

OSGi on Google Android using Apache Felix

**Marcel Offermans
Karl Pauls**

luminis

Who are we?



Who are we?



- Karl Pauls

Who are we?

luminis



think broad, act bright

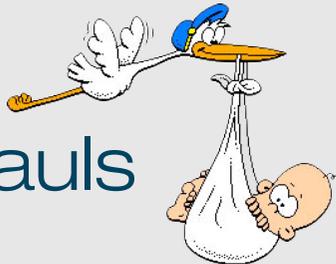
- Karl Pauls
- Marcel Offermans

Who are we?

luminis



think broad, act bright



- Karl Pauls
- Marcel Offermans

Who are we?

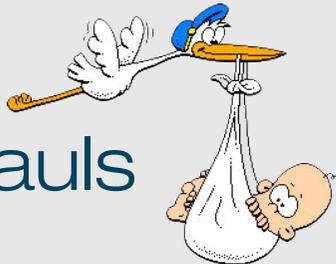
luminis



think broad, act bright



- Karl Pauls
- Marcel Offermans



Who are we?

luminis



think broad, act bright



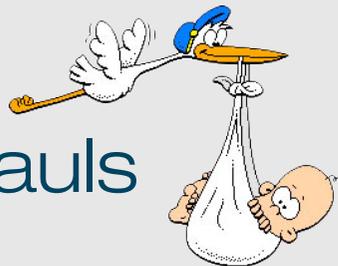
Arnhem



Enschede



- Karl Pauls
- Marcel Offermans



Agenda

- Android
 - Introduction and architecture
 - Hello world demo
- OSGi
 - Introduction
 - Framework and compendium
- Apache Felix on Google Android
 - Getting it to run
 - Creating a dynamic application: paint program

Agenda

- Android
 - Introduction and architecture
 - Hello world demo
- OSGi
 - Introduction
 - Framework and compendium
- Apache Felix on Google Android
 - Getting it to run
 - Creating a dynamic application: paint program

Android

- First SDK release: november 2007
- Android Developer Challenge, \$10M prize money
- Current SDK (M5 RC15): march 2008
- Phones: second half of 2008?

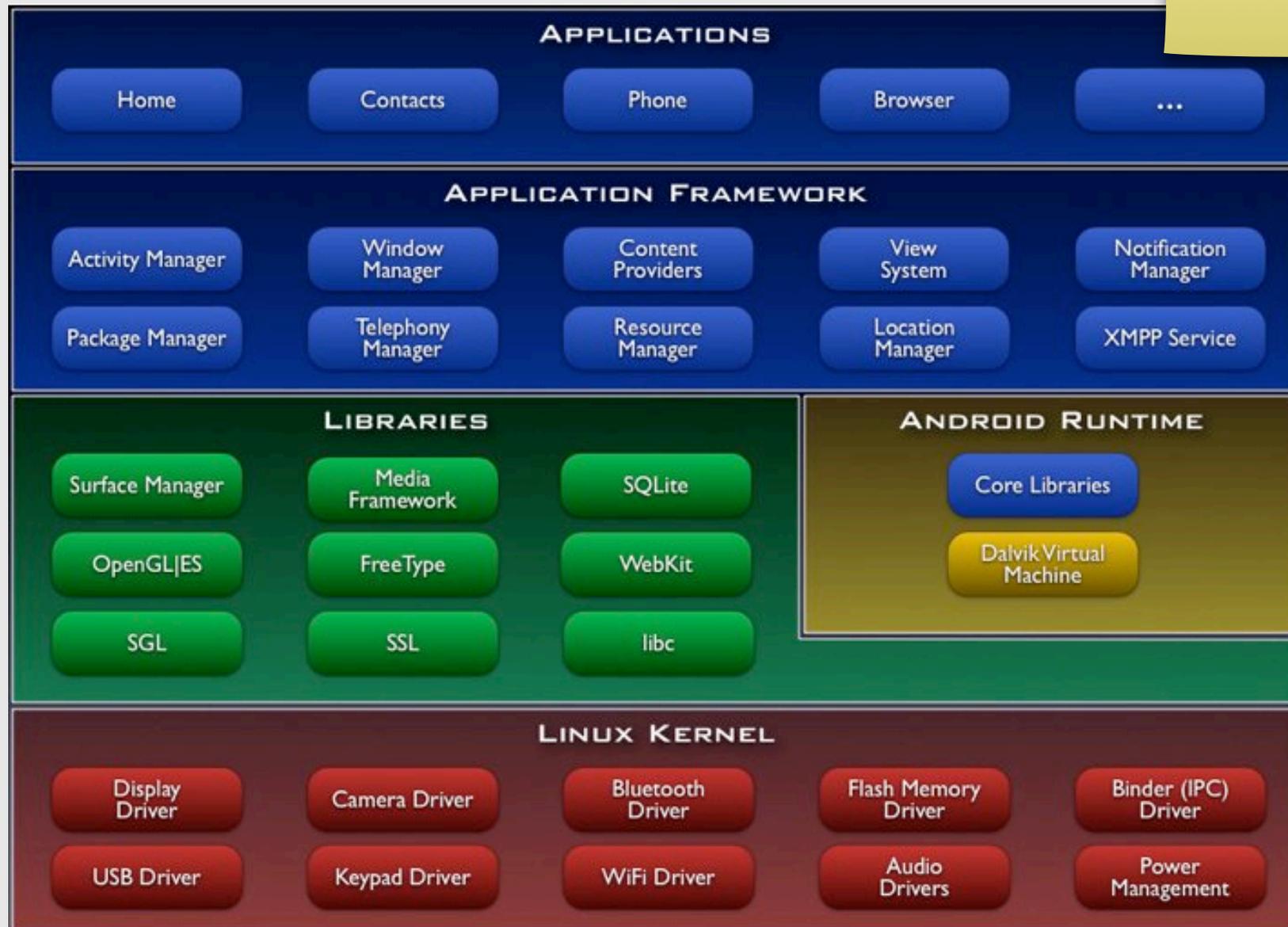


Android

- Device Architecture
- Dalvik Virtual Machine
- From source to deployment
- Anatomy of an application
- Application life cycles

Architecture

This image is rather low-res, so perhaps draw it again...



