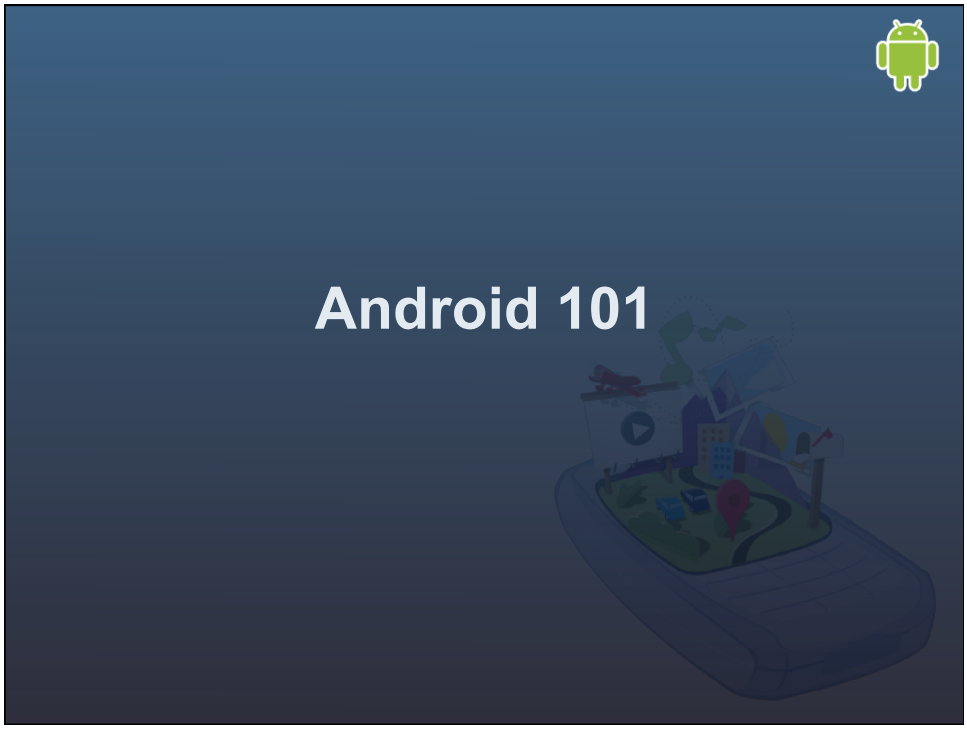

Android 101


Introduction to Android
• Installing the SDK
• Introduction to an android Activity/ App.
• Layouts
• The manifest.xml
• Intro. to Intents

Most important Websites

- SDK ：
- http://developer.android.com

- Android source:
- http://source.android.com/

Additional Website

- Google groups:
- http://groups.google.com

- Android Developers
- android-platform

Supported Operating Systems

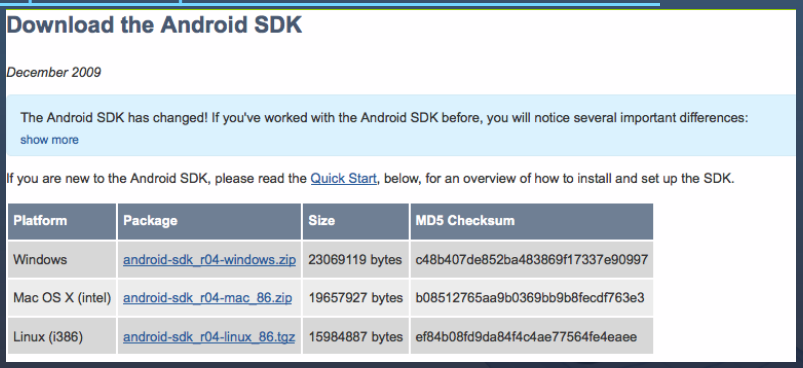
- Linux (tested on Linux Ubuntu Dapper Drake)
- Windows XP (32-bit) or Vista (32- or 64-bit)
- Mac OS X 10.4.8 or later (x86 only)

Pre install Requirements

- Eclipse 3.3 (Europa), 3.4 (Ganymede)
- Eclipse Classic IDE package is not supported.
- JDK 5 or JDK 6 (JRE alone is not sufficient)
- Apache Ant 1.6.5 or later for Linux and Mac, 1.7 or later for Windows???

