# ADVANCED DEBUGGING
# &
# BEST PRACTICES

with **XCode 4 & Friends**


Instructor:

**Kendall Helmstetter Gelner**

@kendalldevdiary

Kendall.Gelner@KiGiSoftware.com


materials found:

https://github.com/KiGi/AdvancedDebuggingCode

# WHO AM I

- Full-Time iPhone developer since release of SDK.
- Independent consultant for iOS development.
- Over two decades programming experience from corporate to end user.
- Over 30000 Reputation on StackOverflow

# THINGS OF NOTE

- All files included on GitHub:

  **https://github.com/KiGi/AdvancedDebuggingCode**

- XCode tips, tricks and discoveries on twitter

  **@kendalldevdiary**

- Also make sure to watch WWDC videos 415 (Debugging With LLDB) & 409 (Learning Instruments) - this talk builds and adds to many ideas there.

# ON DEBUGGING

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

- Brian Kernaghan

Only when you are debugging somebody else's program do you understand that 90% of programmers are below average.

- Unknown

# THE PHILOSOPHY

Debugging is about giving you whatever paths of visibility into your application you desire, so that you can have enough information for inspiration to strike.

# SCHEDULE

- Morning: Performance Instruments, LLDB
- Afternoon: Analysis Instruments, other tools, CoreData

**Demos for each tool, and Hands-on labs between sections!**

# PERFORMANCE INSTRUMENTS & OVERVIEW

# SIMULATOR FEATURES

- The simulator has some helpful debugging features added over time:

  Slow Animation

  Graphics Blending

  Location

# INSTRUMENT: CORE ANIMATION

- Brings all of the blending tools to the device.
- Can use this to examine other apps.
- Also helpful to check framerates.

# INSTRUMENT: TIME PROFILER

- Great way to check why an app is busy - or why it's not.
- Helps you focus on worst performance first.
- Many helpful ways to look at results.

# LAB TIME

## 15 MIN

- Explore Core Animation effects on your favorite app
- Use time profiler and explore time used when scrolling debugging app (or your own)!
-

# ADVANCED LOGGING

# BETTER LOGGING

- NSLog always outputs, even for customers.
- Evaluating data for NSLog can slow things down.
- Instead - consider DNSLog, defining away for AppStore builds.
- debugDescription can also be implemented for extra data.
- Consider using block for expensive log calculations

# LLDB

# LLDB - THE TIME IS NOW

- Not just because Apple says so, but because it is better now.
- Printing things works like you expect (my.value works!)
- Far more powerful expression use.
- Conditional breakpoints work!
- More extensible also (for those that know Python)
- Old GDB commands mostly work.
- Choose debugger in Schemes

# LLDB - EXPRESSIONS

- Expressions now compile real code to run in app.
- Access with "expr" command (GDB used "p" or "po" to evaluate epxressions)
- Use with continued breakpoints for real-time patching!
- Call with expression --unwind-on-error 0 -- to allow breakpoints to be hit in called code.

# DATA FORMATTERS

- Use to provide better view on your local variables.
- Can define in XCode, .lldbinit or just type it in:

```
type summary add -s "value=${var.size%s}"
<TYPENAME>
```

- With LLDB, these generally work!
- If LLDB does not "see" a class, try using NSClassFromString.

# DATA FORMATTERS

- Can find examples of system formatters (in XCode 4.5):

  XCode.app/Contents/SharedFrameworks/
  LLDB.framework/

  Resources/Python/lldb

  /formatters/objc

# LAB TIME

## 20 MIN

- Add DNSLog to DebuggingApp from extra files, convert uses of NSLog.
- Use EXPR when paused to try modifying an existing value (like a cell label).
- Add a custom formatter for KGFlickrItem and examine values as the cells are formatted

# BREAKPOINTS

# CONDITIONAL

- For any expression used in a breakpoint, make sure the command line accepts it first.
- Conditional breakpoints can use any expression (start with expr as before).
- With an auto-continue breakpoint, you can add small blocks of expr code to try adding logic on the fly.
- Note you'll have to cast a LOT, pretty much every term.
- Logging expressions really only work with numbers

# DEBUGGER AS AUDIO PROBE

- Covered in WWDC video, but there is a better why...
- Understand relations between calls in an API, or the flow of data in your app.
- Also, speech (numbers only...)!
- You can use your own sounds placed in ~/Library/Sounds

# LAB TIME

## 20 MIN

- Use EXPR on a continue breakpoint in cellForRow to strip out extra text on flickr user name output.

