash caused by inconsistency between the local copy and the data source data should be fixed.
- Replaced inner rendering parameters data class with a simpler struct together with a better synchronization between threads. Crash on CALayer status error with NaN origin should be fixed.

1.0.5 (May 19th, 2011)

- Added two alternative methods in `MFDocumentManager` for text search and extraction. The methods are `(void)test_searchResultOnPage:(NSUInteger)page forSearchTerms:(NSString *)searchTerms` and `(void)test_wholeTextForPage:(NSUInteger)page`. They return the same results of the non `_test` versions. To use them, replace the occurrence of the older methods in the project, as exemplified in comments of `TextSearchOperation`'s `main()` method and `TextDisplayViewController`'s `selectorWholeTextForPage:` method.

1.0.4 (May 3rd, 2011)

- Fixed the floating page issue, when the page is changed when zoomed in.

1.0.3 (May 2nd, 2011)

- Better handling of the device orientation at startup.

1.0.2 (Apr 29th, 2011)

- Fixed a bug where right drawables were not displayed.
- Zoom animation for `setPage:withZoomLevel:onRect:` is now correct. Moreover, passing 0 as the level of zoom will let the application try to fit the rect on screen.
- Fixed a crash when an annotation with an uri shorter than 7 char is found.
- Added `leftPageNumber` and `rightPageNumber` variables to the `MFDocumentViewController`.
- Fixed the `autoMode` on rotation not being considered at startup.

1.0.1 (Apr 27th, 2011)

- Replaced `URLForResource` with `pathForResources` for 3.X compatibility;
- Added `(float)zoomScale` and `(CGPoint)zoomOffset` methods to `MFDocumentViewController` to get zoom position;
- Added support for `CGPDFDocumentCreateWithProvider` with method `initWithDataProvider:(CGDataProviderRef)provider;`
- Option to remove shadow and render the page fullscreen on `MFDocumentViewController` using `(float)padding` and `(BOOL)showShadow` methods;
- Fixed another crash with search results.

1.0 (Apr 19th, 2011)

- Fixed ignored optional flag for the `didChangeMode:` callback.
- Added `didReceiveTapOnAnnotationRect:with: onPage:` method. This replaces `didReceiveURLRequest:`, but the latter is still called.

1.0RC2 (Apr 6th, 2011)

- Fixed a crash when the searched string will not fit in the mini search view. Bookmarks not being saved when the popover is dismissed by clicking outside fixed. Double tap to zoom out will now work even when the zoom in has been performed manually. Added a callback to ask the `DocumentViewController` delegate if a video will have to autoplay or not. Added page parameter to the double tap annotation callback. Removed a few unneeded logs and minor tweaks.

1.0RC1 (Mar 8th, 2011)

- Kiosk application target added. Kiosk is a demo application with a customizable list of document to choose from. Viewer is enhanced with a scrollable list of page thumbnail and nicer interface.

0.9.5 (Feb 14th, 2011)

- Early support for type 0 fonts for search and text extraction
- Fix on bookmarks controller buttons
- Safer cleanup implementation

0.9.1 (Feb 3rd, 2011)

- Added customizable Td, TD, Tm, T* and TJ behaviour with custom profiles. Look at

[mprofile.h](#) and [MFDocumentManager.h](#)

- Added CMap support for non Type 0 fonts
- Added FastPdfKit+ whitelist
- Fixed SearchTableView dequeue bug
- Added documentation and XCode docset
- Local into the doc folder
- Remote at doc.mobfarm.eu/fastpdfkit
- XCode docset feed at fastpdfkit.com/docset/docset.atom
- Solved some memory leaks

0.9.0 (Dec 12th, 2010)

- Bundle-id protection
- Customizable interface
- Added splash image
- Results table selected words highlighted
- Small view for rapid results scrubbing
- Zoom on the found word
- Fixed first letter highlight bug
- Supported encoding for every non multibyte font

0.7.1 (Dec 3rd, 2010)

- External links support

0.7

- Page thumbnail creation;
- Fixed single tap on screen bug.

0.6

- Internal link support;
- Added legacy mode;
- Removed side [UIButtons](#) and added multitouch areas;
- Support for embedded pdf thumbnails.

0.5

- Search with highlight;
- Text extraction;
- Added more supported encoding.

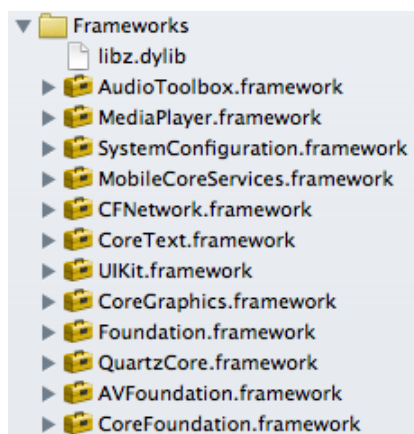
0.4

- Fast PDF rendering with side sliding;

- Page preloading;
 - Large document support;
 - Single, double or auto page modes;
 - Autorotation;
 - Customizable interface;
 - Double tap and pinch to zoom;
 - Landscape and Portrait support;
 - Tap on a side to go forward or backward;
 - Zoom Lock;
 - Auto Zoom;
 - Brightness control;
 - Slider to change page;
 - Bookmarks;
 - Support for password protected documents;
 - Outline - TOC;
 - Full screen view;
 - Partial screen view;
 - Retina display support;
 - Support every iOS version starting from 3.1;
 - Compatible with every iPad, iPhone and iPod touch.
-

Required Frameworks

To compile FastPdfKit you need to link against these frameworks: [AudioToolbox](#), [MediaPlayer](#), [SystemConfiguration](#), [MobileCoreServices](#), [CFNetwork](#), [CoreText](#), [UIKit](#), [CoreGraphics](#), [Foundation](#), [QuartzCore](#), [AVFoundation](#), [CoreFoundation](#) and the `z` library.