Install SDK

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



Download the Android SDK

December 2009

The Android SDK has changed! If you've worked with the Android SDK before, you will notice several important differences:
show more

If you are new to the Android SDK, please read the Quick Start, below, for an overview of how to install and set up the SDK.

Platform	Package	Size	MD5 Checksum
Windows	android-sdk_r04-windows.zip	23069119 bytes	c48b407de852ba483869f17337e90997
Mac OS X (intel)	android-sdk_r04-mac_86.zip	19657927 bytes	b08512765aa9b0369bb9b8fecdf763e3
Linux (i386)	android-sdk_r04-linux_86.tgz	15984887 bytes	ef84b08fd9da84f4c4ae77564fe4eaee

- Example I created directory called android...
- /Users/bruce/android/android-sdk-mac_86

Setup development env.

- On Linux, edit your ~/.bash_profile or ~/.bashrc file. Look for a line that sets the PATH environment variable and add the full path to the tools/ directory to it. If you don't see a line setting the path, you can add one: export PATH=${PATH}:<your_sdk_dir>/tools
- On a Mac, look in your home directory for .bash_profile and proceed as for Linux. You can create the .bash_profile if you haven't already set one up on your machine.
- On Windows, right-click on My Computer, and select Properties. Under the Advanced tab, hit the Environment Variables button, and in the dialog that comes up, double-click on Path (under System Variables). Add the full path to the tools/ directory to the path.

Test your setup

- In the command window type adb
- You should see the following
- Android Debug Bridge version 1.0.20 ....

- Now we're ready for eclipse!

Install Eclipse

- http://www.eclipse.org/downloads/

Eclipse 3.4 (Ganymede) setup

- After you install eclipse
- Start Eclipse, then select Help > Software Updates....
- In the dialog that appears, click the Available Software tab.
- Click Add Site...
- Enter the Location:
- https://dl-ssl.google.com/android/eclipse/
- If you have trouble aqcuiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).
- Click OK.

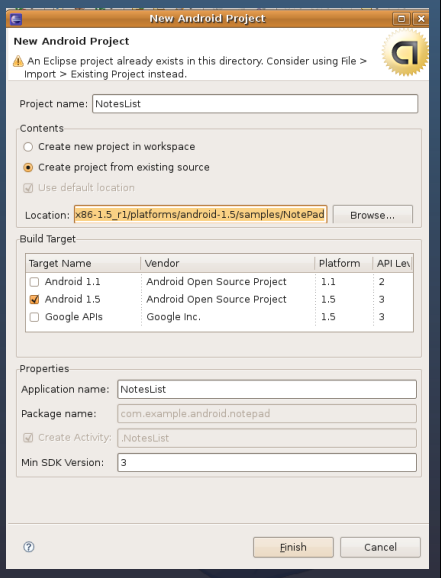
Eclipse 3.4 setup

- Back in the Available Software view, you should see the plugin listed by the URL, with "Developer Tools" nested within it. Select the checkbox next to Developer Tools and click Install...
- On the subsequent Install window, "Android DDMS" and "Android Development Tools" should both be checked. Click Next.
- Read and accept the license agreement, then click Finish.
- Restart Eclipse.