Dalvik Virtual Machine

- interpreter-only, register based virtual machine
- optimized to run multiple VM instances
- executes files in .dex format
- runs on posix-compliant operating systems
- looks like Java ;)

From source to deployment

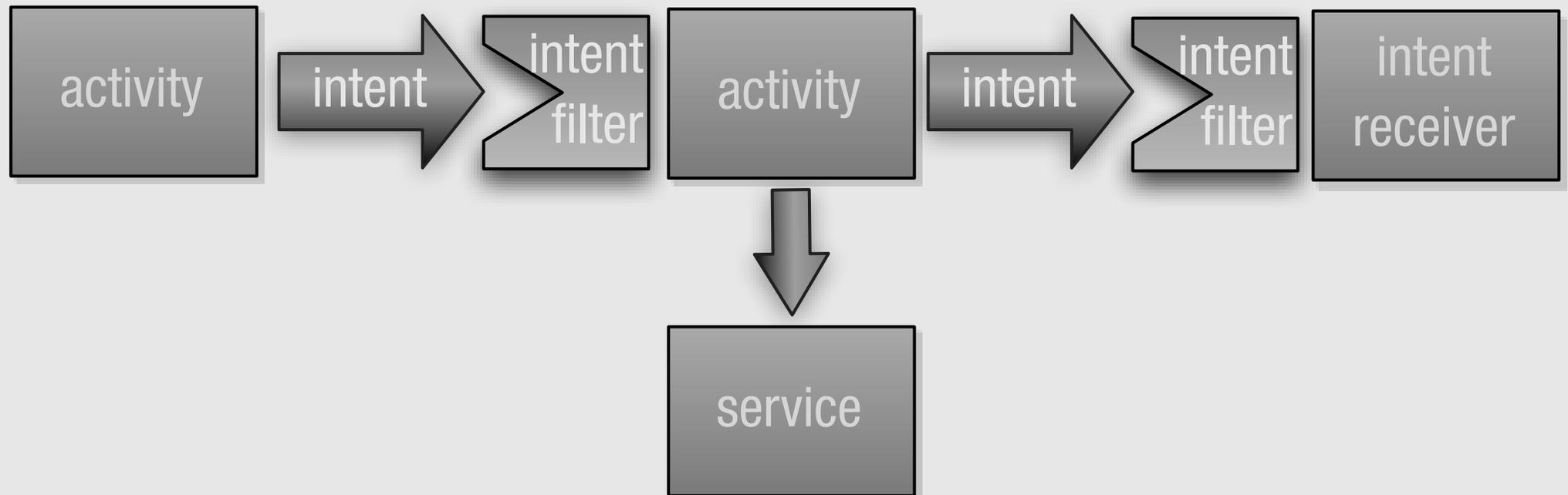


- Eclipse Plugin: Android Development Tools
 - compiles and packages automatically
 - allows you to launch and debug in the emulator
- Command line: `activityCreator.py`
 - generates project structure
 - Ant `build.xml` file, optionally IntelliJ project files

Anatomy

- activity, a single screen
- intent, describes what the application wants done
- intent filter, describes intents that can be handled
- intent receiver, non UI code that reacts to intent
- service, background process with API
- content provider, for shared data access

