Mirah for Android Development

Brendan Ribera - Strange Loop 2011

Hi.

@abscondment
brendan.ribera@gmail.com
http://threebrothers.org/brendan/

I’m Brendan Ribera – abscondment on twitter and Github. There’s my other contact info, too. I fully intend to answer every question you’ve ever had about anything with this talk, but should I fail to do so, please reach out.

History

- Previous job: extending existing Java app.
- New venture: a new app from scratch.

* @Urbanspoon last year - updated Android app.
  * Paid to write Ruby (~exclusively) & using Clojure for personal, Java dev really chafed. Dev driven dev -> Latitude.
  * No sense in rewriting; just extended. New features, WebViews. Mixed source hard, not intractable.
  * Felt safe - could always compile to Java source if things fell apart.
* New venture, app from scratch.
* Both approaches are feasible & we'll touch on them.

.plan

- What is Mirah?
- Why use Mirah on Android?
- Show me.

* What: High level overview of the language, its relation to Ruby and Java/JVM.
* Why: alternatives? Mirah features, goals, benefits.
* Hands-on example. Tools for mirah (pure & mixed source). Walk through & run a toy app.

What is Mirah?

Apparently Mirah is a pop singer from San Francisco.
This is not the Mirah I’m talking about. She will not help you with your Android app. Sorry.
If you want to leave the room, I understand. You can do so discreetly during the next few slides.

What is Mirah?

The Mirah I'm actually talking about is related to this, Indonesia.
 * The Island of Java in particular.
 * Mirah is actually Javanese word that means "Ruby".

What is Mirah?

- Javanese for "Ruby"
- Ruby-like syntax, JVM bytecode.
- No Runtime Dependencies

Monday, September 19, 2011

* Yes, that's a wonderful pun.
* You get to write something that's very close to Ruby. The syntax is very, very similar - the current compiler actually uses a transformation of the JRuby abstract syntax tree.
* It actually emits Java classfiles (or even source files, if that's desired).
* 0 runtime dependencies - unlike JRuby, etc. Important to consider when we think about Android.

A Mingled Monster

- A Chimera
- Ruby with Static Types? Java with Ruby Syntax?

Monday, September 19, 2011

 * What animal for an O'Reilly book cover?
 * I'd go with the Chimera, which Homer described as "A mingled monster of no mortal kind" – goat's body, lion's head, and dragony parts, too.
 * A mixture: Which (Ruby/Java) should you expect? Not 100% of either; still evolving.
 * Syntax typically to Ruby, with Java showing up
Here are some short examples of how the two animals coexist.

```
class Messenger
  def initialize(m:String)
    @message = m
  end

  def message
    @message
  end
end
```

Monday, September 19, 2011

Classes, constructors, instance variables, methods => like Ruby!
 * Here's a really simple, contrived class.
 * You can see it has a Ruby-style constructor and instance variables.
 * The methods don't have explicit return types.
 * But, there is an explicit type for the constructor argument. From that, the compiler can infer...

```
class StrangeHello < Messenger
  def initialize
    super 'Strange Loop'
  end

  def message
    "Hello, #{super}!"
  end
end
```

Monday, September 19, 2011

Subclassing, method overloading, string interpolation => like Ruby!
 * So, that contrived class was a setup for the typical ‘Hello World’ program.
 * All of the typing here is implicit.
 * Runs with:
puts StrangeHello.new.message

```
// Generated from Hello.mirah
public class StrangeHello extends Messenger {
  public  StrangeHello() {
    super("Strange Loop");
  }
  public java.lang.String message() {
    return "Hello, " +
           super.message() + "!";
  }
}
```

Monday, September 19, 2011

Here’s an example of the java source that mirahc generates.
 * Usually you just go straight to bytecode.
 * You’ll notice that this isn’t really idiomatic.
 * It is functional and fairly legible; good for debugging.

```
class Widget
  def name...end
end

interface MakesWidgets do
  def make(name:String):Widget
  end
end
```