Introduction to the Android Environment

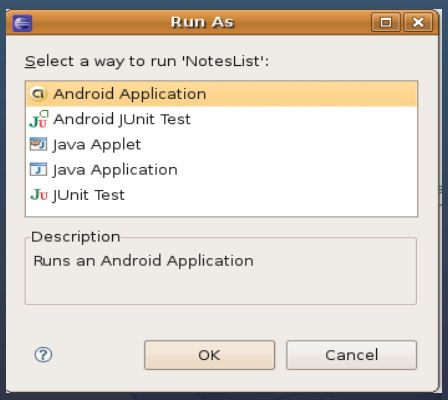
Compile Android Example

- Start Eclipse.
- Click File->New->android project
- Click Create project from existing
- Location : <SDK>platforms/ android-1.5/samples/NotePad
- Click Android 1.5 If its not selected
- Click Finish
- Project is created

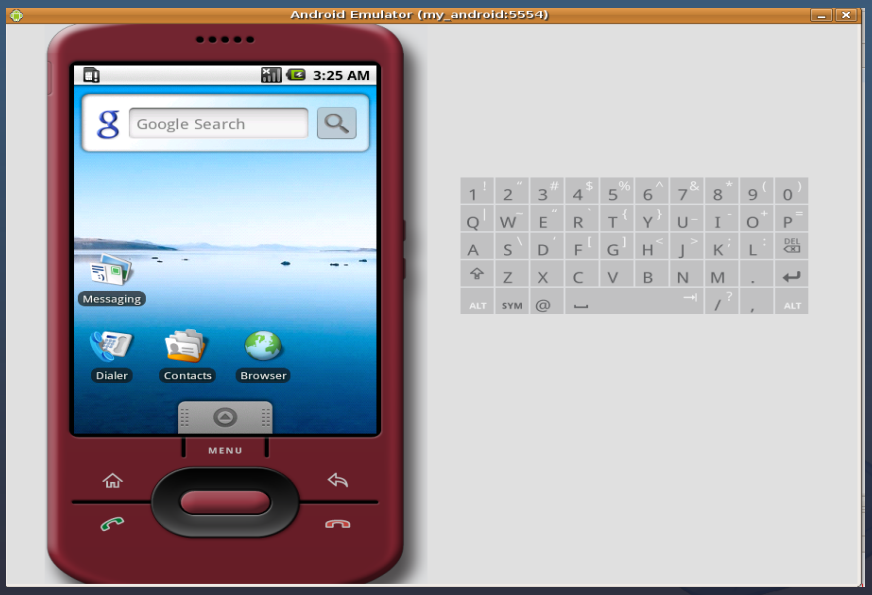


Starting the emulator

- Click Run->Run
- Select Android app click ok

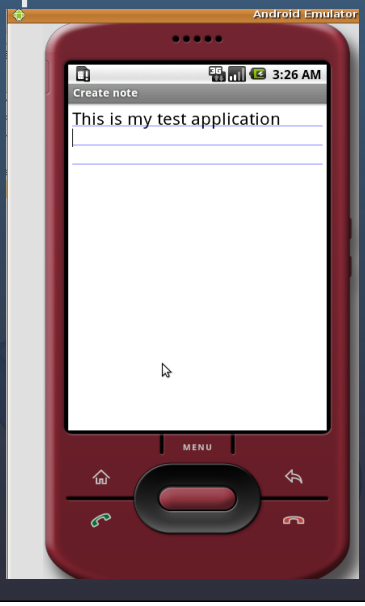


Emulator will Start



Running your application

- Once the application launches select menu.
- Click on add note
- Thats It.....



Introduction to the Android OS

What Android is not

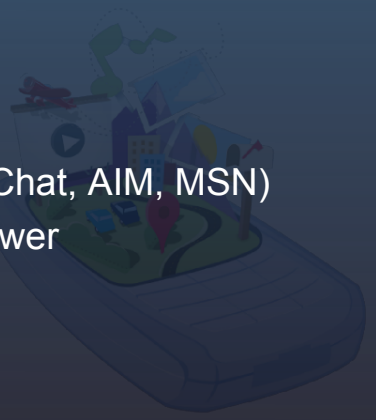
- A Java ME implementation
- An application Layer (UIQ or S60)
- A handset
- Google's Answer to iPhone…
- … nor a way of locking people into Google apps.

