Printing on iOS 4.2

Yutaka Yasuda, Kyoto Sangyo University

まずは資料探し・下調べ

AirPrint

- iPhone / iPad から印刷
- アプリ側に要組込
- Safari など対応済み
- iOS 4.2 から
- 対応プリンタ必要
  - 2011年3月では HP の ePrint 対応モデル幾つかのみ

ASCII.jp x MacPeople

- OS XのプリンターをAirPrint経由で使う -- Apple Geeks 第19回
  http://ascii.jp/elem/000/000/575/575397/

- そこへ1件の情報が届いた。「内容がたった1行のファイルをOS XのCUPS管理領域へコピーするだけで、OS Xにローカル接続されたプリンターをAirPrint対応にできる」というものだ。

- CUPSらしい

呪文？

```
$ echo 'image/urf urf (0,UNIRAST<00>)' > airprint.types
$ sudo cp airprint.types /usr/share/cups/mime/
Password:
$ sudo killall cupsd
```

資料さがし

- 日本語の簡単そうなドキュメントはパッとは見つからず
- iOS Reference Library の Drawing and Printing Guide for iOS から始める
  - 日本語版 “iOS描画および印刷ガイト”
  - ここでは全体のコンセプトを説明
  - Printing はかなり大きな扱い
  - 読むのは Printing とUIPrint* クラスで足りそう
  - UIKit の一部

http://developer.apple.com/library/ios/#documentation/2DDrawing/Conceptual/DrawingPrintingiOS/
http://developer.apple.com/jp/devcenter/ios/library/documentation/DrawingPrintingiOS.pdf

Drawing and Printing Guide for iOS を読む

作業当時日本語版の存在を知らなかったので英語版を追っています

http://developer.apple.com/library/ios/#documentation/2DDrawing/Conceptual/DrawingPrintingiOS/

印刷方法

- UIKit でコンテンツをプリントする方法は三通りある
- NSData, NSURL, UIImage, ALAsset オブジェクトを直接プリントできるフレームワークあり。イメージや PDF データか、その参照を与える。

  「第一の方法 (#1)」と呼ぶ
- print formatter を print job にアサインする。Text や HTML のようなコンテンツを複数ページにレイアウトする。

  「第二の方法 (#2)」と呼ぶ
- ページレンダラー（UIPrintPageRenderer のカスタムサブクラスのインスタンス）を print job にアサインする。（複雑な図形用かな？）

  「第三の方法 (#3)」と呼ぶ

http://developer.apple.com/library/ios/#documentation/2DDrawing/Conceptual/DrawingPrintingiOS/

Printing

- これがハイライトか
  - ユーザインタフェイス（操作法）説明
  - Printing アーキテクチャを概観
  - UIKit の Printing API
  - Printing Class と Protocol 概説
  - 印刷方法・ワークフロー
  - サンプルコード数種

http://developer.apple.com/library/ios/#documentation/2DDrawing/Conceptual/DrawingPrintingiOS/Printing/Printing.html

How Printing Works in iOS

- UIKit は PDF データとしてアプリがドローしたものを使う
- 紙切れなどのエラーは Print Center が処理する
- Figure 6-6 Printing architecture を参照

http://developer.apple.com/library/ios/#documentation/2DDrawing/Conceptual/DrawingPrintingiOS/Printing/Printing.html

UIPrintInteractionController

- UIPrintInteractionController
  - これが主役
  - 必要な情報を全て保持
- 「行ってこい」方式
  - 各種情報・データをセットして
  - システムに預ける
  - 以降の制御をユーザが書くことはない

「行ってこい」までの流れ

世の始まりにコントローラあり

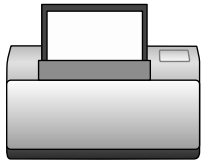
UIPrintInteractionController

インスタンス生成

印刷処理
総代理システム

まるっと引き渡し

どないかして
印刷



pic

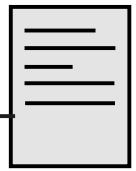
printInfo

•printingItem
•printingItems
•printFormatter
•printPageRenderer

printinfo

outputType
duplex ...

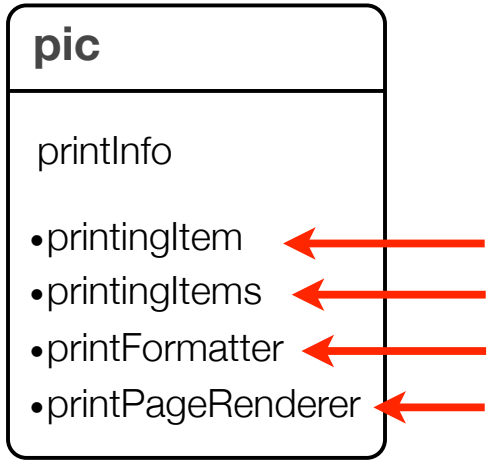
各種設定情報



text,
image, 印刷したいデータ
PDF

エラーハンドラ

４種類の出力支援機能



- 四つのプロパティ
  - printingItem - image あるいは PDF 用 **(#1)**
  - printingItems - printingItem の Array **(#1')**
  - printFormatter - UIPrintFormatter 対応のviewとtext, HTML 用 **(#2)**
  - printPageRenderer - UIPrintPageRenderer 用 **(#3)**
- どれか一つにコンテンツをセットして「行ってこい」で、
  それにふさわしい方法でレイアウトを支援する

Print Formatters

第二の方法 (#2)

- UIPrintFormatter をベースにコンテンツのレイアウトを支援する
  - ヘッダ・フッタなどもサポート
- UISimpleTextPrintFormatter - plain text 用
  （font, color, align, 改行など指定可能）
- UIMarkupTextPrintFormatter - HTML 用
- UIViewPrintFormatter
  - UIView にこれをサポートするインスタンスメソッド二つ追加
    – viewPrintFormatter, – drawRect:forViewPrintFormatter
  - viewPrintFormatter で print formatter を取得して操作
  - 今のところ UIWebView, UITextView, MKMapView が対応

もっと自由を！

- Page Renderers　第三の方法 (#3)
  - より複雑な（あるいはアプリ介入が必要な）印刷用
  - 紙の全領域を自分ですべてコントロールする
  - Renderer の中で局所的に UIPrintFormatter を使える

- UPPrintInteractionControllerDelegate
  - アプリ固有の振る舞いをフックルーチンとして実装
    （呼出は print option 表示の前後、print job 開始と完了時）

UIPrintInfo

- 各種印刷設定項目をプロパティとして保持
  - orientation ：紙の向き
  - duplex ：片面・両面
  - outputType ：紙サイズ
    - UIPrintInfoOutputPhoto - A6 (or 4x6, depends on locale)
    - UIPrintInfoOutputGeneral - A4 or US Letter (on locale too)

実行する機体のロケールに注意が必要

locale 設定を確認（実機で！）

```
// 現在のロケールを表示
NSString *localeIdentifier = [[NSLocale currentLocale] localeIdentifier];
NSString *localeString = [[NSLocale currentLocale]
          displayNameForKey:NSLocaleIdentifier value:localeIdentifier];
NSString *message = [NSString stringWithFormat:@"id=%@ : %@",
                             localeIdentifier, localeString];
UIAlertView* alert = [[UIAlertView alloc]
                      initWithTitle:@"language"
                      message:message
                      delegate:self
                      cancelButtonTitle:@"OK"
                      otherButtonTitles:nil];
[alert show];
[alert release];
```

実行結果：

Sample Code : 二つだけ

- PrintWebView **#2の例**
  - printFormatter に viewPrintFormatter を与えて
  - webView を整形
  - header, footer を出す
  - webView 処理のサンプルにもなる
- PrintPhoto **#3の例**
  - UIPrintPageRenderer のサブクラスを用意し
  - これを printFormatterにレンダラとして与え
  - drawPageAtIndex: でページ全域に自由に描画
  - 写真を紙全面に割り付ける
  - header, footer はなし
  - イメージ処理のサンプルになるかも

http://developer.apple.com/library/ios/#samplecode/PrintWebView/
http://developer.apple.com/library/ios/#samplecode/PrintPhoto/

作成したテストプログラム TestPrint

(#1)

この段は printingItem のテスト

(#2)

ここは formatter のテスト

(#3)

この段は renderer のテスト

ボタン隅に A6 とあるものは写真用紙へプリントする

(#2)

ここは WebView の出力テスト先に load ボタンをタップして読み込んでから print ボタンで印刷させる

(#3)

multi ボタンをタップすると左の選択セグメントで指定した枚数でタイル状に写真を並べたものを出力する

そろそろコードの話をしよう

準備： AirPrint 対応を確認する

- isPrintingAvailable

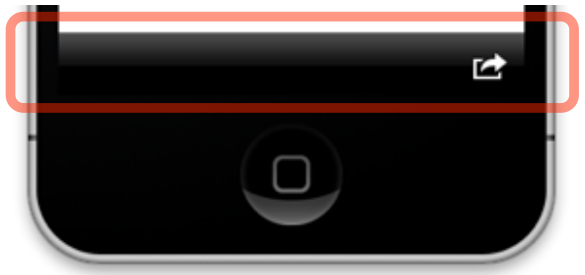
```
- (void)viewDidLoad {
    if (![UIPrintInteractionController isPrintingAvailable])
        [myPrintButton removeFromSuperView];
    // other tasks...
}
```