Monday, September 19, 2011

Interfaces => not like Ruby!
 * Java: no modules, yes interfaces.
 * Ruby: no abstract classes.
 * Simple: method signatures, note argument types & return types.

```
class WidgetFactory
  implements MakesWidgets

  def make(name)
    w = Widget.new(name)
    # set up the widget
    w
  end
end
```

And then actually implementing the interface is quite easy.
* No type information needed here.

```
widget_list.each do |obj|
  # i is an Object
  puts Widget(obj).name
end
```

Monday, September 19, 2011

Iteration, typing.
 * Generics aren't really supported right now. So when you get a List, Mirah always extracts an Object.
 * You get this nice 'each' block for iterating.
 * Then you can cast to the class you expect. So here we print out the name of each Widget.

Summary

- Compiled
- Bytecode (or Java source)
- Type Inference
- No Runtime Dependencies

Why use Mirah on Android?

http://www.flickr.com/photos/gsfc/4632905990/sizes/l/in/photostream/

Monday, September 19, 2011

Let’s move on to the second topic. Why would you pick Mirah for an Android application?

Alternative Alternatives

Scala	[0]
JRuby/Ruboto	[1]
Clojure	[2]
Groovy/Jython/Lua/Others	[3]

References:
[0] http://devblog.bu.mp/how-we-use-scala-in-bump-for-android - http://lamp.epfl.ch/~michelou/android/
[1] http://ruboto.org/
[2] http://www.deepbluelambda.org/programming/clojure/clojure-for-android-source-published
[3] http://thediscobot.blogspot.com/ - http://code.google.com/p/jythonroid/ - http://code.google.com/p/android-scripting/

Monday, September 19, 2011

We can start by talking about other alternatives to Java.
References in inscrutable font; read them when you download the slides.
* Scala: promising; used at Bump, Thoughtbot, probably others. Runtime? Subclass ‘Map’ => ACs?
* Ruboto: cool; startup slow. Big, required library (no proguard?). Ruboto core!?
* Clojure: very expressive language. Runtime?
* Others: old (last posts in 2009)? SL4A not intended/suited for apps?

Summary

- Scala seems really viable.
- Ruboto and Clojure are cool, but startup times, memory, app size need to improve.
- Goal is not to find a "winner".
- Any could serve a niche. Use what works.

My take on this:
 * None are perfect, but no language really is.
 * Scala viable, others not ready (~7 Ruboto load on my Nexus S).
 * Comparison is not the purpose of this talk. I think we as engineers benefit from a plurality of ideas.
 * This is more about what makes Mirah good in its own right.
 * So here are some reasons one might choose it.

A practical decision

- No runtime dependencies.
- No performance penalties.

There's nothing extra for you to ship with you app. Nothing extra to load at runtime.
You can get exactly the same performance as Java – memory and CPU cycles.

An aesthetic decision

- Some people really dislike Java syntax.
- Some people really like Ruby syntax.
- Fewer type declarations, boilerplate.
- Expressive: blocks, macros, etc.

* I’m one of them.
* Meh.
* This is the good stuff. Clean looking, less ceremony.
* Metaprogramming & Macros: *compile time*
Here are some examples of doing some common Android patterns.

Blocks

```
this.closeButton.setOnClickListener(new
OnClickListener() {
  @Override
  public void onClick(View v) {
    finish();
  }
});
```

```
this = self
@close_btn.setOnClickListener do |v|
  this.finish
end
```

Monday, September 19, 2011

Common in Android: set an action on a button.
You can use an anonymous inner class in Java. Mirah lets you use a block.
 * this = self -> scoping
 * interface with one method can become a block.
 * Code doesn’t need to know anything about the interface!

Blocks

```
Thread t = new Thread(new Runnable() {
  public void run() {
    for(int i = 0; i < 10; i++) {
      System.out.println("At " + i);
    }
  }
});
t.start();


t = Thread.new do
  10.times do |i|
    puts "At #{i}"
  end
end
t.start
```

Common in Android: do some work off of the UI thread.
Again, Mirah can do this syntax with a block. Very succinct.