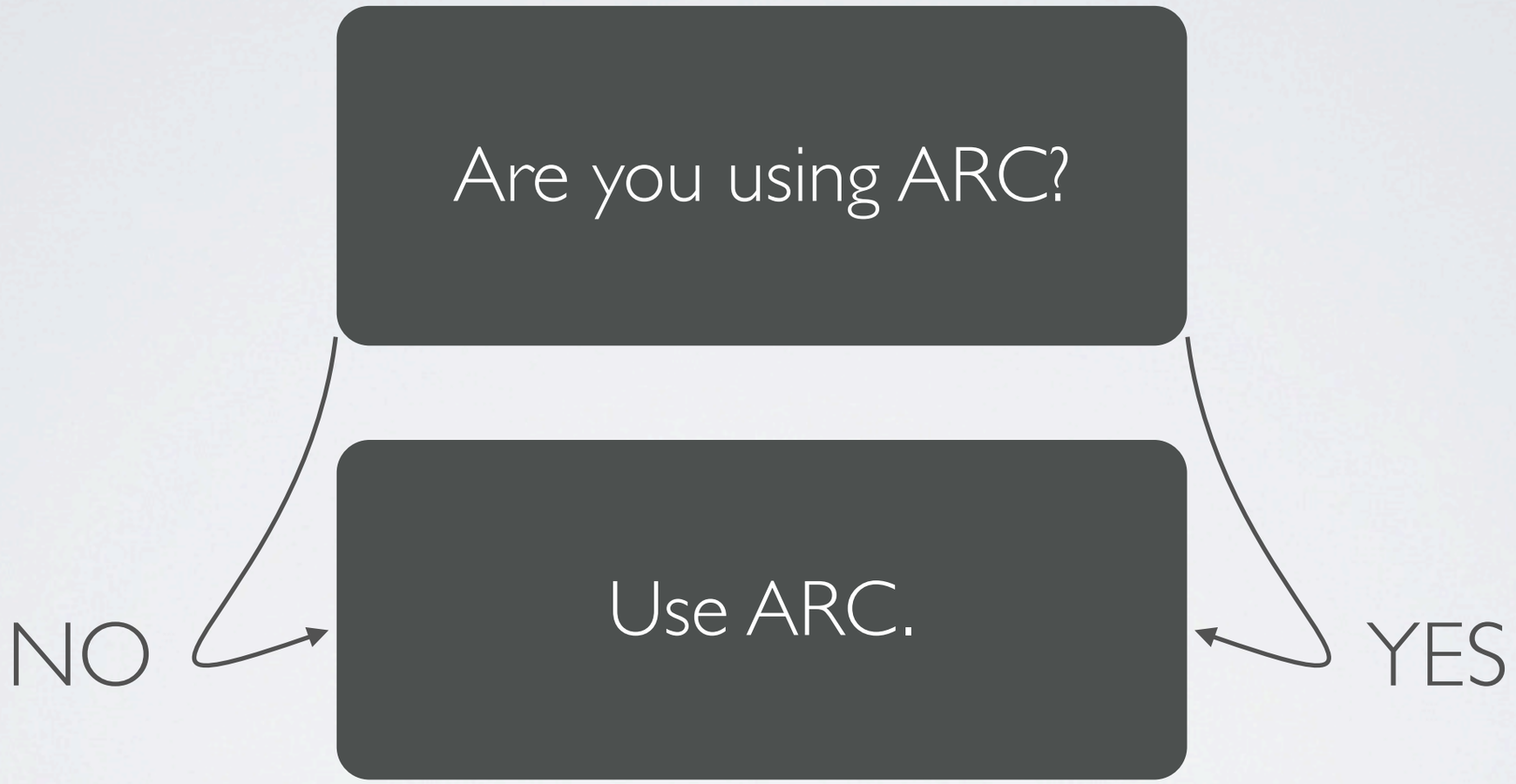
- Add an audio breakpoint and explore when cellForRow is called, vs. heightForRow.

- Successfully have the debugger stop at a cell when the tag count is greater than 10, and speak the index.