推奨、というか恐らく必須

- PrintWebView サンプルに利用例あり

  - `(void)setupToolbarItems` を参照

  ツールバーも内部で作っていて参考になる



こういうバー

準備：PrintInfoの設定

```
// コントローラの取得
UIPrintInteractionController *controller =
        [UIPrintInteractionController sharedPrintController];
```

単純な画像一つを出力する例　↓

```
// PrintInfo の取得と各種プロパティの設定
UIPrintInfo *printInfo = [UIPrintInfo printInfo];
UIImage *image = ((UIImageView *)self.view).image;
printInfo.outputType = UIPrintInfoOutputPhoto;
printInfo.jobName = @"Image from PrintPhoto";
printInfo.duplex = UIPrintInfoDuplexNone;
if (!controller.printingItem && image.size.width > image.size.height)
    printInfo.orientation = UIPrintInfoOrientationLandscape;
```

画像の取得
画像出力を設定
適当にjob名を決める
片面印刷
横長なら
Landscape モードで

準備：後処理の設定

- 後処理として completion-handler を予め登録する
- エラー等の処理も行う
- 実装はブロック形式

```
void (^completionHandler)(UIPrintInteractionController *, BOOL, NSError *) =
        ^(UIPrintInteractionController *pic, BOOL completed, NSError *error) {
    self.content = nil;
    if (!completed && error)
        NSLog(@"FAILED! due to error in domain %@ with error code %u",
            error.domain, error.code);
  };
```

こりゃもう猿まねですかね？

実行：行ってこい

- 既に作成・準備したコントローラに対して「行ってこい」
- iPad, iPhone で個別に用意（表示の作法からして異なる）

「行ってこい」処理：[controller presentAnimated: ...]

```
if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad) {
    [controller presentFromBarButtonItem:self.printButton animated:YES
        completionHandler:completionHandler];
} else {
    [controller presentAnimated:YES completionHandler:completionHandler];
}
```

iPadだったら

バーから飛び出す

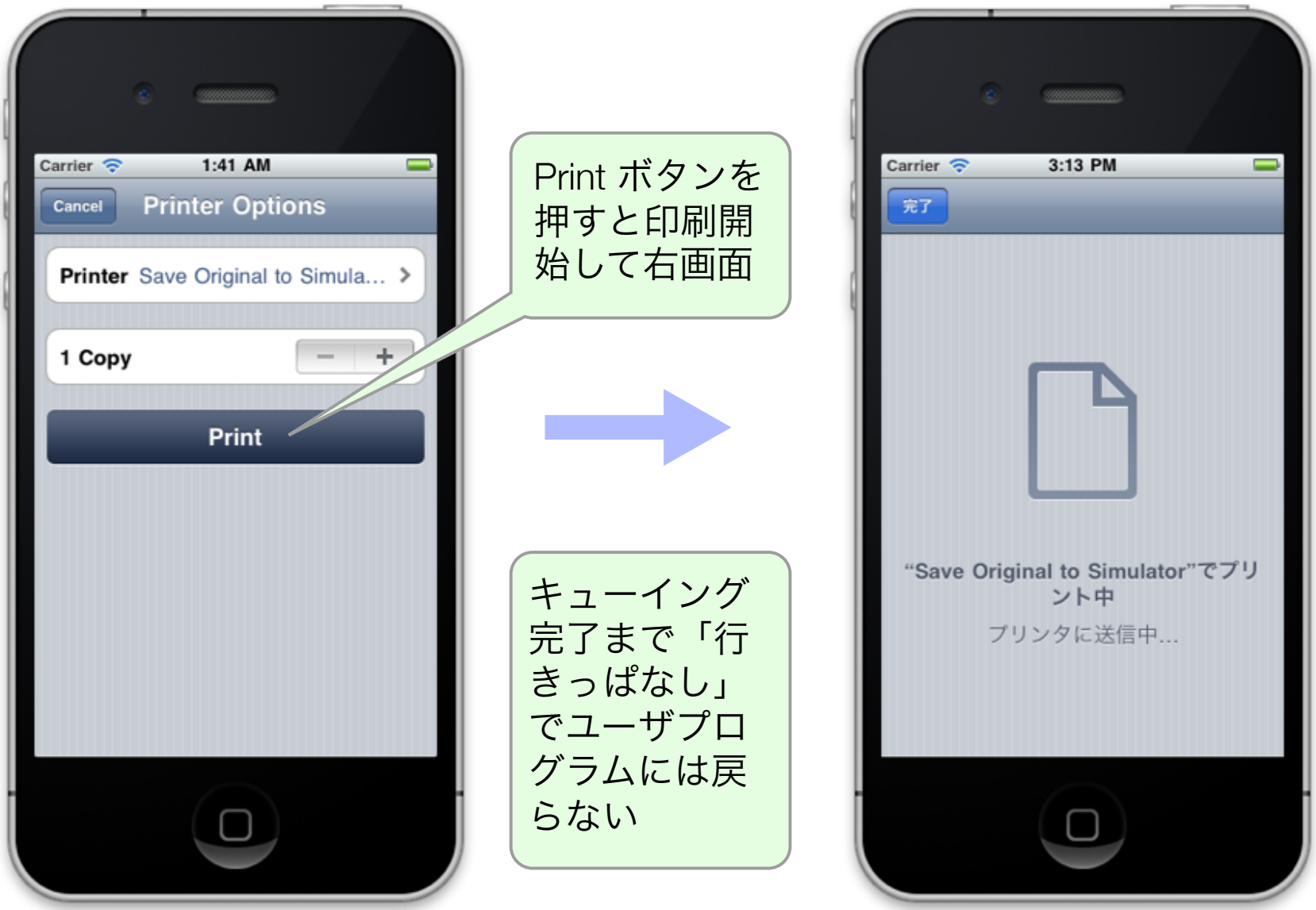
iPhone なら下から出す

先述の completionHandler はこのタイミングで仕掛ける

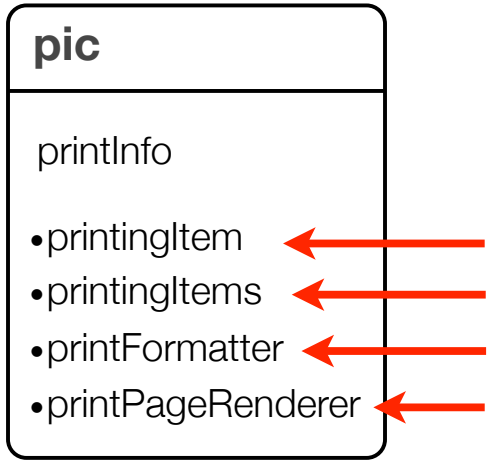
実行：Printer Options の表示

- 完成した controller に「行ってこい」で出る最初の画面
- どの位置から出すか、で三つも用意されてる
  - presentFromBarButtonItem:completionHandler:
    ナビゲーションバーから飛び出す
  - presentFromRect:inView:animated:completionHandler:
    任意の矩形領域から出す
  - presentAnimated:completionHandler:
    画面下から出す

実行：Printer Options の表示



４種類の出力支援機能（再掲）



- 四つのプロパティ　　これをやってみよう
  - printingItem - image あるいは PDF 用 **(#1)**
  - printingItems - printingItem の Array **(#1')**
  - printFormatter - UIPrintFormatter 対応のviewとtext, HTML 用 **(#2)**
  - printPageRenderer - UIPrintPageRenderer 用 **(#3)**
- どれか一つにコンテンツをセットして「行ってこい」で、
  それにふさわしい方法でレイアウトを支援する

実装：例えばこんな配置

(#123 共通)

printTestViewController.h

```
#import <UIKit/UIKit.h>

@interface printTestViewController : UIViewController
                            <UIPrintInteractionControllerDelegate> {
}

-(IBAction)printContent:(id)sender;

@end
```

例えば「ボタンを押したら出力」とすると、その実装はこんな感じになる

printTestViewController.m

```
-(IBAction)printContent:(id)sender {
    ....
    1.コントローラを用意して
    2. 印刷するデータ（text or PDF）を読み込んで
    3. それが印刷できる内容かどうか事前チェックして
    4. 問題なければ設定項目とデータをセットして
    5. 行ってこい
    ....
}
```

詳細は次スライド以降

実装概略（printingItem によるシンプルな方法）(#1)

```
-(IBAction)printContent:(id)sender {

  NSData *myData = [....] 出力データの読み出し
  UIPrintInteractionController *pic = ... pic を用意

  if( pic && ... ) { 対象データが印刷可能なものどうか確認

     pic.delegate = self; delegate の設定

     pic, printinfo を対象に各種設定項目を指定
     pic.printingItem = myData;  pic にプリント対象を指定
     終了時ハンドラを設定

     [pic プリントダイアログを出して処理を全部やんなさい];
  }
}
```