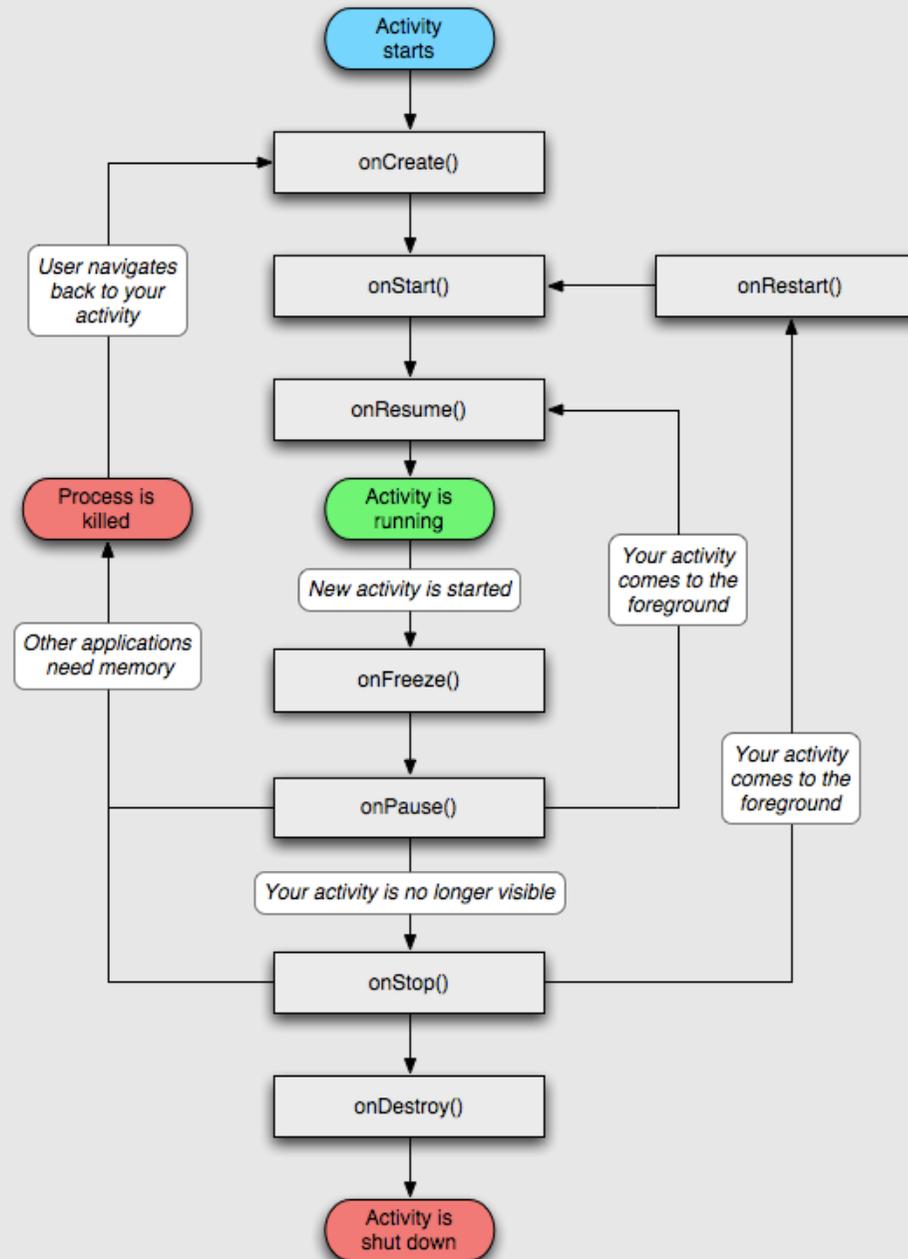
Anatomy Example



Life cycle

- Application life cycle is not controlled by the application itself
- Android maintains an “importance hierarchy” based on the components and their state:
 - foreground process
 - visible process
 - service process
 - background process
 - empty process

Life cycle (Activity)



Hello world demo!

- Create an application with an activity in Eclipse
- Set “hello world” text
- Create a breakpoint
- Deploy and debug the application

Agenda

- Android
 - Introduction and architecture
 - Hello world demo
- OSGi
 - Introduction
 - Framework and compendium
- Apache Felix on Google Android
 - Getting it to run
 - Creating a dynamic application: paint program

OSGi history

- Started as an embedded platform for the “home gateway”
- Originally under the JCP as JSR-8 (1999)
- OSGi alliance, consists of a large number of big companies, with the following mission:
 - Maintaining and publicizing the OSGi specification.
 - Certifying implementations.
 - Organising events.
- Current version: OSGi Release 4.1 (JSR-291)

OSGi today

OSGi technology is the dynamic module system for Java™

OSGi technology is Universal Middleware.

OSGi technology provides a service-oriented, component-based environment for developers and offers standardized ways to manage the software lifecycle. These capabilities greatly increase the value of a wide range of computers and devices that use the Java™ platform.

OSGi Alliance

- Expert Groups:
 - core platform (CPEG)
 - mobile (MEG)
 - vehicle (VEG)
 - enterprise (EEG)
 - residential (REG)
- Working Groups:
 - marketing
 - requirements

OSGi specification

**OSGi Service Platform
Core Specification**
The OSGi Alliance

Release 4, Version 4.1
April 2007



OSGi
Alliance

**OSGi Service Platform
Service Compendium**
The OSGi Alliance

Release 4, Version 4.1
April 2007



OSGi
Alliance

OSGi Framework Layering

SERVICE MODEL

L3 - Provides a publish/find/bind service model to decouple bundles

LIFECYCLE

L2 - Manages the life cycle of a bundle in a framework without requiring the vm to be restarted

MODULE

L1 - Creates the concept of a module (aka. bundles) that use classes from each other in a controlled way according to system and bundle constraints

Execution Environment

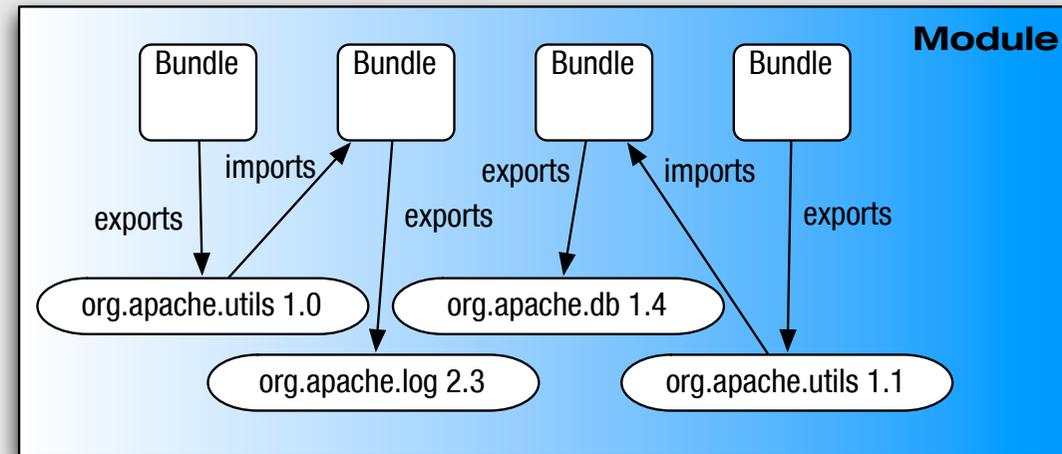
L0 -
OSGi Minimum Execution Environment
CDC/Foundation
JavaSE

Module Layer (1/3)

- Unit of deployment is the bundle i.e., a JAR
- Separate class loader per bundle
 - Class loader graph
 - Independent namespaces
 - Class sharing at the Java package level

Module Layer (1/3)

- Unit of deployment is the bundle i.e., a JAR
- Separate class loader per bundle
 - Class loader graph
 - Independent namespaces
 - Class sharing at the Java package level