# LUNCH - BACK AT 1:00PM

# INSTRUMENTS FOR ANALYSIS

# GET ON THE ARC NOW



# ARC CONVERSION ISSUES

- If you are using CF objects, try __bridge cast first to fix.
- Replace autorelease pools with @autorelease
- For anything really odd (or third party libraries), you can opt not to convert per file.

# LEAKS VS CLOGS

- ARC prevents common memory issues, but not all!
- Leaks are like a murder mystery.
- You can also "clog" memory with memory you didn't know you were still using.
- Use Leaks first, then Object Alloc instrument with Heapshot.

# INSTRUMENT REGIONS

- You can flag points of interest in Instruments.
- How to automatically flag points of interest?
- Custom instruments with DTrace!

# ZOMBIE HUNT

- Use the Zombie instrument to find dead objects that cause crashes.

# LAB TIME

## 20 MIN

- Create a custom instrument to track awakeFromNib, try various controls.
- Use Object Alloc and try clicking on Flickr items then traversing other users.
- Do a leak check when opening Flickr items and closing.
- Run the "Modern ObjC Converter" (right by Arc convert) and see what it wants to do.

# A PONY FOR EVERYONE!

# PONY DEBUGGER

- Included in your "Extra" folder
- Add project : libraries : linker flag : header path. (PonyAdd.txt)
- Install XCode command line tools.
- Install & run the server (PonyDebuggerQuickstart.txt)
- Then add code to attach to Pony server.
- Experiment, some things may not work.

# NETWORK LINK CONDITIONER

- Now download through XCode:

  XCode >Open Developer Tool > More Developer Tools

  Look for "Hardware IO Tools" package (added to "Extra")

- Affects all the apps on the system!

- Existing settings are OK, but you can make a better version of bad 3G...

# CHARLES

- Also in "Extra" folder.
- This one costs money, but is worth it.
- Free trial for quite a while.
- Great reporting of all the traffic details you could wish.
- Set HTTPS hosts to proxy traffic.
- Make sure a "curl" or browser call works before you try to code against something or look for bugs.

# IPHONE CONFIGURATION UTILITIY

- Download from web (Google for it).
- Can be used to quickly switch to proxy server for traffic examination.
- Can have multiple profiles installed, with only some active.

# LAB TIME

## 20 MIN

- Install PonyDebugger in the sample app, and try browsing the Flickr entries there.
- Install Charles and examine traffic when Flickr is called.
- Try installing a profile on your device to go through the Charles proxy (your IP / port 8888 by default)

# BEST PRACTICES

# COMMON APPROACHES

- Carefully set up app structure first.
- Make good use of real folders to keep file separation.
- Keep a common naming scheme.
- Make sure to use refactoring to rename classes and properties.

# CODING THINGS

- Make sure to consider where app data is stored, use no-backup attribute as needed:

```
const char* attrName = "com.apple.MobileBackup";
u_int8_t attrValue = 1;

int result = setxattr(filePath, attrName, &attrValue, sizeof(attrValue), 0, 0);
```

- Look for libraries before coding something complex.

- Keep things simple first, then use Time Profiler to find what needs to be more complex.

# BREAK - 15 MIN

# CORE DATA DEBUGGING

- Instruments can help track saves and other operations.
- Also just before a save, look at ManagedObjectContext updatedObjects or insertedObjects to see what will be saved.
- Output from Core Data objects in debugger truncates strings! Your full string is there.
- Use valueForKey: to get properties out of a Core Data object in the debugger.

# CORE DATA STRUCTURE

- Be careful of deletes, very dangerous.
- isDeleted does not really mean isDeleted.
- Inheritance means flat tables.
- Make everything you can optional, validations dangerous.
- Helps to make use of helper library like Magical Record.
- Always create model with more than one version to start.
- Create new model version for every app store release.

# CORE DATA FILES

- Keep in caches.
- Auto migrate, destroy if that doesn't work.
- iCloud key/value pretty stable, Core Data documents fiddly.
- You can pro-load databases, copy them into writable directory on app start.
- You can copy DB from the simulator directories to keep different cases around (as long as you have the same version) or transplant databases from device.

# OPEN LAB

# THANKS FOR ATTENDING!


Instructor:

**Kendall Helmstetter Gelner**

@kendalldevdiary

Kendall.Gelner@KiGiSoftware.com


materials found:

https://github.com/KiGi/AdvancedDebuggingCode