印刷可否チェック

- データが印刷可能なものかどうかチェックする
- “before assigning objects, you should validate the objects”
  とあるのでかなりマジな話らしい

```
// とりあえず printInteractionController を用意
UIPrintInteractionController *pic = [UIPrintInteractionController sharedPrintController];

// printingItem の機能で直接プリント可能かどうか事前チェック
if(pic && [UIPrintInteractionController canPrintData: myData] ) {
        pic.printingItem = myData;
        ..... 続きの処理
```

canPrintURL: もある

「imageのUTI があるなら printableUTIs クラスメソッドで評価可能」だそうだが意味不明

```
if ([[UIPrintInteractionController printableUTIs] containsObject:myImageUTI])
    pic.printingItem = myImage;
```

```
-(IBAction)printContent:(id)sender {
   NSString *path = [[NSBundle mainBundle] pathForResource:@"sample" ofType:@"jpg"];
   NSData *myData = [NSData dataWithContentsOfFile: path];

   UIPrintInteractionController *pic =
       [UIPrintInteractionController sharedPrintController];

   if(pic && [UIPrintInteractionController canPrintData: myData] ) {

       pic.delegate = self;

       UIPrintInfo *printInfo = [UIPrintInfo printInfo];
       printInfo.outputType = UIPrintInfoOutputGeneral;
       printInfo.jobName = [path lastPathComponent];
       printInfo.duplex = UIPrintInfoDuplexLongEdge;
       pic.printInfo = printInfo;
       pic.showsPageRange = YES;
       pic.printingItem = myData;

       void (^completionHandler)(UIPrintInteractionController *, BOOL, NSError *) =
       ^(UIPrintInteractionController *pic, BOOL completed, NSError *error) {
          //self.content = nil;
          if (!completed && error) {
             NSLog(@"FAILED! due to error in domain %@ with error code %u",
                error.domain, error.code);
          }
       };

       [pic presentAnimated:YES completionHandler:completionHandler];
   }
}
```

ファイル読み出し

pic 準備

印刷可否の確認

delegate の設定

printInfo を取得
用紙は普通で
job name はどうやらお決まり？
両面印刷時の綴じ込み側マージン？

printInfo を pic に登録
ページ範囲設定用の UI を表示する
印刷データはこれだ！！！ ← (#1 特有の箇所)

終了時のハンドラーを設定

プリントダイアログを出して処理を全部やんなさい、という決めゼリフ

参考：NSDataへの読み込みとNSURL での参照

printingItem **(#1)** による画像あるいは PDF の直接印刷では、

データを (1) NSData に読み込むか (2) NSURL で参照した状態を作る

NSData に読み込む場合

```
NSString *path = [[NSBundle mainBundle] pathForResource:@"sample" ofType:@"jpg"];
NSData *myData = [NSData dataWithContentsOfFile: path];

if(pic && [UIPrintInteractionController canPrintData:myData]) {
  pic.printingItem = myData;
  ....
```

NSURL で参照する場合

```
NSString *path = [[NSBundle mainBundle] pathForResource:@"sample" ofType:@"jpg"];
NSURL *myURL = [NSURL fileURLWithPath: path];

if(pic && [UIPrintInteractionController canPrintURL:myURL]) {
  pic.printingItem = myURL;
  ....
```

下記の書き方では動作しないが何故？

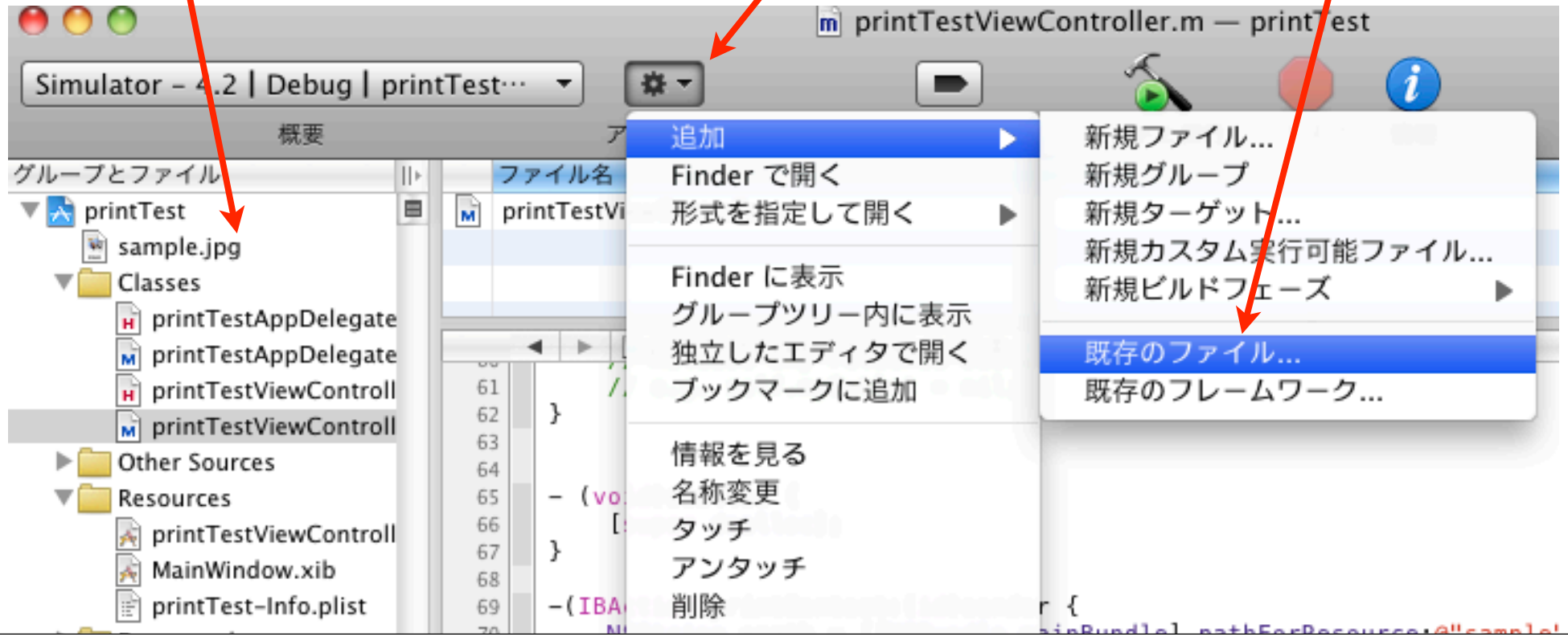
```
NSURL *myURL = [NSURL URLWithString:@"file:///sample.jpg"];
```

参考：ファイルの配置

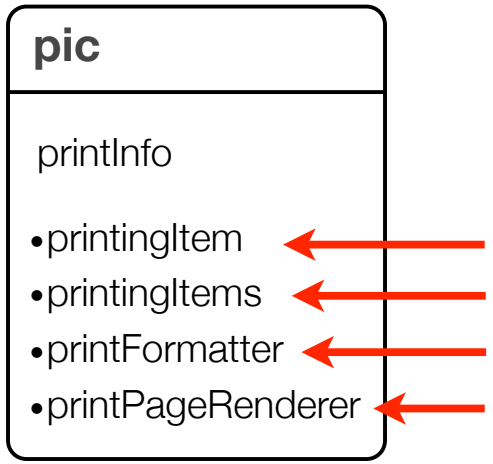
step 3.
（どこでもいいけど例えば）
ここに置く

step 1.
ここをクリック

step 2.
これを選択



４種類の出力支援機能（再掲）



- 四つのプロパティ
  - printingItem - image あるいは PDF 用 **(#1)** **done**
  - printingItems - printingItem の Array **(#1')** ただ複数出すだけなのでパス
  - printFormatter - UIPrintFormatter 対応のviewとtext, HTML 用 **(#2)**
  - printPageRenderer - UIPrintPageRenderer 用 **(#3)** 次はこれ
- どれか一つにコンテンツをセットして「行ってこい」で、
  それにふさわしい方法でレイアウトを支援する

利用するレイアウト機能によるコードの変化

```
-(IBAction)printContent:(id)sender {

  NSData *myData = [....] 出力データの読み出し
  UIPrintInteractionController *pic = ... pic を用意

  if( pic && ... ) { 対象データが印刷可能なものどうか確認

     pic.delegate = self; delegate の設定

     pic, printinfo を対象に各種設定項目を指定
     pic.printingItem = myData;  pic にプリント対象を指定
     終了時ハンドラを設定

     [pic プリントダイアログを出して処理を全部やんなさい];
  }
}
```

ここが変化する

printFormatter を使ったレイアウト調整 (#2)

printingItem(s) - image あるいは PDF を単純印刷する場合

```
pic.printingItem = myData; // 直接データを放り込むだけ
```

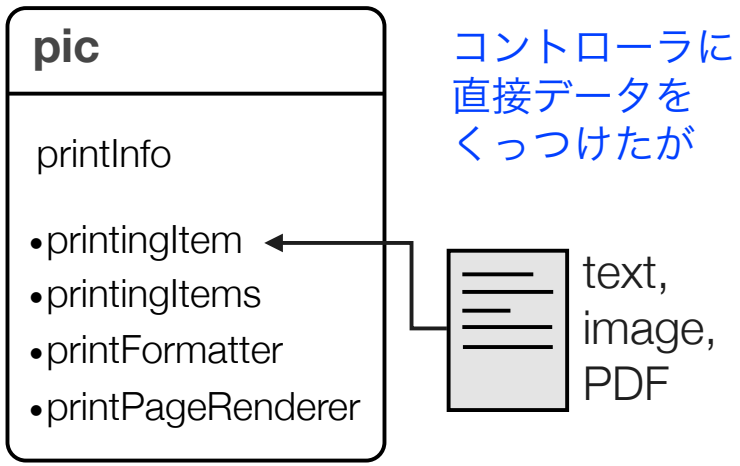
printFormatter - text, HTML と幾つかの View をレイアウトする