Openness of Android

- "The first truly open and comprehensive platform for mobile devices…"
- Android Components:
  - A hrdw. reference design describing the min. requirements to support the stack
  - Linux Kernel
  - Open Source Libraries
  - Run time environment (Dalvik)
  - Application Framework
  - A user interface framework
  - Set of pre-installed applications (a fully functioning Smart Phone)
  - Software devel. kit (Tools, plug-ins, and documentation)

Android Applications

- An eMail client (GMail)
- SMS management app.
- PIM (Google calendar, etc)
- Google Maps App.
- WebKit based browser
- Instant Messaging Client (GChat, AIM, MSN)
- Music Player and Picture viewer
- Android Market Place



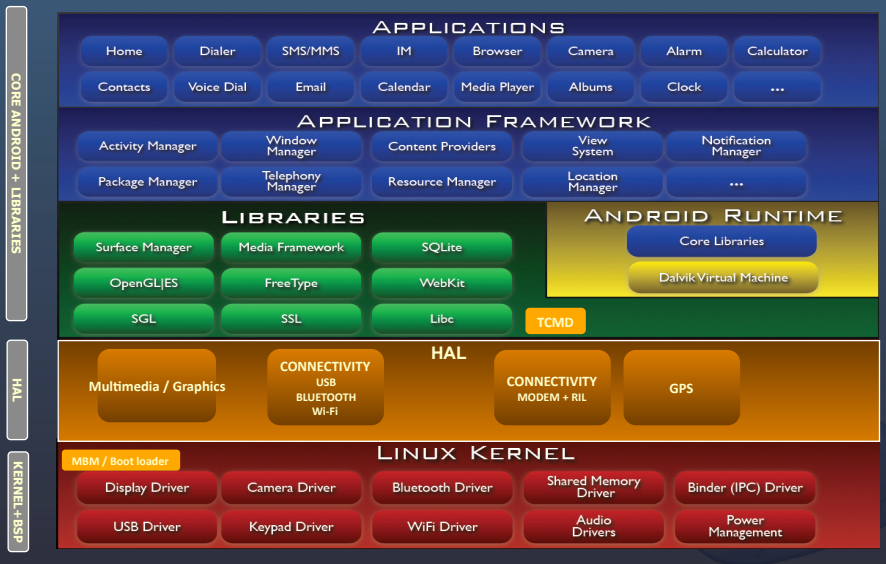
Android SDK Features

- Open platform (no fees, no licensing)
- Wi-fi hrdw. acess
- Full comm. stack (GSM, EDGE, 3G, Bluetooth)
- GPS
- Multimedia (playback and recording of audio, video, etc)
- APIs to accel. And compass hrdwr.
- IPC messaging
- Share Data stores
- Web-Kit browser
- P2P via Google Talk
- Eventually hwrd. accel. 3D graphics (Open GL ES)
- Media Libraries (Licensed for MP3, etc…)
- And open Application Framework (reuse and replacement)

Platform Features

- Agnostic Access to Hardware (GPS, accel., 3D, Geocoding, etc.)
- Background services
- SQLite DB
- Share data and Interapp. Communication
- P2P service with Google Talk
- Extensive Media Support
- Optimized Mem. and Process Mngmnt.

Android Layer Cake

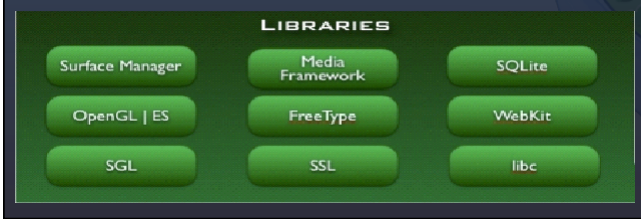


Linux Kernel

- Works as a HAL
- Device drivers
- Memory management
- Process management
- Networking