Macros

```
class NoMacroPerson
  def name_set(name:String)
    @name = name
  end

  def name
    @name
  end

  def age_set(age:Integer)
    @age = age
  end

  def age
    @age
  end
end
```

```
class Person
  [...]
  attr_accessor :name, String
  attr_accessor :age, Integer
end
```

Suppose you have a Person with a name and an age. Typically, you'd create these horribly repetitive getters/setters.
 * NB: the foo_set postfix allows for self.foo = 'bar'
What if you could just do it like this – Ruby style (i.e. attr_reader/writer/accessor)?
Well, you can.

Macros

```
macro def attr_accessor(name_node, type)
  name = name_node.string_value

  quote do
    def `name`
      @`name`
    end

    def `"#{name}_set"`(value:`type`)
      @`name` = value
    end
  end
end
```

Monday, September 19, 2011

Obviously, this code is longer and more complex than 2 getters and 2 setters.
Stick it in an interface (no methods) – you can reuse it in any subclass that ‘implements’ that!
Define & reuse. Let’s you make some really powerful and expressive code.

Show me.

http://travel.charmingasian.net/2010/11/java-island-indonesia.html

Monday, September 19, 2011

Let’s get our hands dirty.

Get Android

- http://developer.android.com/sdk/index.html

```
brendan@flask:~$ brew install android-sdk #osx/homebrew

# note install directory - mine was:
# /usr/local/Cellar/android-sdk/r12

# add ‘bin’ and ‘platform-tools’ to $PATH

brendan@flask:~$ android
# install desired packages. We want at least API 10 rev 2.
```

Monday, September 19, 2011

Prerequisites:
You need the Android SDK. Not going to discuss setting that up in depth, refer to this slide later.

Get Mirah

```
brendan@flask:~$ rvm install jruby
brendan@flask:~$ rvm jruby
brendan@flask:~$ gem install mirah
brendan@flask:~$ gem install pindah
```

Current state-of-the-art:
 - jruby-1.6.4
 - mirah 0.0.9
 - pindah 0.1.2

You need JRuby to install mirah.
 * I use rvm, but all you really need is a copy of JRuby 1.6.x. Newest ones are nice and fast.
 * Mirah is a gem (compiler is written in JRuby, depends on bitescript).
 * Pindah is a build tool. We'll talk about it in a minute.
Again, you should be able to figure this out pretty quickly.

Mixed Source

Monday, September 19, 2011

I’m going to touch on tools mixed-source builds briefly.
* Our toy app is going to be pure Mirah.
 * But this is an important use case -- many existing apps that Mirah could fit into nicely.

Maven

- Plugin: https://github.com/mirah/maven-mirah-plugin
- Pretty easy to set up and use.
- Theoretically, you could couple with the maven-android-plugin...

There's a maven plugin for building Mirah projects. It'll compile your Java, too. Haven't tried to integrate it with Android. Should work, YMMV.

Ant

- Default build tool for Android projects.
- You can get it to do what you want, sorta.

I'm not an ant expert, nor do I want to be.
I used it for the previous Mirah-Java integration I did.

Better Ant

```
<taskdef name="mirahc"
         classname="org.mirah.ant.Compile" />
<target name="javac"
        description="Compiles R.java & gen/ files.">
  <javac srcdir="${gen.dir}" destdir="${classes}"
         includeantruntime="false" failonerror="true" />
</target>
<target name="compile"
        depends="-resource-src, -aidl, javac">
  <mirahc dir="${src}" destdir="${classes}">
    <classpath refid="java.classpath" />
  </mirahc>
</target>
```

What I did originally was...
 * create a target that called out to mirahc – puts out.classes on CP. make compile depend on that
Don't do this; creates 2 JVMs.
Mirah now provides an simple Ant task. You just need to:
 * make the javac call compile your Java source
 * give mirahc those classfiles on its classpath

Problem

- mirahc can't compile Java sources.
- Who gets compiled first?
- Treat one part of your codebase like a lib.

You can go 1 of 2 ways: Mirah can see your Java classes, or Java can see your Mirah classes.
 * I've always done the former - new Mirah treats existing Java like just another library.
 * Complicated. Annoying. But you can manage if necessary.
 * Known issue. Hope that a good solution will emerge in the future.