例はtextの場合

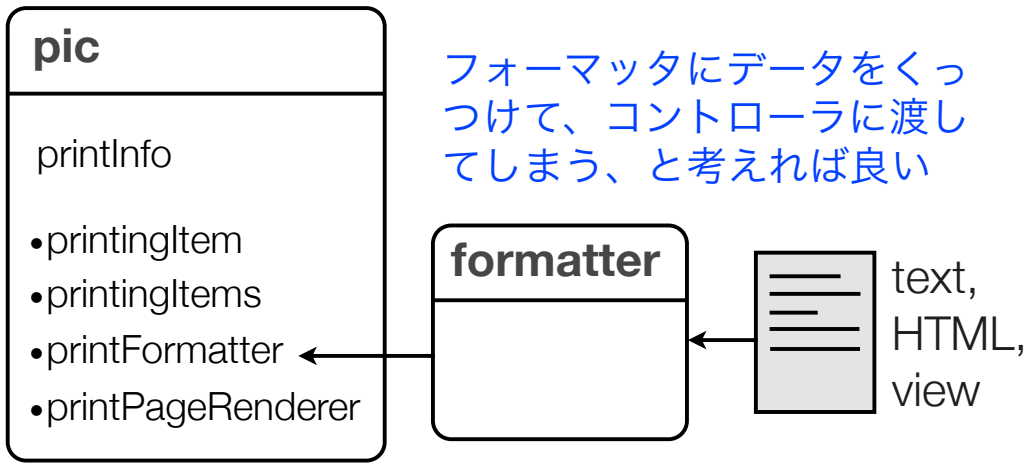
```
UISimpleTextPrintFormatter *fmt =
  [[UISimpleTextPrintFormatter alloc] initWithText:@"Hello!!"];
pic.printFormatter = fmt;
```

他に fmt を通して font, color, lineBreakMode, textAlignment の設定が可能

printingItem の場合 (#1)



printingFormatter の場合 (#2)



マージン等の設定

(#2)

- Figure 6-8 The layout of a multi-page print job を参照
- printFormatではヘッダ・フッタの高さ設定はできない
- 設定可能なのは inset と maximumContentWidth, Height のみ

単位は points
(72points / inch)

```
fmt.contentInsets = UIEdgeInsetsMake(72.0, 72.0, 72.0, 72.0);
fmt.maximumContentWidth = 6 * 72.0;
```

http://developer.apple.com/library/ios/#documentation/2DDrawing/Conceptual/DrawingPrintingiOS/Printing/Printing.html

UISimpleTextPrintFormatter による text 整形 (#2)

```
UISimpleTextPrintFormatter *textFormatter =                    // formatter を用意する
        [[UISimpleTextPrintFormatter alloc] initWithText:@"....”];

textFormatter.startPage = 0;

textFormatter.contentInsets = UIEdgeInsetsMake(72.0, 72.0, 72.0, 72.0); // 1”
textFormatter.maximumContentWidth = 6 * 72.0;

pic.printFormatter = textFormatter;  // formatter を与える
```

NSString

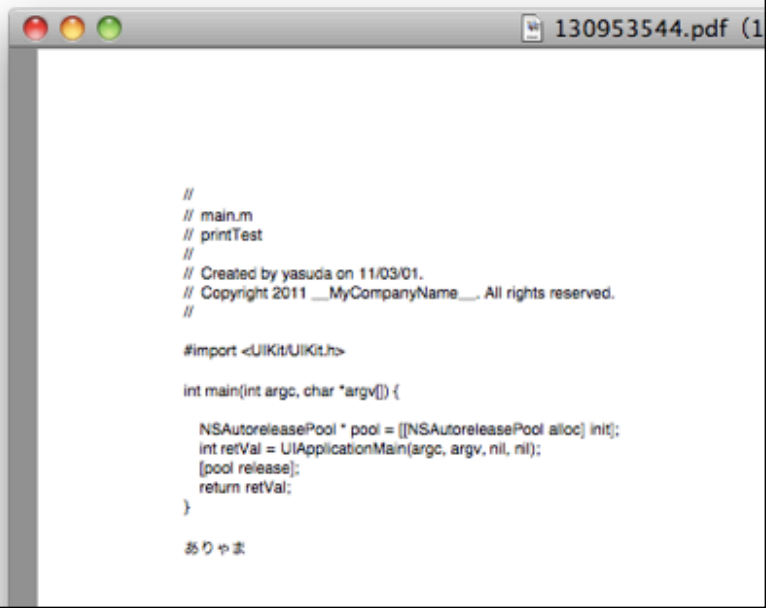
Printer Simulator 経由の Preview

```
//
//  main.m
//  printTest
//
//  Created by yasuda on 11/03/01.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

int main(int argc, char *argv[]) {

    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    int retVal = UIApplicationMain(argc, argv, nil, nil);
```



参考：text ファイルの読み込みとエラー処理

```
NSString *path = [[NSBundle mainBundle] pathForResource:@"sample" ofType:@"txt"];

NSError *error=nil;
```

エラー受け口

path のファイルから読む

```
NSString *text = [[NSString alloc] initWithContentsOfFile:path
                                                 encoding:NSUTF8StringEncoding
                                                    error:&error];


if([[error domain]isEqual:NSCocoaErrorDomain]) {
   UIAlertView* alert =
       [[UIAlertView alloc] initWithTitle:@"text file reading"
                                  message:[error localizedDescription]
                                 delegate:self
                        cancelButtonTitle:@"OK"
                        otherButtonTitles:nil];
   [alert show];
   [alert release];
   return;
}
```

エラーがあれば Alert を出す

注意：
エンコーディングが違っても
このエラーが出る

テキストの改行文字

ところで改行も
要注意です

現象：

一行空行があれば改行するが、

それ以外はつなげてしまう

原因：

改行コードが CR だった

→ LF にする

元データ

```
//
//  main.m
//  printTest
//
//  Created by yasuda on 11/03/01.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//
                                                    ◀ 空行
#import <UIKit/UIKit.h>
                                                    ◀ 空行
int main(int argc, char *argv[]) {
                                                    ◀ 空行
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    int retVal = UIApplicationMain(argc, argv, nil, nil);
    [pool release];
    return retVal;
}
                                                    ◀ 空行
ありゃま
```

出力結果

```
//// main.m// printTest//// Created by yasuda on 11/03/01.// Copyright 2011 __MyCompanyName__.
All rights reserved.//
#import <UIKit/UIKit.h>
int main(int argc, char *argv[]) {      NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];    int
retVal = UIApplicationMain(argc, argv, nil, nil);    [pool release];    return retVal;}
ありゃま
```

lineBreakMode（がない！）

lineBreakMode なる設定項目がテキスト用フォーマッタにあると言うが、実際に書いてみると

```
textFormatter.lineBreakMode = UILineBreakModeWordWrap;
```

そんなプロパティは無いと。。。　　じっと定義を見る、、、

```
UIKIT_CLASS_AVAILABLE(4_2)
   @interface UISimpleTextPrintFormatter : UIPrintFormatter {
}

- (id)initWithText:(NSString *)text;
@property(nonatomic,copy)     NSString       *text;
@property(nonatomic,retain)   UIFont         *font;
@property(nonatomic,retain)   UIColor        *color;
@property(nonatomic)          UITextAlignment textAlignment;

@end
```