Libraries

- C/C++ libraries
- Interface through Java
- Surface manager – Handling UI Windows
- 2D and 3D graphics
- Media codecs, SQLite, Browser engine



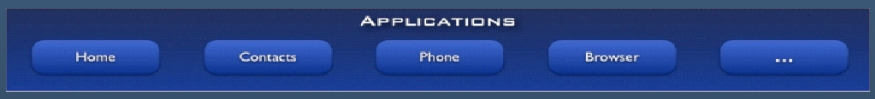
Android Runtime

- Dalvik VM
  - Dex files
  - Compact and efficient than class files
  - Limited memory and battery power
- Core Libraries
  - Java 5 Std edition
  - Collections, I/O etc…

Application Framework

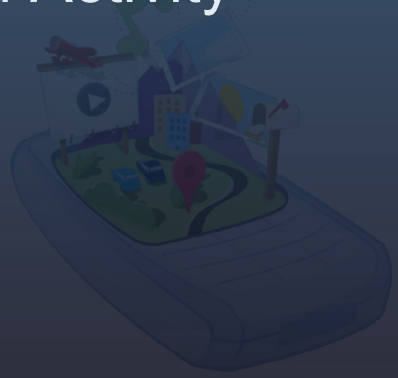
- API interface
- Activity manager – manages application life cycle.

Applications



- Built in and user apps
- Can replace built in apps

The Android Activity



Activities and Tasks

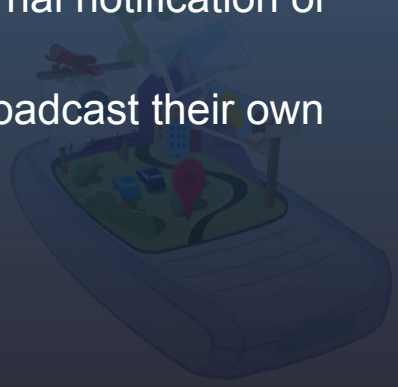
- Dan Morrill's definition:
  - An Activity is a "molecule": a discrete chunk of functionality
  - A task is a collection of Activities
  - A "process" is a standard Linux process

Activities

- Typically correspond to one UI screen
- But, they can:
  - Be faceless
  - Be in a floating window
  - Return a value

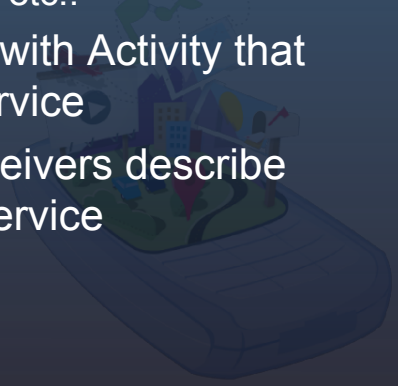
IntentReceivers

- Components that respond to broadcast ‘Intents’
- Way to respond to external notification or alarms
- Apps can invent and broadcast their own Intent

