Working with System Frameworks in Python and Objective-C

by James Barclay

Feedback :)

j.mp/psumac2015-62

Dude, Where's My Source Code?

CODE

https://github.com/futureimperfect/psu-pyobjc-demo

https://github.com/futureimperfect/PSUDemo

SLIDES

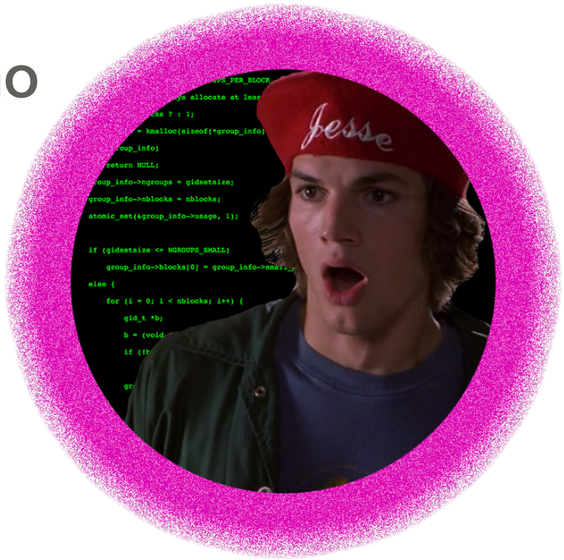
https://github.com/futureimperfect/slides

Dude, Where's My Source Code?

CODE

https://github.com/futureimperfect/psu-pyobjc-demo

https://github.com/futureimperfect/PSUDemo



SLIDES

https://github.com/futureimperfect/slides

Dude, Where's My Source Code?

CODE

https://github.com/futureimperfect/psu-pyobjc-demo

https://github.com/futureimperfect/PSUDemo

SLIDES

https://github.com/futureimperfect/slides

Agenda

1. What are system frameworks, and why should you care?
2. Brief overview of the frameworks, classes, and APIs that will be demonstrated.
3. Demo 1: PyObjC
4. Demo 2: Objective-C
5. Wrap up and questions.

What's a System Framework?

...and why should you care?

(OS X) system frameworks provide interfaces you need to write software for the Mac.

Many of these are useful for Mac admins creating:

- scripts
- GUI applications
- command-line tools

Learning about system frameworks will teach you more about OS X, which will probably make you a better admin.

```
/*  Foundation.h
    Copyright (c) 1994-2014, Apple Inc. All rights reserved.
*/


#include <CoreFoundation/CoreFoundation.h>

#import <Foundation/NSObjCRuntime.h>

#import <Foundation/NSArray.h>
#import <Foundation/NSAutoreleasePool.h>
#import <Foundation/NSBundle.h>
#import <Foundation/NSByteOrder.h>
#import <Foundation/NSCalendar.h>
#import <Foundation/NSCharacterSet.h>
#import <Foundation/NSCoder.h>
#import <Foundation/NSData.h>
#import <Foundation/NSDate.h>
#import <Foundation/NSDateFormatter.h>
#import <Foundation/NSDateIntervalFormatter.h>
#import <Foundation/NSMassFormatter.h>
#import <Foundation/NSLengthFormatter.h>
#import <Foundation/NSEnergyFormatter.h>
#import <Foundation/NSDecimal.h>
#import <Foundation/NSDecimalNumber.h>
#import <Foundation/NSDictionary.h>
#import <Foundation/NSEnumerator.h>
#import <Foundation/NSError.h>
#import <Foundation/NSException.h>
#import <Foundation/NSFileHandle.h>
#import <Foundation/NSFileManager.h>
#import <Foundation/NSFormatter.h>
#import <Foundation/NSHashTable.h>
#import <Foundation/NSHTTPCookie.h>
#import <Foundation/NSHTTPCookieStorage.h>
#import <Foundation/NSIndexPath.h>
#import <Foundation/NSIndexSet.h>
#import <Foundation/NSInvocation.h>
#import <Foundation/NSJSONSerialization.h>
#import <Foundation/NSKeyValueCoding.h>
#import <Foundation/NSKeyValueObserving.h>
#import <Foundation/NSKeyedArchiver.h>
#import <Foundation/NSLocale.h>
#import <Foundation/NSLock.h>
#import <Foundation/NSMapTable.h>
#import <Foundation/NSMethodSignature.h>
#import <Foundation/NSNotification.h>
#import <Foundation/NSNotificationQueue.h>
#import <Foundation/NSNull.h>
#import <Foundation/NSNumberFormatter.h>
#import <Foundation/NSObject.h>
#import <Foundation/NSOperation.h>
#import <Foundation/NSOrderedSet.h>
#import <Foundation/NSOrthography.h>
#import <Foundation/NSPathUtilities.h>
#import <Foundation/NSPointerArray.h>
#import <Foundation/NSPointerFunctions.h>
/System/Library/Frameworks/Foundation.framework/Versions/C/Headers/Foundatio
"/System/Library/Frameworks/Foundation.framework/Versions/C/Headers/Foundati
```