Xcode docset

Add this [Xcode DocSet link](#) to your Xcode Organizer.



FastPdfKit Reader

What is it?

FastPdfKit Reader is an application available on the App Store that let you do three things:

- Get the latest guides and informations regarding **FastPdfKit**;
- Easily try FastPdfKit in a real app with some pdf documents;
- Get pdf from any source with standard rss feed.

FastPdfKit



FastPdfKit is a library that let you show pdf documents in iOS applications bypassing all performances and missing features problems related to QuickLook.

Side scrolling, search with highlighted results, preview and thumbnails, text extraction, overlay views, embedded multimedia, optimization for every device, single and double page are just some of countless features included.

Get more informations at fastpdfkit.com.

Get the guides and informations regarding FastPdfKit

In FastPdfKit Reader the xml documents are linked to a feed that contains every pdf document regarding FastPdfKit.

- Knowledge Base: a comprehensive document with guides and tips get the best from FastPdfKit.
- Complete documentation: classes and methods that you could use in your own implementations.
- License: the license agreement.
- Reseller agreement: become a Reseller and start selling applications in a breeze.
- Sources: a source list of pdf documents to choose from.
- Gliphit: a demonstration pdf document with a multimedia enriched pdf.

Easily try FastPdfKit in a real app with some pdf documents

You have now a new way to try FastPdfKit and its performances. FastPdfKit Reader includes the latest 3.0 library with the new rendering engine and multimedia support. If you are satisfied you can simply clone from [github](#) the whole repository and use it in your applications.

Another option is to buy the latest [FastPdf+](#) version from the App Store and have a complete document reader with FastPdfKit as engine.

Get pdf from any source with standard rss feed

This is the most useful feature for common people. With a 0.99\$ In-App Purchase you can unlock the rss feed source and add your custom address.

In the **Sources** pdf you can find some ready to use feeds that can replace this one. Take a look!

If you've created a feed write a line at [support.fastpdfkit.com](#) and let us add it as official source.

List of noteworthy feeds

We are reporting some useful links of public pdf documents that you can insert in the rss feed field.

Juice an italian Mac Magazine realized by APstore and Macitynet

<http://reader.fastpdfkit.com/juice.xml>

20 Project Alexander Dellal's 20 Projects, formerly 20 Hoxton Square Projects operates as a platform for emerging contemporary artists, whilst also acts as a creative hub for independent and collaborative projects.

<http://reader.fastpdfkit.com/20projects.xml>

Unless you will

<http://reader.fastpdfkit.com/unlessyouwill.xml>

Project Gutenberg

<http://reader.fastpdfkit.com/projectgutenberg.xml>

United Spinal

<http://reader.fastpdfkit.com/unitedspinal.xml>

How to generate a custom rss feed

The rss is a standard xml document that can be created manually or with common tools, but mainly with blog and news platforms like [Wordpress](#).

The rss must respect these guidelines:

- You need to include the xml header like `<?xml version="1.0" encoding="UTF-8"?>`;
- The root element must be of rss type like `<rss version="2.0">`;
- The second level tag must be `<channel>`;
- Each element should be included in tags `<item></item>`;
- The pdf title is in the `<title>` tag;

- The pdf link should be in the `<enclosure url="" length="" type="application/pdf" />` tag;
- The pdf cover image should be in the `<enclosure url="" length="" type="image/jpg" />` tag or with `image/png`.

Below there is an example

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <item>
      <title>Item 1</title>
      <enclosure url="http://fastpdfkit.com/license.jpg" length="25345"
type="image/jpg" />
      <enclosure url="http://fastpdfkit.com/license.pdf" length="225850"
type="application/pdf" />
    </item>
  </channel>
</rss>
```

Use WordPress to generate rss feeds for pdf documents

First of all you need a WordPress instance, your own or a hosted one. Take a look at wordpress.org for more informations.

Then go to the administration page and create a new blog post.

In the **content** you need to specify at least two things:

- The cover address ``;
- The pdf link `License`;

The add two custom fields with name `enclosure` and value:

```
http://reader.fastpdfkit.com/license.png
225850
image/png
```

for the cover.

```
http://reader.fastpdfkit.com/license.pdf
1345452
application/pdf
```

for the pdf.

The first row of the custom field is clearly the address, the second the size in Bytes and the third the mime type. For the images you can use png or jpeg images and the corresponding `image/png` and `image/jpg` mime types.

Then publish the post.

Go to your main blog page and get the rss feed. You can even assign the posts that contains a pdf to a custom category and point FastPdfKit Reader to its rss feed.

t