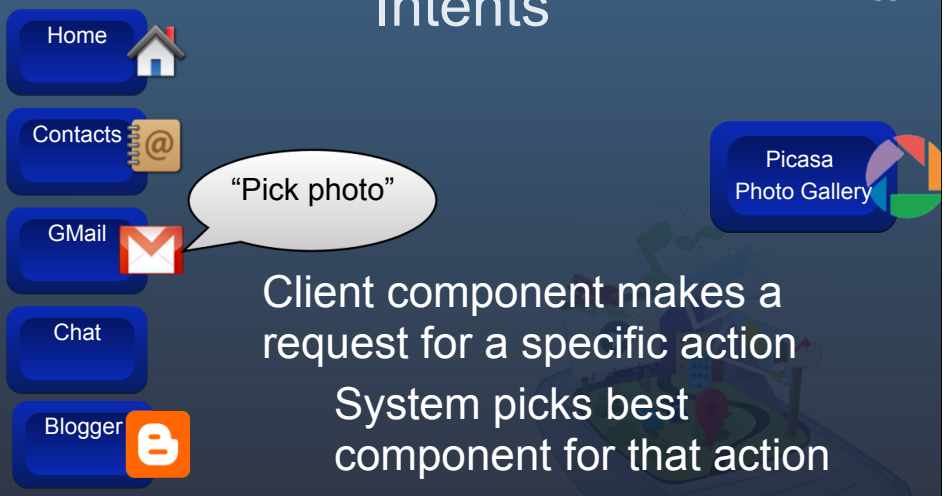


Intents

- Think of Intents as a verb and object; a description of what you want done
  - E.g. VIEW, CALL, PLAY etc..
- System matches Intent with Activity that can best provide the service
- Activities and IntentReceivers describe what Intents they can service



Intents



Client component makes a request for a specific action

- System picks best component for that action
- New components can use existing functionality

Services

- Faceless components that run in the background
  - E.g. music player, network download etc…

ContentProviders

- Enables sharing of data across applications
  - E.g. address book, photo gallery
- Provides uniform APIs for:
  - querying
  - delete, update and insert.
- Content is represented by URI and MIME type

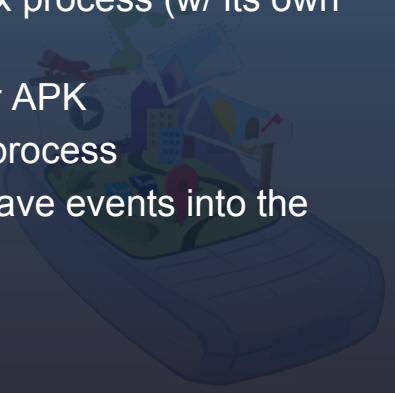
Activities vs Tasks (Apps)

- A concrete class in the API
- An encapsulation of a particular operation
- They run in the process of the .APK which installed them
- Optionally associated with a window (UI)
- An execution Context

- More of a notion than a concrete API entity
- A collection of related Activities
- Capable of spanning multiple processes
- Associated with their own UI history stack
- What users on other platforms know as “applications”

Process Basics

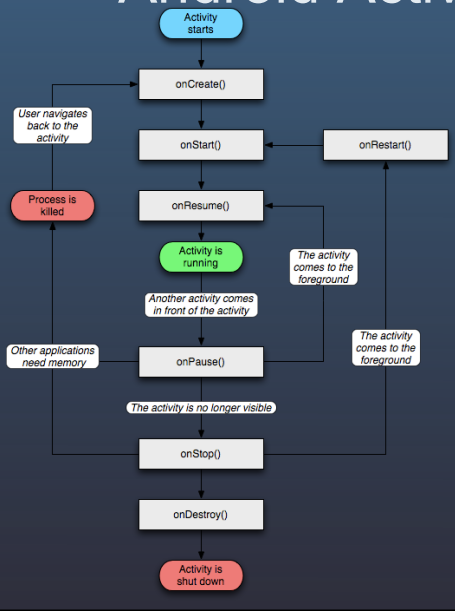
- How does it all of this relate to the Unix roots of Android?
  - Android process == Linux process (w/ its own unique UID)
  - By default, 1 process per APK
  - By default, 1 thread per process
  - Most components interleave events into the main thread



Application Lifecycle

- Application run in their own processes (VM, PID)
- Processes are started and stopped as needed to run an application's components
- Processes may be killed to reclaim resources

Android Activity Life Cycle



- Activities have several states
- Lifecycle methods are called on transitions
- You typically don't need to use them all, but they are there

Life Cycle example (Child Activity)

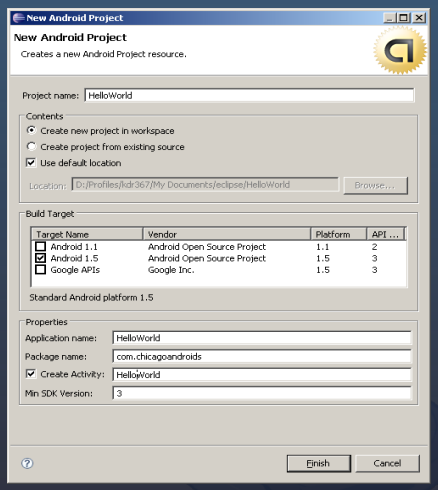
- Call sequence:
  - onCreate()
  - onStart()
  - onResume()
  - onFreeze()
  - onPause()
  - onStop()
  - onRestart()
  - onStart(), onResume(), ...