Frameworks, Classes, and APIs oh my!

Cocoa

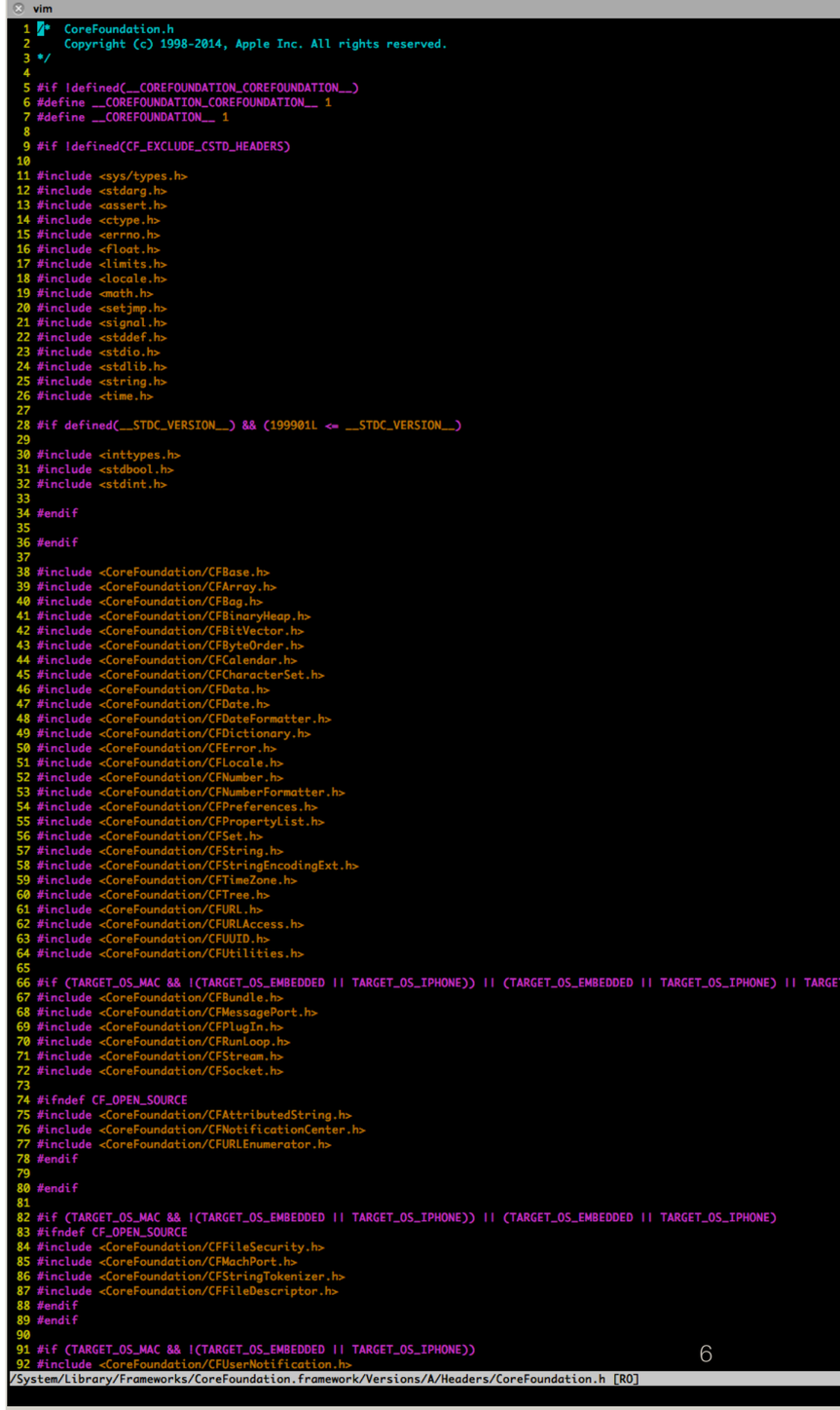
- Foundation
  - NSFileManager
  - NSTask
  - NSURLSession
  - NSUserDefaults
- AppKit
  - NSApplication