Module

Module Layer (2/3)

- Multi-version support
 - i.e., side-by-side versions
- Explicit code boundaries and dependencies
 - i.e., package imports and exports
- Support for various sharing policies
 - i.e., arbitrary version range support
- Arbitrary export/import attributes
 - Influence package selection

Module

Module Layer (3/3)

- Sophisticated class space consistency model
 - Ensures code constraints are not violated
- Package filtering for fine-grained class visibility
 - Exporters may include/exclude specific classes from exported package
- Bundle fragments
 - A single logical module in multiple physical bundles
- Bundle dependencies
 - Allows for tight coupling when required

Module

Life-cycle Layer

- Managed life cycle
 - States for each bundle;
- Allows updates of existing bundles.
 - Dynamically install, start, update, and uninstall

Module

luminis

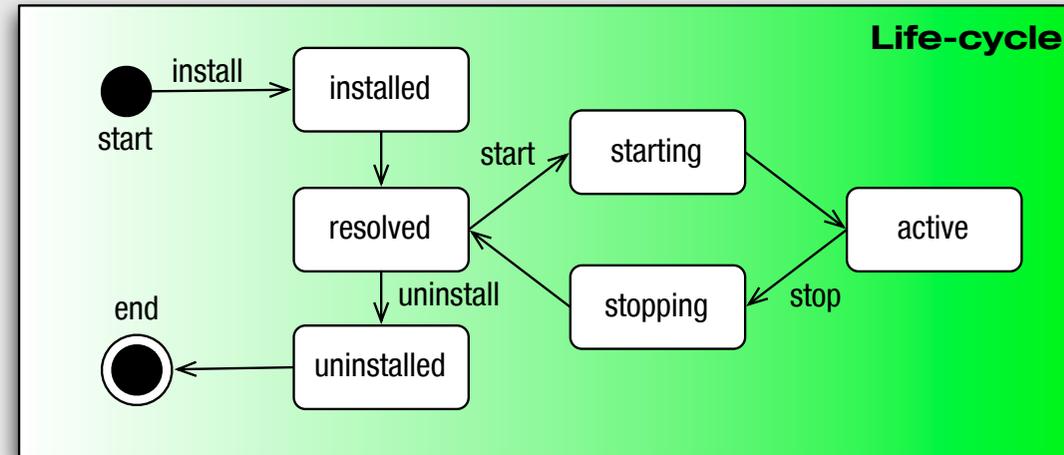
Life-cycle Layer

- Managed life cycle

- States for each bundle;

- Allows updates of existing bundles.

- Dynamically install, start, update, and uninstall

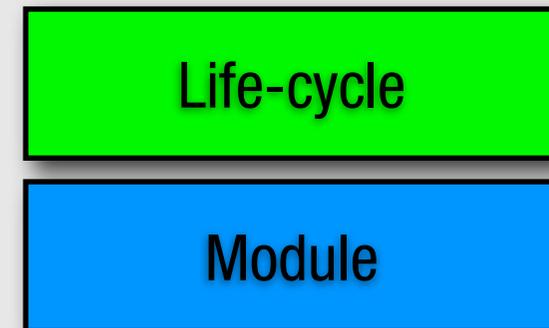
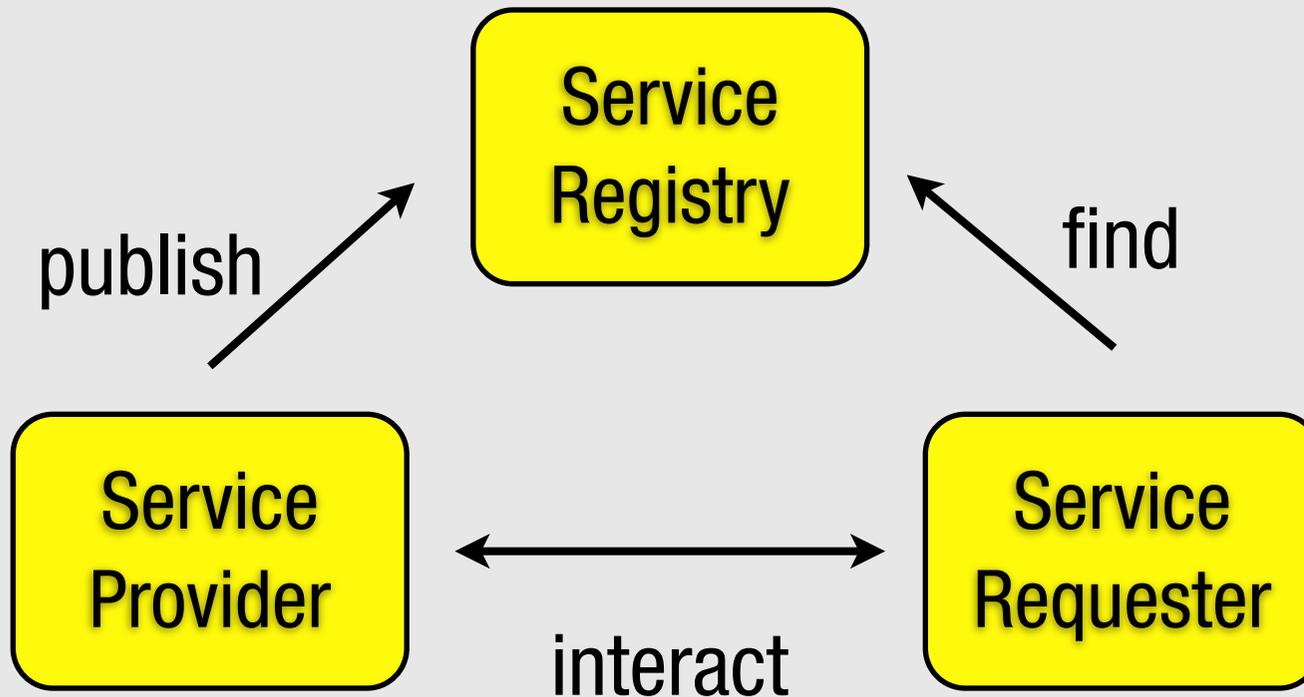