Android Application Building Blocks

- **Activities**: Building block of the UI. Every screen in your application will be an extension of the Activity class. You can think of an activity as being analogous to a window or dialog in a desktop environment.
- **Services**: Headless (non-UI) application that runs in the background. They are designed to keep running independent of any activity.
- **Content Providers**: Provide a level of abstraction for any data stored on the device that is accessible by multiple applications.
- **Intents**: A simple message passing framework. Using intents you can broadcast messages system-wide or to a target Activity or Service.
- **Broadcast Receivers**: Intent broadcast consumers. By registering a broadcast receiver your application an listen for broadcast Intents that match specific filter criteria.
- **Notifications**: User notification framework. Let you signal users without interrupting their current activity. For instance an incoming call can alert you with flashing lights, making sounds, or showing a dialog.

Hello World!!

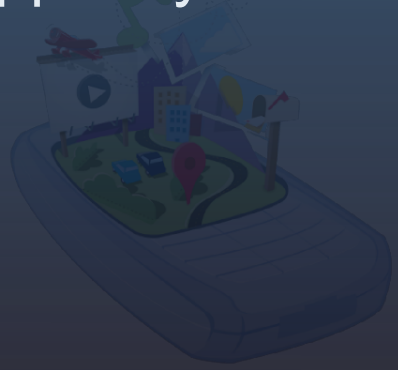
1. Create a new Android Project
   - Select **File** > **New** > **Android Project**
2. Fill out the project details
   - Enter *HelloWorld* for **Project Name**
   - Select **"Create new project in workspace"**
   - Enter HelloWorld in App name.
   - Enter *com.uwandroids.HelloWorld* in **Package Name**
   - Enter *HelloWorld* in **Activity name** (and yes we want to create an Activity)



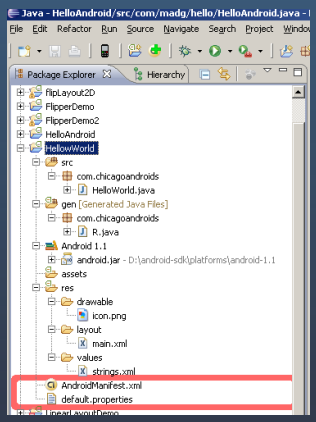
Project Properties

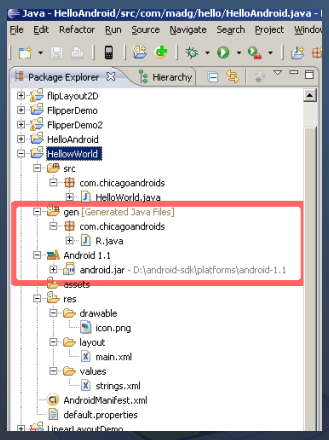
Project Name	This is the name of the directory or folder on your computer that you want to contain the project.
Package Name	This is the package namespace (following the same rules as for packages in the Java programming language) that you want all your source code to reside under. This also sets the package name under which the stub Activity will be generated. The package name you use in your application must be unique across all packages installed on the system; for this reason, it's very important to use a standard domain-style package for your applications. In the example above, we used the package domain "com.chicagoandroids".
Activity Name	This is the name for the class stub that will be generated by the plug-in. This will be a subclass of Android's Activity class. An Activity is simply a class that can run and do work. It can create a UI if it chooses, but it doesn't need to.
Application Name	This is the human-readable title for your application.