Pure Mirah

Pindah

- https://github.com/mirah/pindah
- Tools for building Android applications with Mirah
- Works via rake to generate build files and run them.

Monday, September 19, 2011

I actually used pindah master, since it contains some important bugfixes.

```
brendan@flask:~/code$ pindah -h
Usage: pindah create org.example.hello [/path/
to/hello_world] [HelloActivity]
brendan@flask:~/code$ pindah create \
org.threebrothers mirah-guide Guide
```

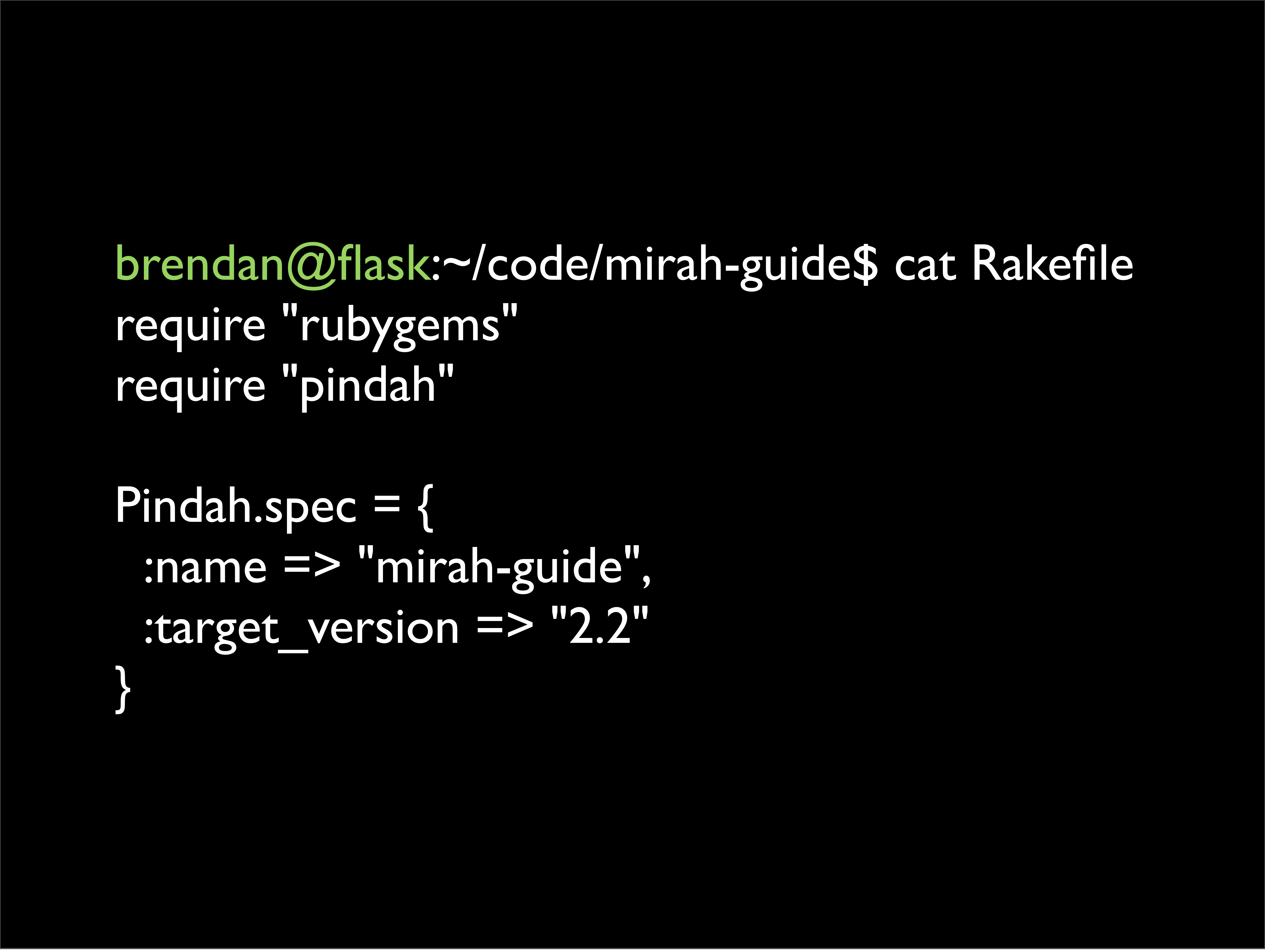
Monday, September 19, 2011

Pindah usage is really simple. You give it a package name and an optional directory/activity. Here’s how I created the toy project.

```
brendan@flask:~/code/mirah-guide$ ls
total 16
-rw-r--r--  1 brendan  staff  714 Sep 18 13:43 AndroidManifest.xml
-rw-r--r--  1 brendan  staff  108 Sep 18 13:43 Rakefile
drwxr-xr-x  7 brendan  staff  238 Sep 18 12:23 bin
drwxr-xr-x  3 brendan  staff  102 Sep 18 12:23 gen
drwxr-xr-x  2 brendan  staff   68 Sep 14 15:03 libs
drwxr-xr-x  7 brendan  staff  238 Sep 18 13:43 res
drwxr-xr-x  3 brendan  staff  102 Sep 14 15:03 src
```

Monday, September 19, 2011

Here's a Pindah project layout. Pretty standard for Android.
If you've done any Andoid work, you'll notice the property files and the build.xml are missing.
Pindah generates these from templates when you invoke rake.
 * This is why Pindah doesn't support mixed-source. Submit a patch! You'll learn a lot.



```
brendan@flask:~/code/mirah-guide$ cat Rakefile
require "rubygems"
require "pindah"