確かにない、、（ドキュメントにはあるのよマヂで 2011.3 現在）

ご注意ください皆様

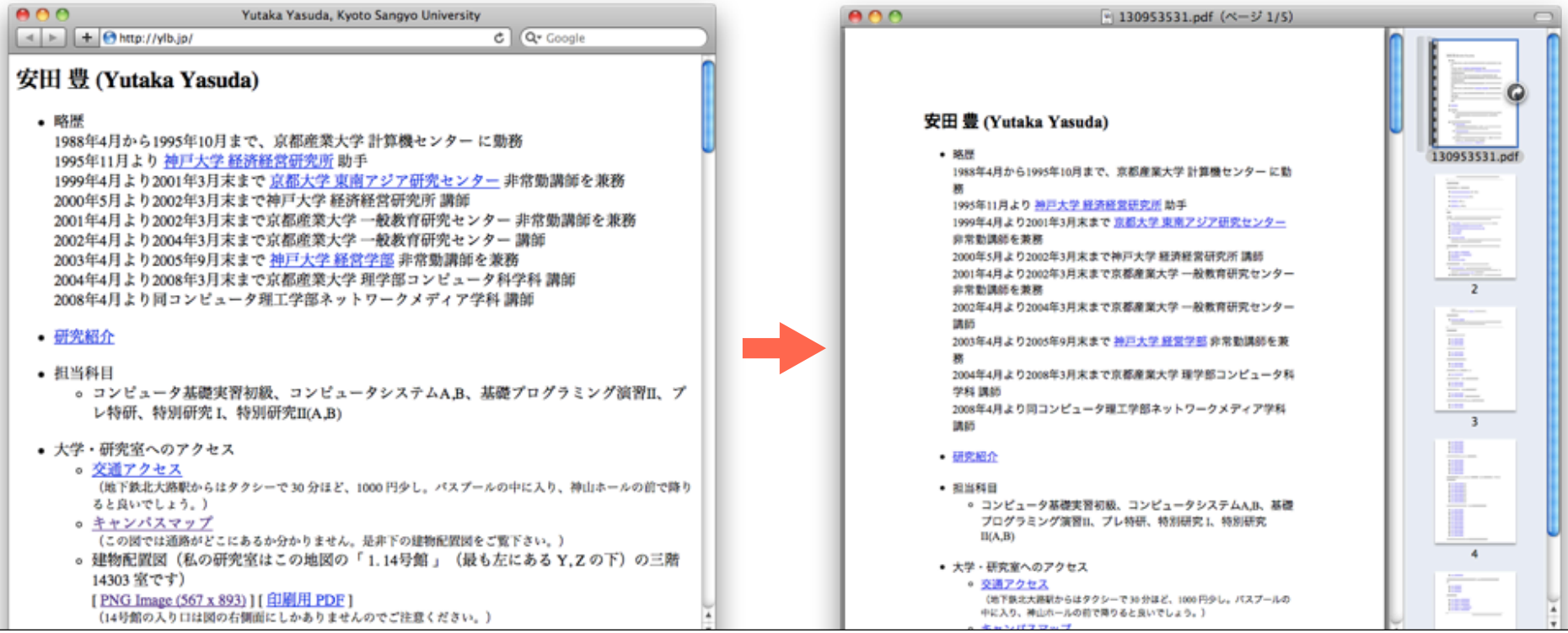
UIMarkupTextPrintFormatter による HTML 整形 (#2)

```
UIMarkupTextPrintFormatter *htmlFormatter =                    // formatter を用意する
        [[UIMarkupTextPrintFormatter alloc] initWithText:@"<HTML>...."];

pic.printFormatter = htmlFormatter;  // formatter を与える
```

NSString で HTML

フォーマッターが違うだけで UISimpleTextPrintFormatterと全く同じ



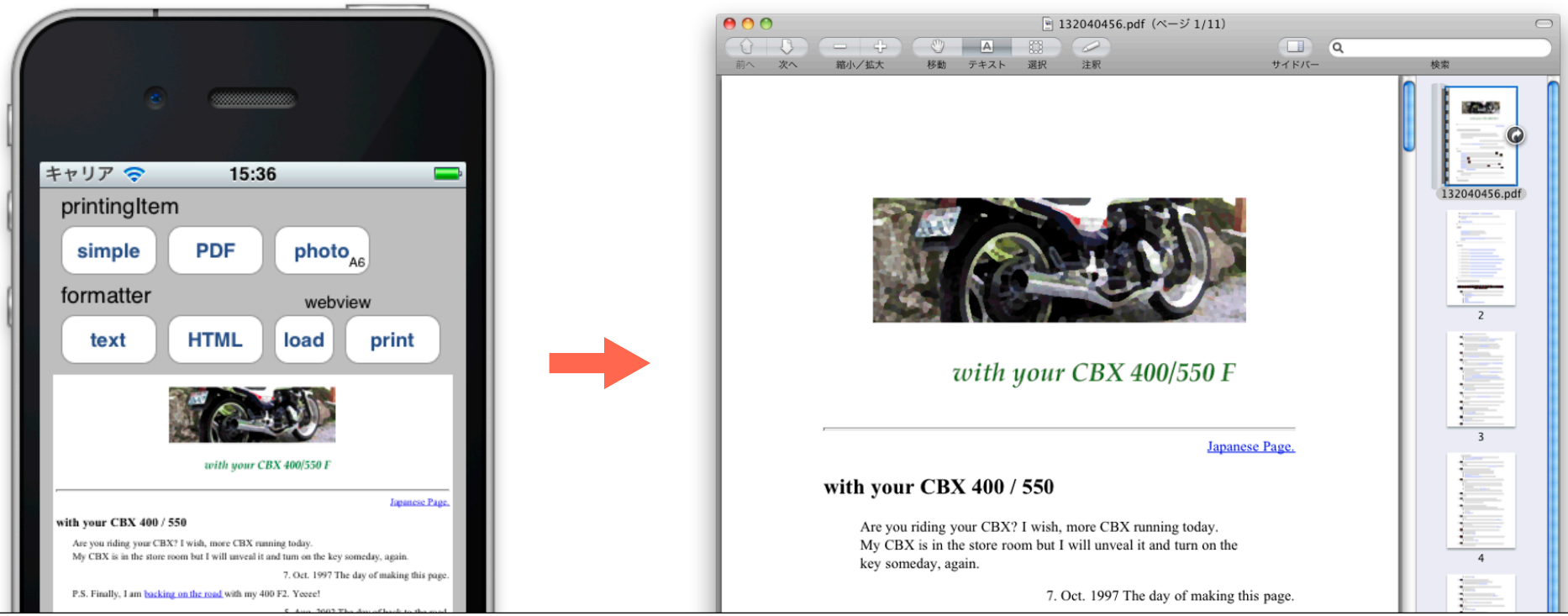
UIViewPrintFormatter による webView の整形 (#2)

```
UIViewPrintFormatter *viewFormatter =        // formatter を用意する
        [myWebView viewPrintFormatter];
```

webView に聞くとフォーマッタが得られる

```
pic.printFormatter = viewFormatter;  // formatter を与える
```

やはりフォーマッターが違うだけで UISimpleText, UIMarkUpTextPrintFormatterと同じ



各種フォーマッター比較

(#2)

UISimpleTextPrintFormatter : plain text 用

```
UISimpleTextPrintFormatter *formatter =            // formatter を用意する
        [[UISimpleTextPrintFormatter alloc] initWithText:@"my name is ...."];

pic.printFormatter = formatter;  // formatter を与える
```

NSString で plain text

UIMarkUpTextPrintFormatter : HTML 用

```
UIMarkupTextPrintFormatter *formatter =            // formatter を用意する
        [[UIMarkupTextPrintFormatter alloc] initWithText:@"<HTML>...."];

pic.printFormatter = formatter;  // formatter を与える
```

NSString で HTML

UIViewPrintFormatter : View 用

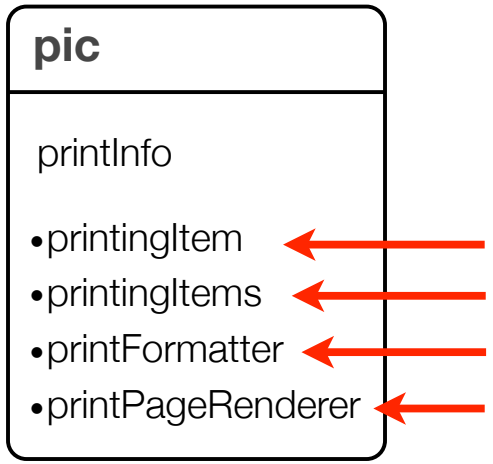
```
UIViewPrintFormatter *formatter =       // formatter を用意する
       [myWebView viewPrintFormatter];

pic.printFormatter = formatter;  // formatter を与える
```

これだけ作法が違う

作法の違いは UIWebView 専用でなく UITextView, MKMapView に対応しているため

４種類の出力支援機能（再掲）

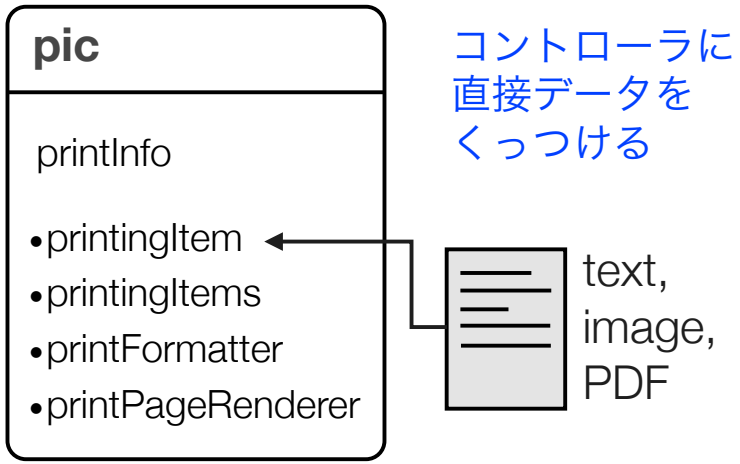


- 四つのプロパティ
  - printingItem - image あるいは PDF 用 **(#1)** **done**
  - printingItems - printingItem の Array **(#1')** パス
  - printFormatter - UIPrintFormatter 対応のviewとtext, HTML 用 **(#2)** **done**
  - printPageRenderer - UIPrintPageRenderer 用 **(#3)**
- どれか一つにコンテンツをセットして「行ってこい」で、それにふさわしい方法でレイアウトを支援する

