Sharing Models Between Android and iOS

Mark Schisler
STL Mobile Dev

Motivation

- Do it once if you can.

Mechanics

- iOS
  - Static C++ library, *.mm
- Android NDK
  - Android Make System
  - JNI Wrappers => C++
  - Shared Library C++ library

Resources

- boost
- STL
- CoreFoundation (Lite)
- sqlite

Demo

Strengths

- One set of tests
- One initial coding
- One set of logic to maintain
- Less bugs? time?
- iOS implementation is easy

Weaknesses

- File system differences
- Android is complicated
- Lack of IDE refactoring abilities
- Compiled languages increase dev time
- JNI complicates interface changes

Objectively speaking

- Not for every situation
- Benefit characteristics

Before getting started

- The Android NDK will give you problems
  - Read the Android NDK docs first
  - Android NDK examples are helpful
  - Do **NOT** search the Internet for your problem

Best Practices

- Wrap primitive datatypes
- Create helper methods that make it easy
- Use javah and javap to make JNI easier
- Create your own tools to make it easier

Resources

- Debugging Android NDK in Eclipse
- CoreFoundationLite for Android
- JNI Tutorial
- JNI Programmer's Guide
- Boost for Android
- Code from the demo
- Android NDK Docs