CoreFoundation

- CFPreferences

CoreGraphics

- Quartz



CoreFoundation

CoreFoundation is a C framework that knows about Objective-C objects.

Some parts of CoreFoundation are written in Objective-C.

- Other parts are written in C.

CoreFoundation uses the `CF` class prefix, and it provides `CFString`, `CFDictionary`, `CFPreferences`, and the like.

Some Objective-C objects are really CF types behind the scenes.

- This is how toll-free bridging works.

Cocoa

Cocoa is a wrapper framework consisting of the Foundation, AppKit, and CoreData frameworks.

Foundation

- A base layer of Objective-C classes.
- Useful primitive object and utility classes.
- The Foundation framework's class hierarchy is rooted in the `NSObject` class.
- Provides `NSString`, `NSDictionary`, `NSUserDefaults`, etc.

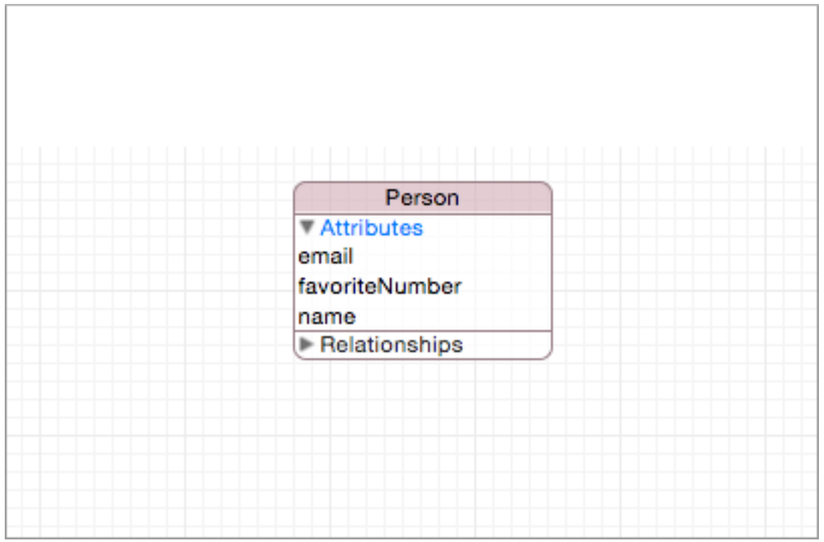
AppKit

- The UI system framework for OS X, (much like `UIKit` for iOS).
- Contains the objects you need to implement your graphical user interface.
  - Windows, buttons, text fields, scrollers, etc.

Cocoa, cont.

CoreData

- An object graph and persistence framework.
- Interfaces with SQLite, but abstracts this from the developer.
- Allows data to be manipulated with high-level objects.



CFPreferences

A thread-safe class for working with application and user preferences.

Is not toll-free bridged to the corresponding Foundation framework class `NSUserDefaults`.

Requires root privileges to modify preferences not owned by the current user.