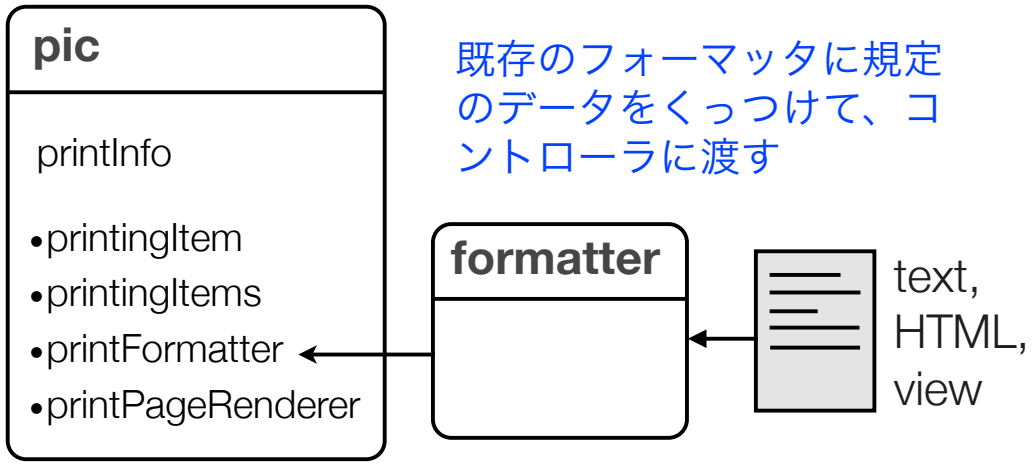
次はこれ

printPageRenderer を使ったレイアウト調整 (#3)

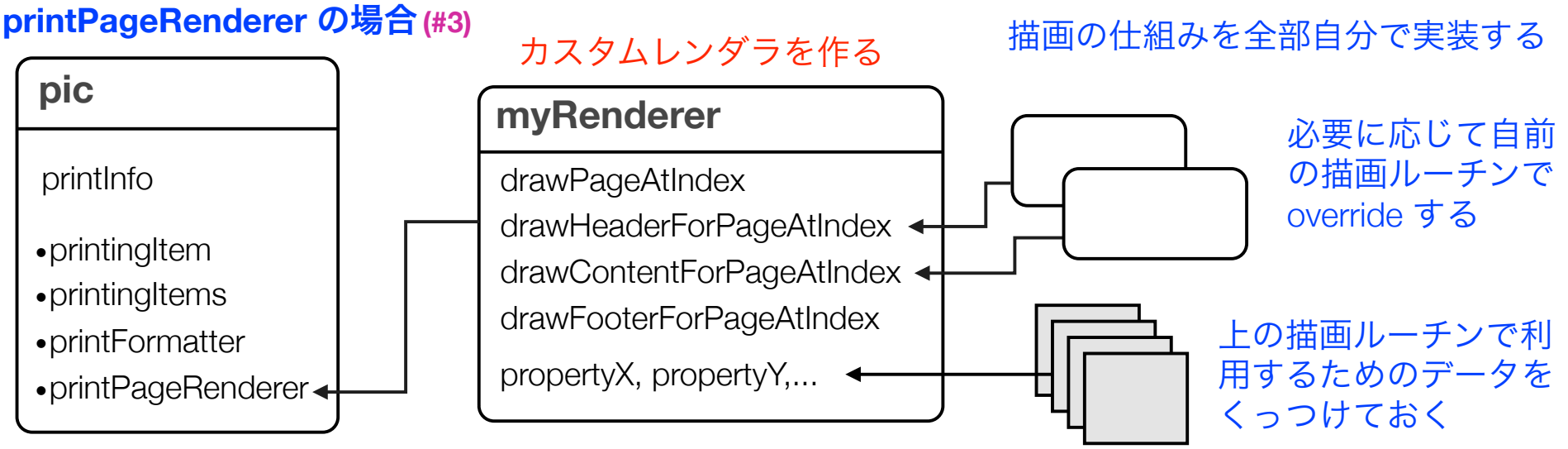
printingItem の場合 (#1)



printingFormatter の場合 (#2)



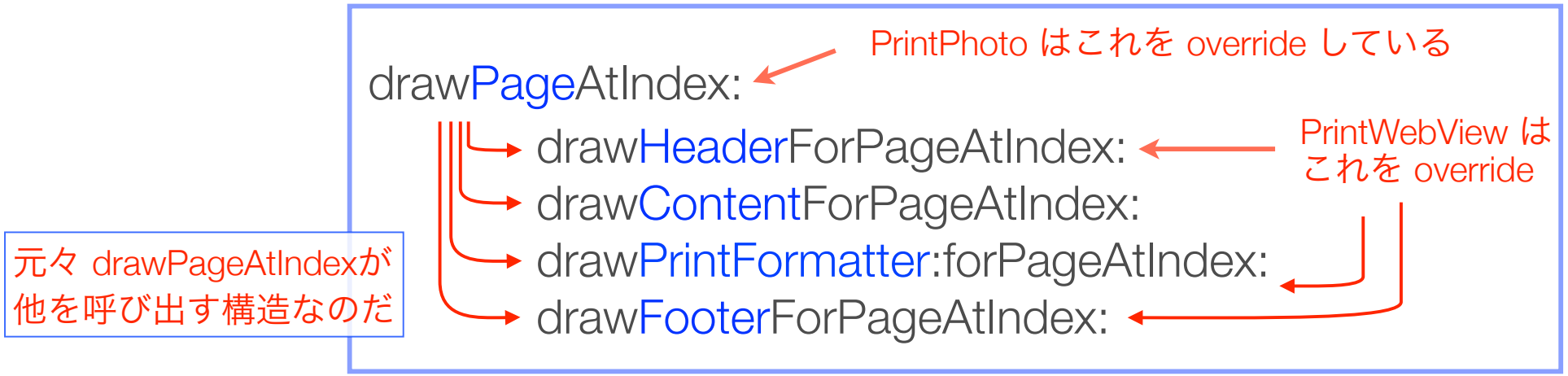
printPageRenderer の場合 (#3)



実装：override する箇所

(#3)

- drawPageAtIndex：全領域を自分で支配する場合に置換
- drawHeaderForPageAtIndex, drawFooterForPageAtIndex：
  ヘッダ、フッタの置換
- drawContentForPageAtIndex：コンテント部分の描画を置換
- drawPrintFormatter：コンテント部分に既存のフォーマッターを適用



http://developer.apple.com/library/ios/#samplecode/PrintPhoto/Introduction/Intro.html

コード例：PrintPhoto

(#3)

PrintPhotoPageRenderer.h

UIPrintPageRenderer のサブクラスを作成

```
@interface PrintPhotoPageRenderer : UIPrintPageRenderer {
  UIImage *imageToPrint;
}
```

描画データなどを保存するためにプロパティ等を用意

PrintPhotoPageRenderer.m

```
- (void)drawPageAtIndex:(NSInteger)pageIndex inRect:(CGRect)printableRect
{
    ... プロパティ等に設定した描画データをprintableRect範囲内になんとかして描画する

    destRect = CGRectMake(......);
    [self.imageToPrint drawInRect:destRect];
}
```

今回はUIImageなので
drawInRectで楽に描画可能

PrintPhotoViewController.m

用意したサブクラスのインスタンスを得る

```
PrintPhotoPageRenderer *pageRenderer = [[PrintPhotoPageRenderer alloc] init];
pageRenderer.imageToPrint = ((UIImageView *)self.view).image;  イメージをセット

controller.printPageRenderer = pageRenderer;  レンダラーをセット
```

UIPrintInteractionController のインスタンス

renderer のdrawPageAtIndex: は印刷の際にシステム側から呼び返される
つまりデータをセットして、呼ばれるのを待つ格好で controller に渡して放置、な状態

マージン等の設定

(#3)

- Figure 6-8 The layout of a multi-page print job を参照
- renderer ではヘッダ・フッタの高さ設定ができる
- 逆に inset と maximumContentWidth, Height は設定できない

単位は points
(72points / inch)

```
pageRenderer.headerHeight = 10;
```
のようにして値を設定する

http://developer.apple.com/library/ios/#documentation/2DDrawing/Conceptual/DrawingPrintingiOS/Printing/Printing.html

ヘッダ・フッタ高さの受け渡し (#3)

SamplePrintViewController.m

```
PrintGridPageRenderer *pageRenderer = [[PrintGridPageRenderer alloc] init];
...
pageRenderer.headerHeight = 10.0;
pageRenderer.footerHeight = 15.0;
```

このようにしてレンダラ側の値を設定して印刷処理に飛び込むと

SamplePrintPageRenderer.m

```
-(void)drawHeaderForPageAtIndex:(NSInteger)index inRect:(CGRect)headerRect
       drawFooterForPageAtIndex:
```

ここに設定した領域が渡されてくる

```
-(void)drawContentForPageAtIndex:(NSInteger)index inRect:(CGRect)contentRect
```

こちらは描画可能な領域が計算されて与えられる

参考：文字の描画

や、、ヘッダやフッタの処理で要るかなと･･･

文字の背の高さの算出と設定

```
UIFont *font = [UIFont fontWithName:@"Helvetica" size:10];
CGSize titleSize = [@"Sample Header String." sizeWithFont:font];
pageRenderer.headerHeight = titleSize.height * 1.5;
```