Life-cycle

Module

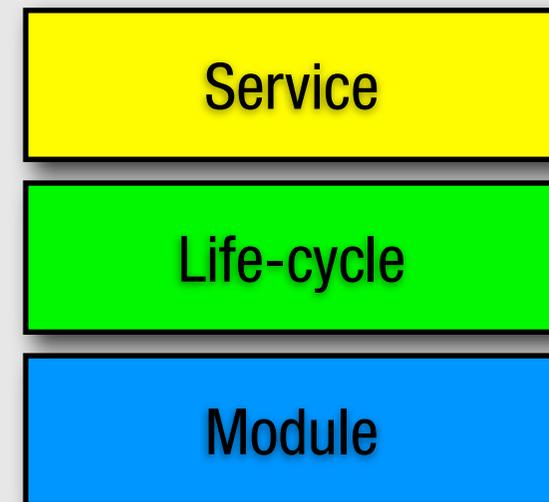
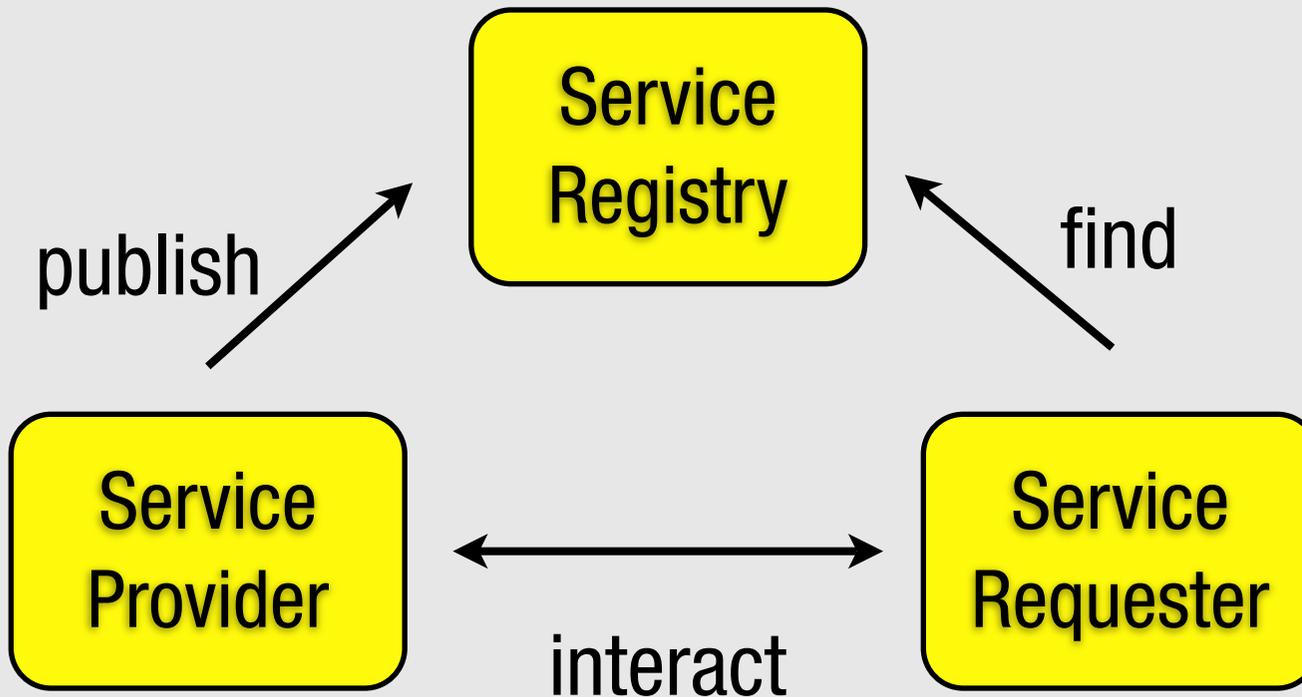
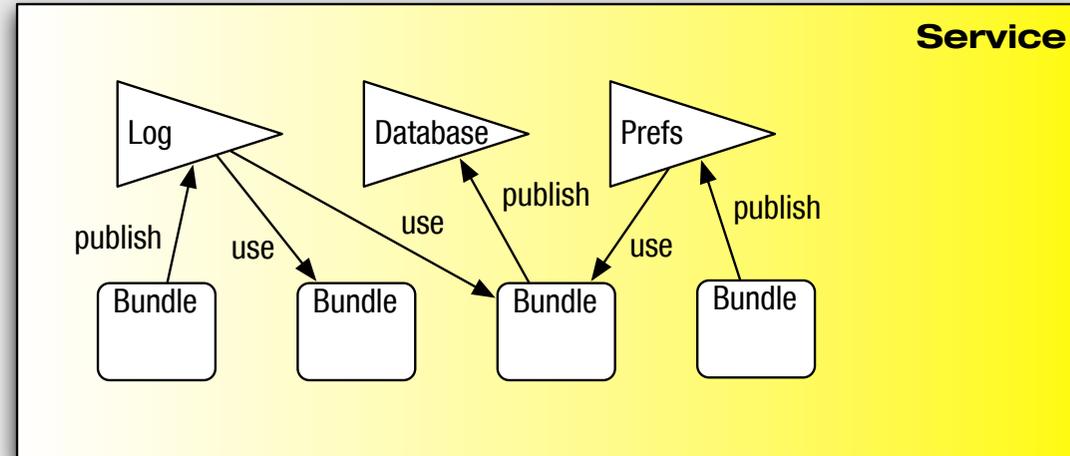
Service Layer