Retrieving a preference value is done like this in Objective-C:

- `id value = CFBridgingRelease(CFPreferencesCopyAppValue((__bridge CFStringRef)(@“AutoFillPasswords”), (__bridge CFStringRef)(@“com.apple.safari”)));`

…and like this in Python:

- `value = CFPreferencesCopyAppValue("AutoFillPasswords", "com.apple.safari")`

NSApplication

Manages the main event loop and application resources.

- Every app has one instance of this class, accessible via the `NSApp` global variable.
- The `main.m` file generated by Xcode returns the `NSApplicationMain` method, which never actually returns.
  - The `exit` function is called to terminate the process.
- The NSApplicationMain method (mostly) does the following:
  - Creates an instance of `NSApplication` by calling `sharedApplication`.
  - Loads the 'MainMenu' nib.
  - Starts the run loop.

NSFileManager

NSFileManager is a high-level API for interacting with the file system.

- Allows you to create, remove, and examine the contents of the file system.
- Also allows you to create symbolic and hard links, as well as retrieve file attributes such as creation and modification date.
- It can also be used to manage items in an iCloud container using the `[Uu]biquit(y|ous)` methods.

NSTask

Allows you to run another program as a subprocess of your main program.

- Gives you the ability to monitor the external program's standard output and standard error.
  - You can also capture the termination status and reason from the subprocess.
- Differs from `NSThread` in that memory is not shared between your application and the subprocess.
- Useful for creating GUIs for command-line tools.
- Similar to the `subprocess` module in Python.

NSURLSession

Provides an API for working with HTTP requests.

- GET, POST, DELETE, etc.
- Creates a series of... sessions! Each session coordinates a group of data transfer tasks.
- There are three types of sessions:
  - default: Similar to `NSURLConnection`, (i.e., sessions without benefits).
  - ephemeral: Nothing is cached to disk. Good for applications such as streaming audio and video.
  - download: Stores results in a file and continues downloading even if your app quits or crashes.

NSUserDefaults

Provides an API for working with the defaults (preferences) system.

Similar in scope to its `CF` counterpart, `CFPreferences`, but is a much simpler Objective-C equivalent.

Is the preferred way of setting and retrieving your own application's preferences.

- Can also be used to retrieve another app's preferences with the `…persistentDomainForName:` method.

PyObjC

colons -> underscores

Pros

- Allows Python programmers to take advantage of OS X system toolkits.
- Python is powerful and easy to pick up.

Cons

- Probably fewer answers on Stack Overflow.
- Can be awkward to use.
  - All colons following selectors are replaced with underscores, which can make the intent more ambiguous.
    - `[myCar refuelWith:gas replaceCap:@YES];`
    - `myCar.refuelWith_replaceCap_(gas, True)`

Demo 1

System Frameworks in Python

Quartz

```
def is_computer_locked():
    '''
    Quartz.CGSessionCopyCurrentDictionary()
    will return None if no UI session exists
    for the user that spawned the process,
    (e.g., root).

    :returns: True if the screen is locked.
    '''
    ret = False
    if os.geteuid() == 0:
        console_user = get_console_user()
        cmd = '/usr/bin/python -c \'import Quartz; print Quartz.CGSessionCopyCurrentDictionary()\''
        d = subprocess.check_output(['/usr/bin/su',
                                     console_user,
                                     '-c',
                                     cmd])
        not_on_console = 'kCGSSessionOnConsoleKey = 0'
        screen_is_locked = 'CGSSessionScreenIsLocked = 1'
        if not_on_console in d or screen_is_locked in d:
            ret = True
    else:
        d = Quartz.CGSessionCopyCurrentDictionary()
        if d.get('kCGSSessionOnConsoleKey') is False or d.get(
                'CGSSessionScreenIsLocked') is True:
            ret = True

    return ret
psu_demo.py                                                    146,1          58%
```