文字の描画

```
UIFont *font = [UIFont fontWithName:@"Helvetica" size:10];
[@"Sample Header String." drawAtPoint:startPoint withFont:font];
```

ヘッダ領域の縦方向中央位置にヘッダ文字列を描きたい

```
-(void)drawHeaderForPageAtIndex:(NSInteger)index inRect:(CGRect)headerRect

    UIFont *font = [UIFont fontWithName:@"Helvetica" size:10];
    CGFloat startX = headerRect.origin.x + 10;
    CGSize  stringSize = [headerString sizeWithFont:font];
    CGFloat startY = headerRect.origin.y +
                    ( headerRect.size.height - stringSize.height ) / 2 ;
    CGPoint startPoint = CGPointMake(startX, startY);
    [headerString drawAtPoint:startPoint withFont:font];
```

おまけ

- ヘッダ、フッタは printFormatter では指定できない
    - それは printPageRenderer カスタムサブクラスを作って処理せよ
    - では printFormatter のカスタムサブクラスで実装する手は？
- と思ったら UIPrintFormatter Class Reference に注意アリ

    “Third-party subclasses of UIPrintFormatter are not recommended. If you have custom content to print, use a custom UIPrintPageRenderer object.”
- つまりカスタムレンダラを作ってヘッダ、フッタを指定し、コンテント部分に printFormatter を使うように指定せよ、と
- これの真っ当な例が PrintWebView アプリである

http://developer.apple.com/library/ios/#documentation/uikit/reference/UIPrintFormatter_Class/
http://developer.apple.com/library/ios/#samplecode/PrintWebView/

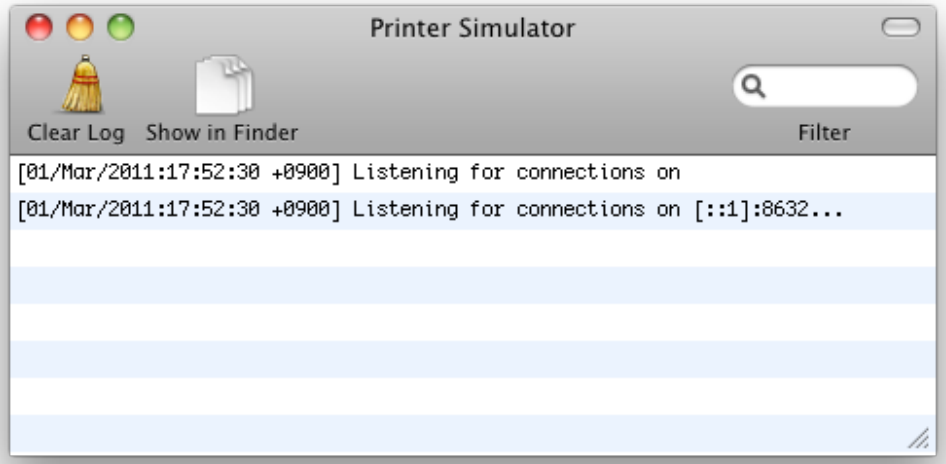
そろそろ結果を見よう

準備：PrinterSimulator.app

- 開発時に仮想プリンタとなるシミュレータ
- 場所はここ

  /Platforms/iPhoneOS.platform/Developer/Applications/PrinterSimulator.app

- 予め手で起動しておく



実行

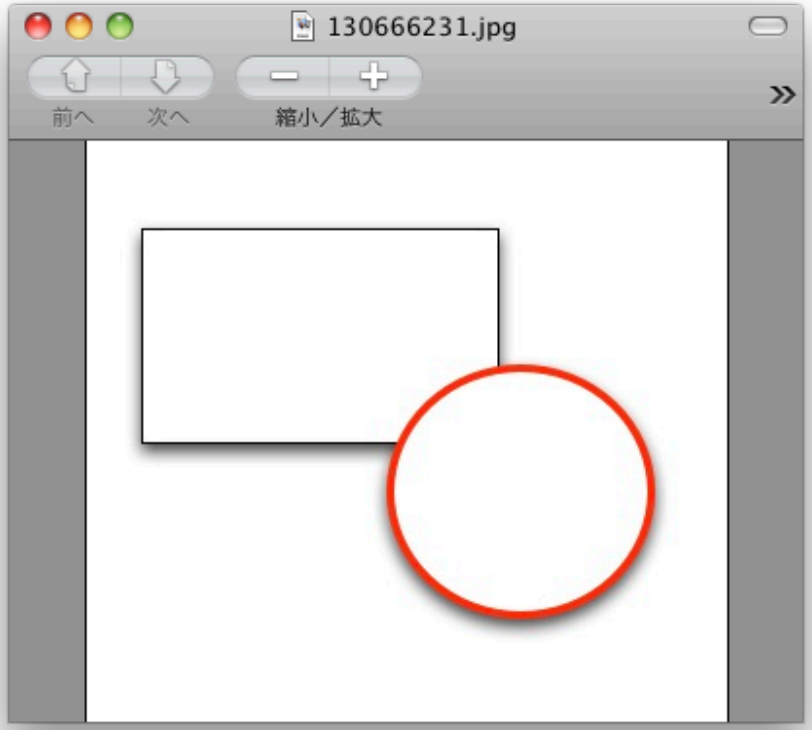
- Print Simulator を起動しておく
  （事後でも良いが）
- ボタン押下で Printer Options を表示
- Select Printer > を選ぶ