- OSGi framework promotes service oriented interaction pattern among bundles



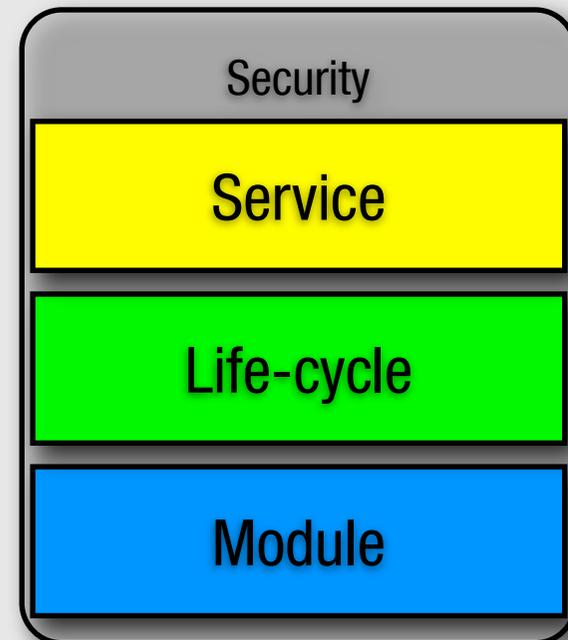
Service Layer

- OSGi framework promotes service oriented interaction pattern among bundles



Security

- Optional Security Layer based on Java permissions
- Infrastructure to define, deploy, and manage fine-grained application permissions
- Code authenticated by location or signer
- Well defined API to manage permissions
 - PermissionAdmin
 - ConditionalPermissionAdmin



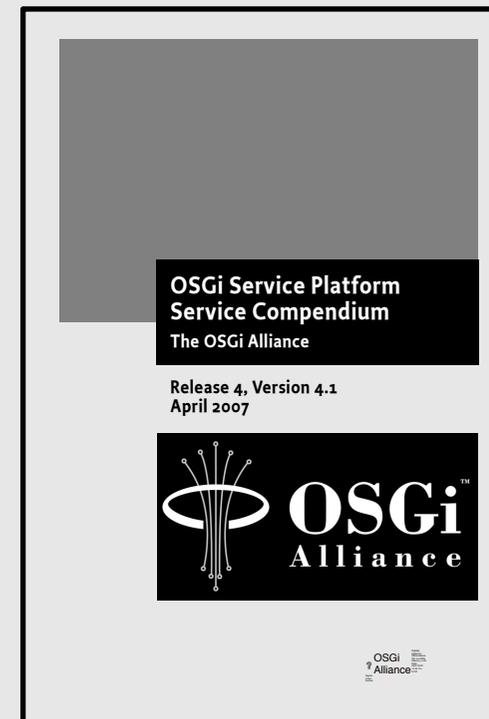
Shameless plug:

if you want to know more about security in
OSGi, come to our talk about
Building Secure OSGi Applications

Leveraging standard services

- Specification:
 - OSGi compendium – catalog of standard service descriptions
- Implementations:
 - OBR repository at bundles.osgi.org – over 1400 bundles, implement compendium and other services
 - Maven repository and third party OBR's
 - More and more projects are made OSGi compatible, for example:
 - Apache Commons OSGi