NSWorkspace and NSApplication

```
def block_user():
    '''
    Kiosk Mode

    Set Application Presentation Options.
    '''
    ws = NSWorkspace.sharedWorkspace()
    ws.hideOtherApplications()

    # OR'ing the desired options yields the correct result.
    options = NSApplicationPresentationHideDock | \
        NSApplicationPresentationDisableProcessSwitching | \
        NSApplicationPresentationDisableForceQuit | \
        NSApplicationPresentationDisableSessionTermination

    NSApp().setPresentationOptions_(options)


def unblock_user():
    '''
    Revert Application Presentation Options to the default.
    '''
    options = NSApplicationPresentationDefault

    NSApp().setPresentationOptions_(options)


psu_demo.py                                                    47,0-1         18%
```

CFPreferencesCopyAppValue

```
def get_pref_val(key, domain):
    '''
    Returns the preference value for the specified key
    and preference domain.

    :param key: The preference key to get.
    :param domain: The preference domain to search.
    :returns: The preference value.
    '''
    if os.geteuid() == 0:
        console_user = get_console_user()
        cmd = '/usr/bin/python -c \'import CoreFoundation; print CoreFoundation.
CFPreferencesCopyAppValue("%s", "%s")\'' % (
            key, domain)
        val = subprocess.check_output(['/usr/bin/su',
                                       console_user,
                                       '-c',
                                       cmd])
    else:
        val = CoreFoundation.CFPreferencesCopyAppValue(key, domain)

    return val
```

CFPreferencesSetAppValue

```
def set_pref_val(key, data, domain):
    '''
    Sets the preference value (data) for the specified key
    and preference domain. It can set nested defaults such as:

    :param key: The preference key to set.
    :param data: The data which should be set for the specified key.
    :param domain: The preference domain which should be updated.
    '''
    if os.geteuid() == 0:
        console_user = get_console_user()
        cmd = '/usr/bin/python -c \'import CoreFoundation; CoreFoundation.CFPreferencesSetAppValue("%s",
"%s", "%s")\'' % (
            key, data, domain)
        subprocess.check_output(['/usr/bin/su',
                                 console_user,
                                 '-c',
                                 cmd])
    else:
        CoreFoundation.CFPreferencesSetAppValue(key, data, domain)

    # Synchronize defaults
    CoreFoundation.CFPreferencesAppSynchronize(domain)
```

CFPreferencesAppValueIsForced

```
# Check if the 'askForPassword' preference is managed.
ask_for_password_managed = CoreFoundation.CFPreferencesAppValueIsForced(
    'askForPassword',
    'com.apple.screensaver'
)
print_color('Managing the screensaver askForPassword preference:')
print('\t%s' % ask_for_password_managed)
```