プリンタを選択

- Printer Simulator 要起動
- 一覧が見えるはず
- Save Original to Simulator を選択

印刷！

- Print をタップ

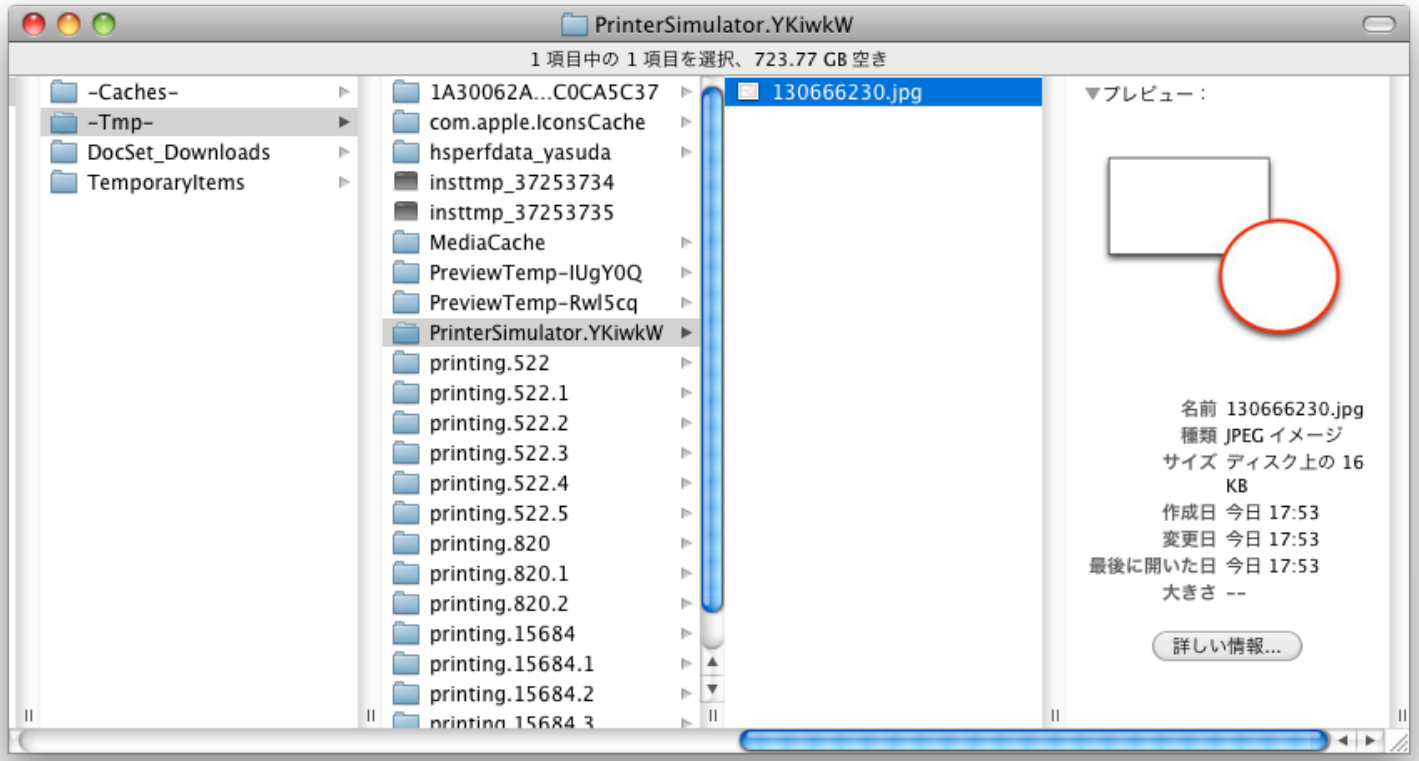


ボン、と Preview で出てくる

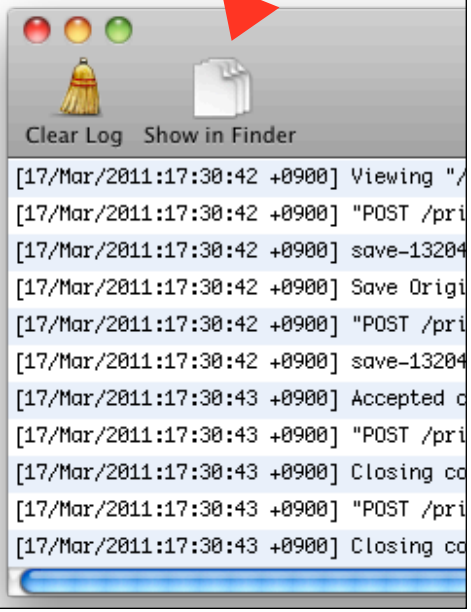
キュー

- ホームボタン二度押し
- 最も左側に出る
- タップすることで当該印刷要求のキャンセルなどが可能
- 印刷完了後自動的に消滅

シミュレータの出力はどこに？

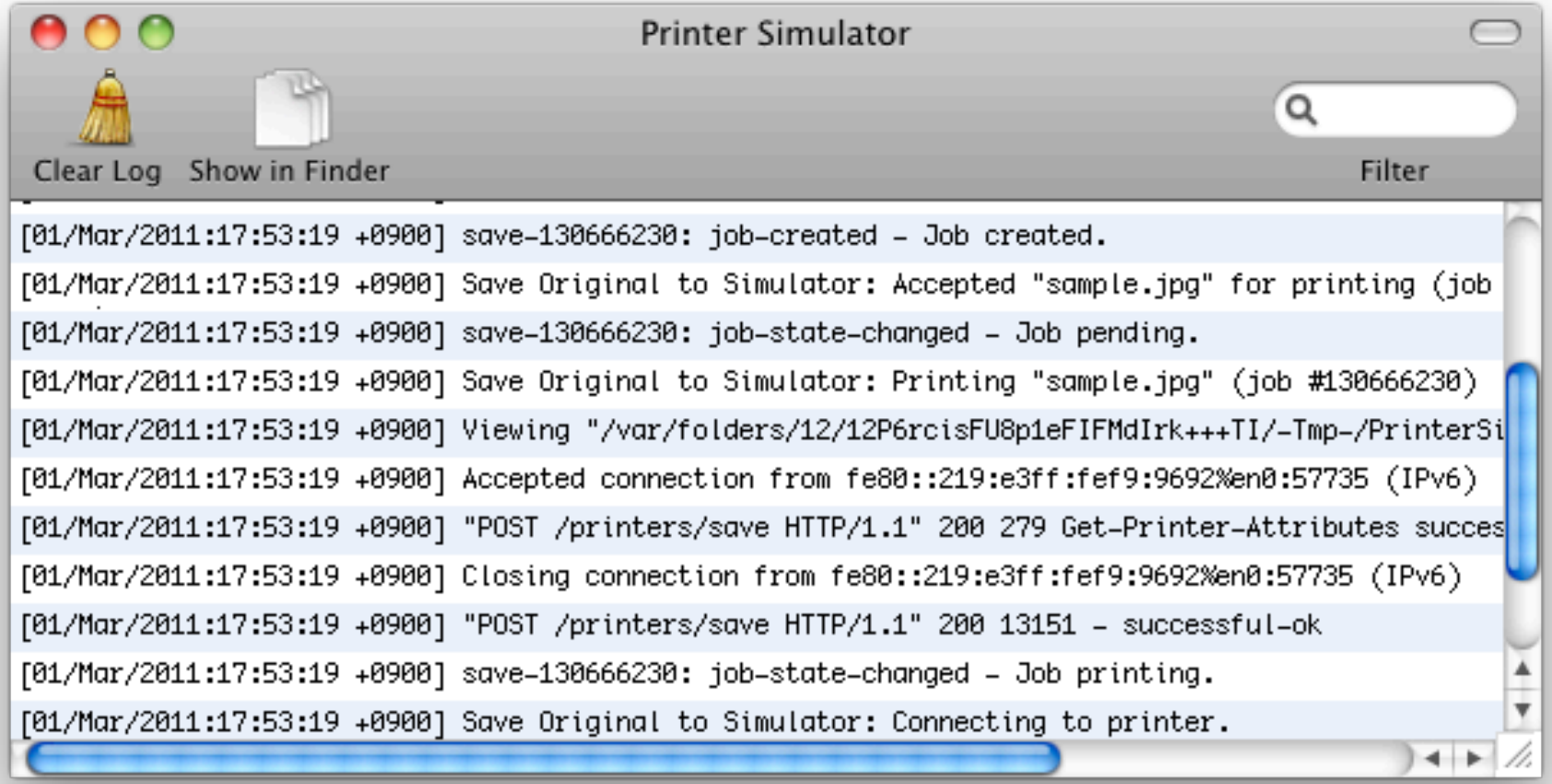


Show in Finder
をクリック



/private/var/folders/12/12P8rcisFU8d1eFIFMdIrk+++TI/-Tmp-/PrinterSimulator.YKiwkW

ログ


Printer Simulator
Clear Log
Show in Finder
Filter
[01/Mar/2011:17:53:19 +0900] save-130666230: job-created - Job created.
[01/Mar/2011:17:53:19 +0900] Save Original to Simulator: Accepted "sample.jpg" for printing (job
[01/Mar/2011:17:53:19 +0900] save-130666230: job-state-changed - Job pending.
[01/Mar/2011:17:53:19 +0900] Save Original to Simulator: Printing "sample.jpg" (job #130666230)
[01/Mar/2011:17:53:19 +0900] Viewing "/var/folders/12/12P6rcisFU8p1eFIFMdIrk+++TI/-Tmp-/PrinterSi
[01/Mar/2011:17:53:19 +0900] Accepted connection from fe80::219:e3ff:fef9:9692%en0:57735 (IPv6)
[01/Mar/2011:17:53:19 +0900] "POST /printers/save HTTP/1.1" 200 279 Get-Printer-Attributes succes
[01/Mar/2011:17:53:19 +0900] Closing connection from fe80::219:e3ff:fef9:9692%en0:57735 (IPv6)
[01/Mar/2011:17:53:19 +0900] "POST /printers/save HTTP/1.1" 200 13151 - successful-ok
[01/Mar/2011:17:53:19 +0900] save-130666230: job-state-changed - Job printing.
[01/Mar/2011:17:53:19 +0900] Save Original to Simulator: Connecting to printer.

ログ

[01/Mar/2011:17:52:30 +0900] Listening for connections on 0.0.0.0:**8632**..
[01/Mar/2011:17:52:30 +0900] Listening for connections on [::1]:**8632**..
[01/Mar/2011:17:53:15 +0900] **Accepted connection** from fe80::219:e3ff:fef9:9692%en0:57733 (**IPv6**)
[01/Mar/2011:17:53:15 +0900] "**POST /printers/save HTTP/1.1**" 200 3006 **Get-Printer-Attributes successful-ok**
[01/Mar/2011:17:53:15 +0900] Closing connection from fe80::219:e3ff:fef9:9692%en0:57733 (IPv6)
[01/Mar/2011:17:53:19 +0900] Accepted connection from fe80::219:e3ff:fef9:9692%en0:57734 (IPv6)
[01/Mar/2011:17:53:19 +0900] "POST /printers/save HTTP/1.1" 200 262 **Validate-Job successful-ok**
[01/Mar/2011:17:53:19 +0900] save-130666230: **job-created - Job created**
[01/Mar/2011:17:53:19 +0900] Save Original to Simulator: **Accepted "sample.jpg" for printing**
(job #130666230, image/jpeg, 1 pages)
[01/Mar/2011:17:53:19 +0900] save-130666230: **job-state-changed - Job pending**
[01/Mar/2011:17:53:19 +0900] Save Original to Simulator: **Printing "sample.jpg"** (job #130666230)
[01/Mar/2011:17:53:19 +0900] Viewing "/var/folders/12/12P6rcisFU8p1eFIFMdIrk+++TI/-Tmp-/
PrinterSimulator.YKiwkW/130666230.jpg"
[01/Mar/2011:17:53:19 +0900] Accepted connection from fe80::219:e3ff:fef9:9692%en0:57735 (IPv6)
[01/Mar/2011:17:53:19 +0900] "POST /printers/save HTTP/1.1" 200 279 **Get-Printer-Attributes successful-ok**
[01/Mar/2011:17:53:19 +0900] Closing connection from fe80::219:e3ff:fef9:9692%en0:57735 (IPv6)
[01/Mar/2011:17:53:19 +0900] "POST /printers/save HTTP/1.1" 200 13151 - **successful-ok**
[01/Mar/2011:17:53:19 +0900] save-130666230: **job-state-changed - Job printing**
[01/Mar/2011:17:53:19 +0900] Save Original to Simulator: **Connecting to printer**
[01/Mar/2011:17:53:19 +0900] save-130666230: **job-completed - Job completed**
[01/Mar/2011:17:53:19 +0900] "POST /printers/save HTTP/1.1" 200 366 **Get-Job-Attributes successful-ok**
[01/Mar/2011:17:53:19 +0900] Closing connection from fe80::219:e3ff:fef9:9692%en0:57734 (IPv6)

いろいろ判ります・・・

終了・・・