OSGi compendium



luminis

OSGi compendium

User Admin

Initial Provisioning

Wire Admin

Log

Device Access

XML Parser

Measurement and State

Preferences

UPnP™ Device

Position

Configuration Admin

Metatype

Event Admin

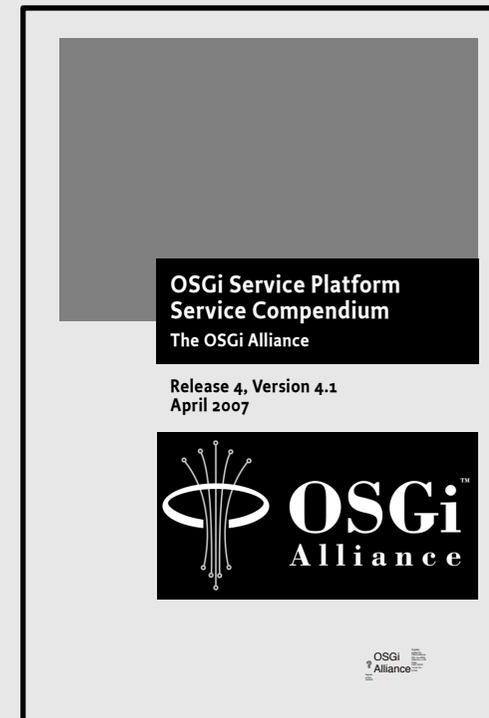
Service Tracker

IO Connector

HTTP

Execution Environment Spec

Declarative Services

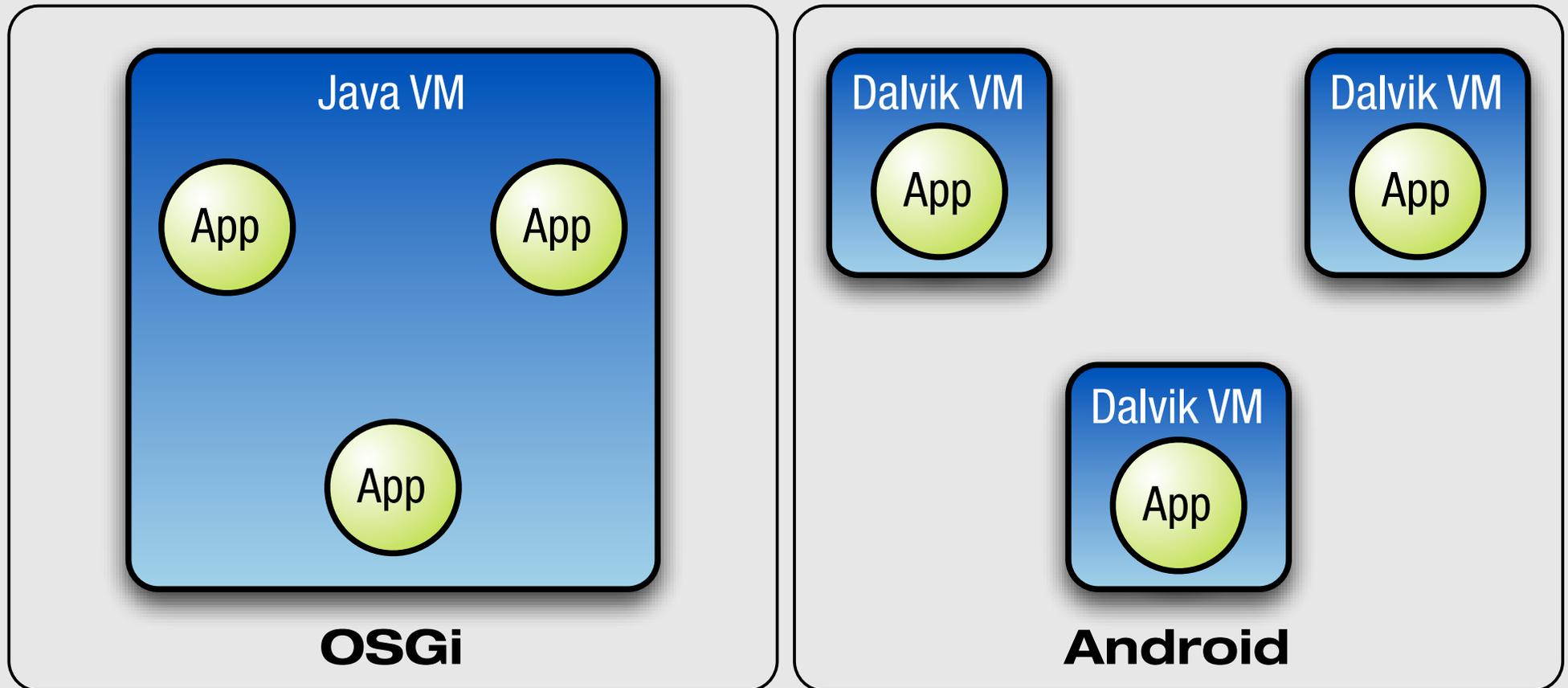


Agenda

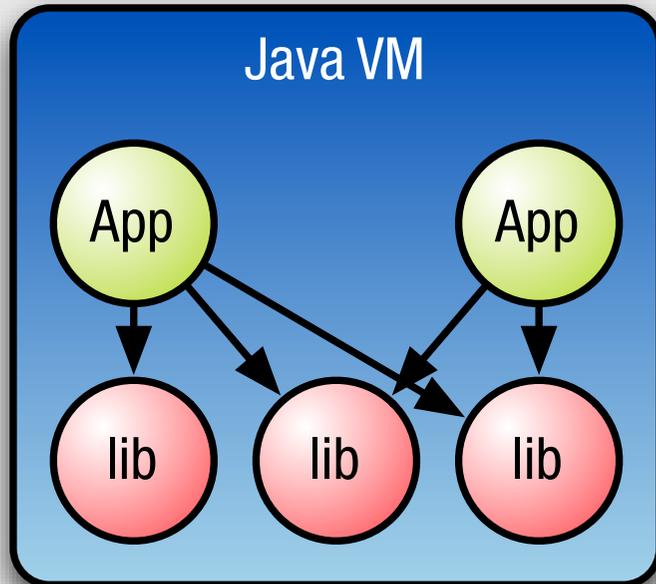
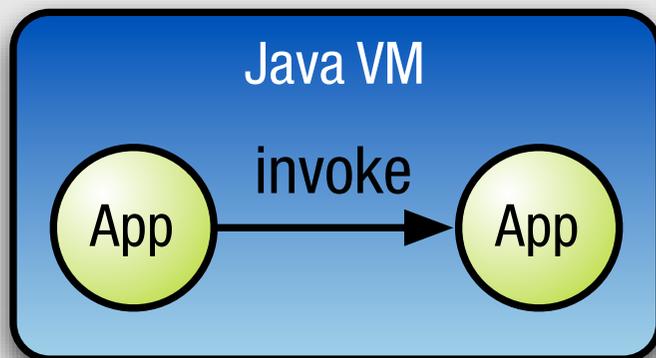
- Android
 - Introduction and architecture
 - Hello world demo
- OSGi
 - Introduction
 - Framework and compendium
- Apache Felix on Google Android
 - Getting it to run
 - Creating a dynamic application: paint program

Why OSGi and Android?

Models are different



Benefits of each model



OSGi



Android

Services in Android

- Must be declared in AndroidManifest.xml
- Can be started and stopped:
Context.startService and Context.stopService()
- You can bind to a service if you want to talk to it
- Services can run in remote processes, in which case there is an Android IDL compiler to generate stubs
 - handles primitives, some collections and Parcelable's by value
 - handles other AIDL interfaces by reference

Getting Felix to run...