Pindah.spec = {
  :name => "mirah-guide",
  :target_version => "2.2"
}
```

Monday, September 19, 2011

Here’s our Rakefile with some default options: name and target (API level).
I pinned this project on 2.2, which is API level 8 – for AndroidHttpClient.
There are other options, check the README.

```
brendan@flask:~/code/mirah-guide$ rake -T
[...]
rake clean      # Removes output files created by other targets.
rake compile    # Compiles project's .mirah files into .class files
rake debug      # Builds the application and signs it with a debug key.
rake install    # Installs/reinstalls the debug package onto a...
rake javac      # Compiles R.java and other gen/ files.
rake logcat     # Tail logs from a device or a device or emulator
rake release    # Builds the application.
rake spec       # Print the project spec
rake uninstall  # Uninstalls the application from a running emulator...
```

Monday, September 19, 2011

Rake is used to provide the things that Ant uses.
 * You’ll actually see that it’s a shell around Ant, letting you avoid as much XML config.
 * If you want use Ant, easy to copy the Pindah templates and go from there.

Mirah Guide

- Nearby (georeferenced) WikiPedia articles
- Uses the free GeoNames API: http://www.geonames.org

Monday, September 19, 2011

Simple app to show you nearby Wikipedia articles.
 * Grabs your location.
 * First time or you've moved, grabs list from API.
 * Displays summary with Map & browser links.
GeoNames is free; flaky: weird timeout responses.