NSWorkspace

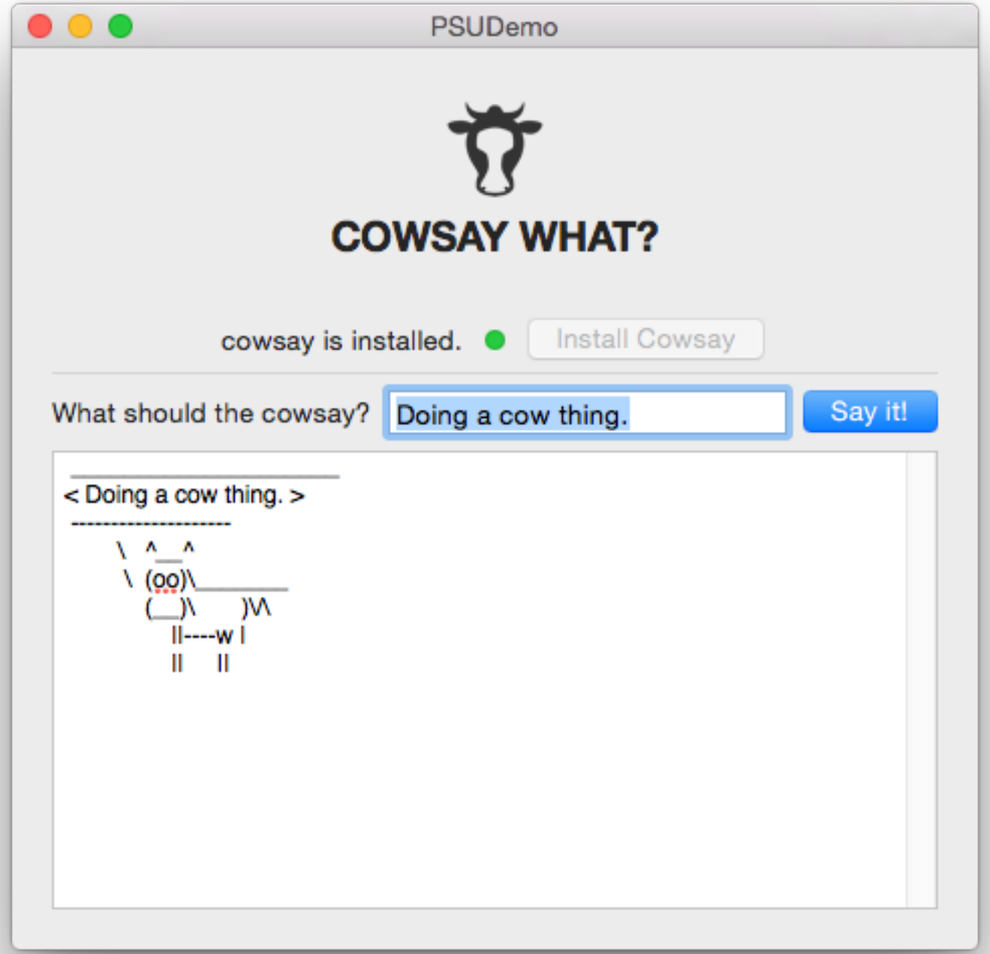
```
def get_running_apps():
    ws = NSWorkspace.sharedWorkspace()
    running_apps = ws.runningApplications()
    apps = []
    for app in running_apps:
        apps.append(app.localizedName())
    apps = set(apps)

    return apps
```

Demo 2

System Frameworks in Objective-C

What we'll be building.

!usefulForSystemAdministrators



What We Covered

Basic usage of CFPreferences, AppKit, and Quartz in Python.

- How to retrieve and set preference values.
- How to determine if an application preference value is forced.
- How to access the kiosk mode APIs and set presentation options.
- How to determine if a Mac’s screen is locked or if a console user is logged in.

Basic usage of NSTask, NSURLSession, NSUserDefaults, etc.

- How to make a cow app.

References

- https://developer.apple.com/library/mac/documentation/CoreFoundation/Conceptual/CFPreferences/Concepts/BestPractices.html
- https://developer.apple.com/library/mac/documentation/CoreFoundation/Reference/CFPreferencesUtils/
- https://developer.apple.com/library/mac/technotes/KioskMode/Introduction/Introduction.html
- http://www.gnu.org/software/gnustep/resources/documentation/Developer/Gui/Reference/NSApplication.html
- https://developer.apple.com/library/mac//documentation/Cocoa/Reference/ApplicationKit/Classes/NSApplication_Class/index.html
- http://www.cocoawithlove.com/2009/01/demystifying-nsapplication-by.html
- https://developer.apple.com/library/prerelease/ios/documentation/Cocoa/Reference/Foundation/Classes/NSFileManager_Class/index.html
- http://nshipster.com/nsfilemanager/
- https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSTask_Class/
- https://developer.apple.com/library/ios/documentation/Foundation/Reference/NSURLSession_class/
- https://developer.apple.com/library/prerelease/ios/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/index.html
- http://pythonhosted.org/pyobjc/
- http://en.wikipedia.org/wiki/Core_Foundation
- http://ridiculousfish.com/blog/posts/bridge.html
- https://developer.apple.com/library/ios/documentation/CoreFoundation/Conceptual/CFDesignConcepts/CFDesignConcepts.html
- https://developer.apple.com/library/prerelease/ios/documentation/CoreFoundation/Reference/CoreFoundation_Collection/index.html
- https://developer.apple.com/library/mac/documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemFrameworks/SystemFrameworks.html

Questions?

So Long, and Thanks for All the Cows