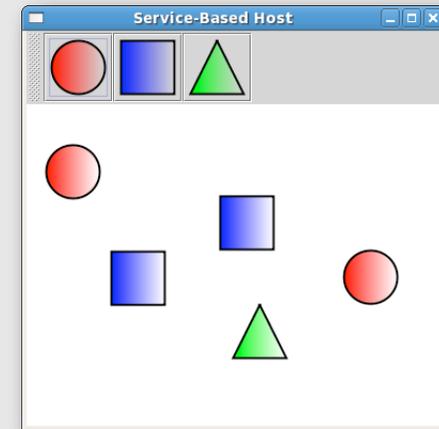
- First step, getting the framework to run
 - Apache Felix is very portable, so we just dex'ed it
 - found a couple of issues, fixed in release 1.0.3
- Second step, dynamically loading bundles
 - the hard part was finding a way to load classes
 - found undocumented internal class
 - **Google, we need an official API for this!**
- For more details:
http://blog.luminis.nl/roller/luminis/entry/osgi_on_google_android_using

...others soon followed

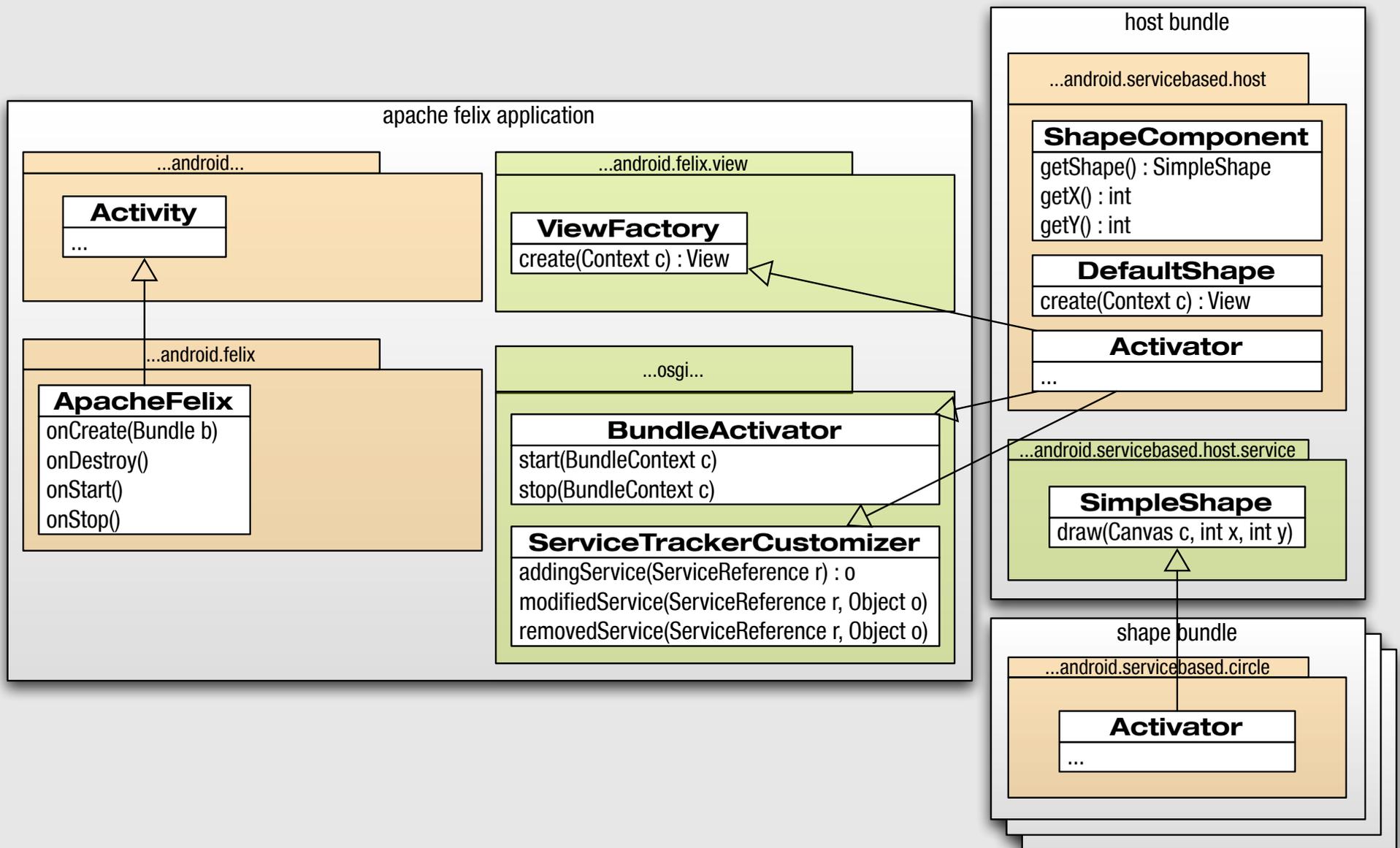
- At EclipseCon 2008, Santa Clara:
 - Neil Bartlett and BJ Hargrave, ported both Equinox and Concierge to Android
 - Slides at:
<https://eclipsecon.greenmeetingsystems.com/attachments/download/390>
- ProSyst announced:
 - A port of their mBedded Server:
<http://www.adon-line.de/kunden/prosystBlog/?p=24>
- Knopflerfish
 - We talked to Eric Wistrand and Christer Larsson of MakeWave but they have no plans yet

A dynamic application

- Apache Felix framework
 - embeds an activity to hook into;
 - embeds file install bundle for easy deployment.
- Host bundle that provides a canvas and a toolbar
- Shape bundles that add new shapes
- Based on example from Apache Felix website:
<http://felix.apache.org/site/apache-felix-application-demonstration.html>



Architecture



Live demo!

- Deploying Felix to the Phone Emulator
- Installing the first bundle, the host application
- Adding and removing plugins: square, circle and triangle

Links

- Slides, docs and code:
<http://opensource.luminis.net/>
- Android SDK:
<http://code.google.com/android/>
- Open Handset Alliance:
<http://www.openhandsetalliance.com/>
- Apache Felix and OSGi:
<http://felix.apache.org/>
<http://www.osgi.org/>
- Karl Pauls: karl.pauls@luminis.nl
Marcel Offermans: marcel.offermand@luminis.nl

Questions?!

? & !