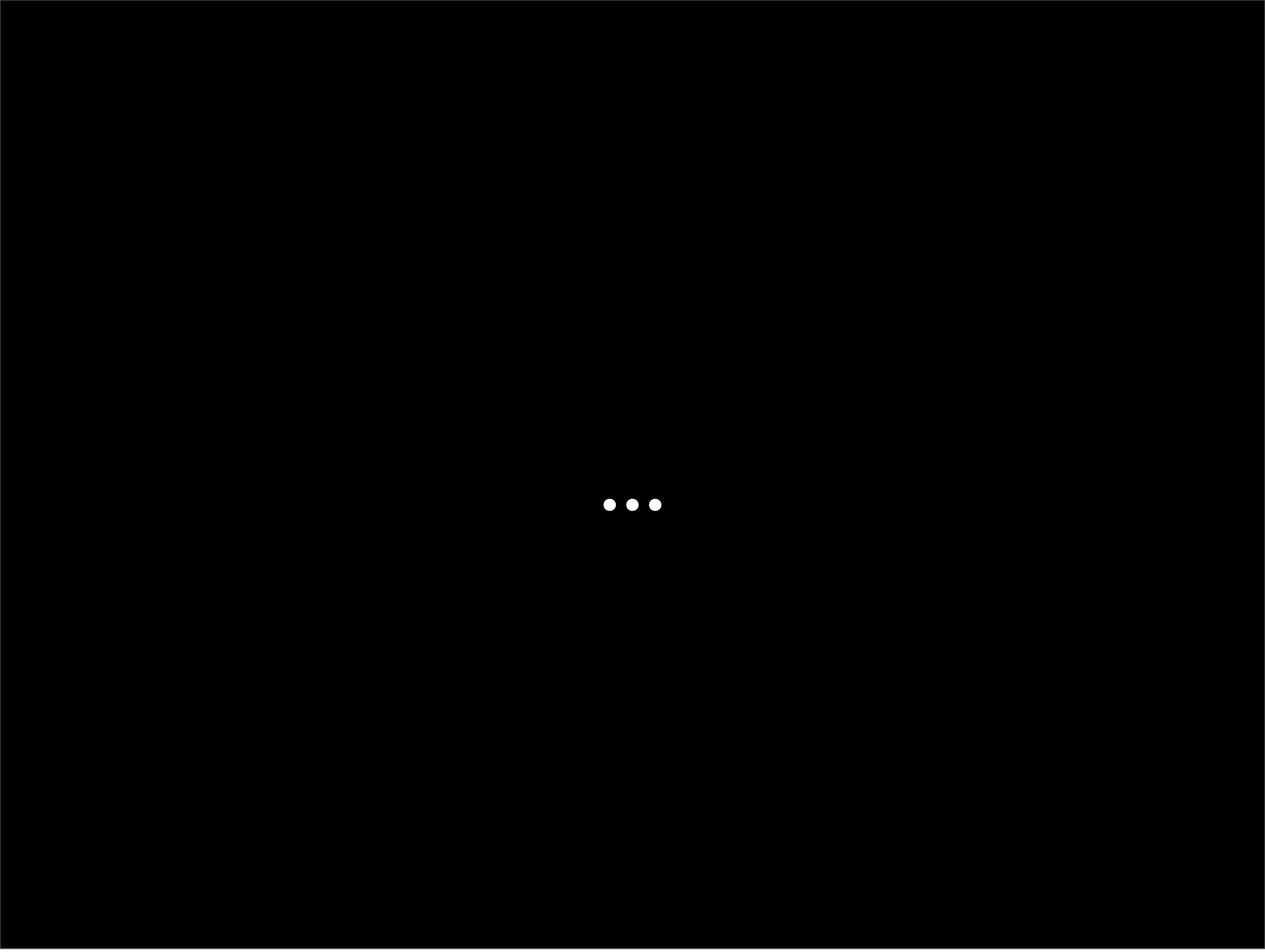
Follow along on Github

- https://github.com/abscondment/mirah-guide
- Look at the tags: `git tag -l`

Monday, September 19, 2011

I threw this together in under 24 hours. Not production-ready... but fun anyway!

If we have time, I’m going to highlight a few interesting parts of the code.
I tagged the project at various stages of development, too. Pretty linear, but could be helpful.



•••

Monday, September 19, 2011

So, here’s the part where I switch over to Emacs. Plan:
 * 000-new-project: look at the default Activity (Guide.mirah)
 * 001-first-button: Added a stupid Button with Toast. Block example.
 * 002-locate-client: Locator - manager, register, interface -> onLocationChanged
 * 003-fetch-json: GeoNamesClient - simple client, GET api, return JSON; Helper - exceptions, tr. Java.
 * 004-display-list: Adapter & Entry; Guide does JSON -> Entry; Adapter does Entry -> View.
 * 005-view-details: Detail - show the summary; buttons -> built-in Intents.

Gps Information: (38.624516, -90.193474)

Questions?