The Activity/App. Layout



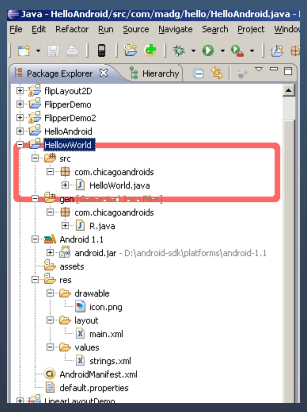
The Automatic* Portions…

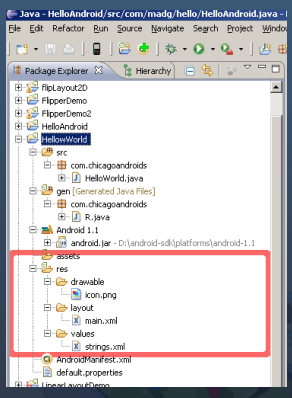




- Left: Manifest (* not that automatic)
- Right: R class and the android library (no need to touch)

The Automatic* Portions…

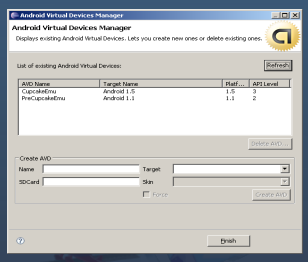


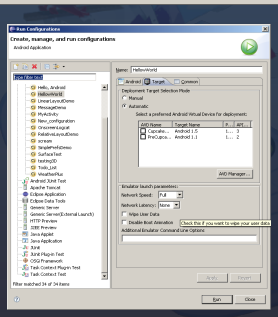


- Left: Source directories, where your classes go…
- Right: Resources (this is what gets automatically build into the R class)

A word about the emulator

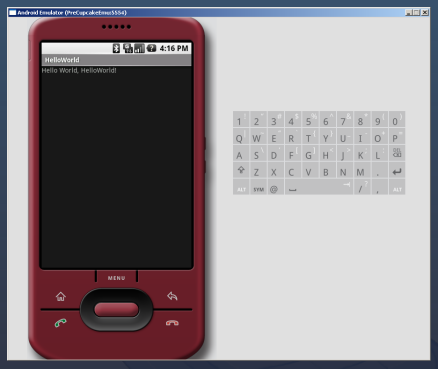
- You can create different Run configurations for different target devices.
- It is possible to target different resolutions (HVGA, HVGA-P, HVGA-L, etc)
- Network speed and latency, etc.
- Use the AVD manager and the ‘Run->Run configurations’ to manipulate





Run hello world

- Select the root of the project.
- Click in the ‘green play icon’.
- Pick Android Project
- That will get the emulator going…



Activity Layouts

- Where do they live?
- Why? Dynamic instantiation is possible, but it is discourage (compile vs dynamic).
- What are they? An XML-based layout is a specification of widget's relationships to each other encoded in XML.

Layout's most basic example…

```
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/button"
  android:text=""
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"/>
```

In this example Button is the root element…but a container would be more typical.

Containers pour a collection of widegets (and maybe child containers) into a specific layout. More on that in the future

So, what can you Intent to do?

Intents are system messages that notify applications of various events:

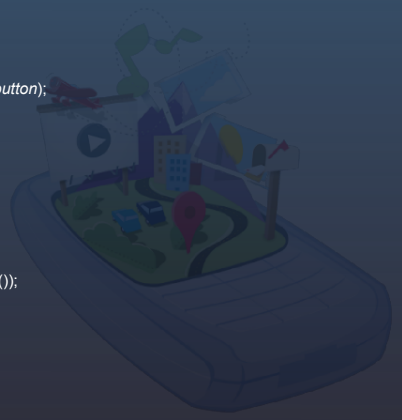
- Activity events (launch app, press button)
- Hardware state changes (acceleration change, screen off, etc)
- Incoming data (Receiving call, SMS arrived)

You can also create your own to launch applications, etc.

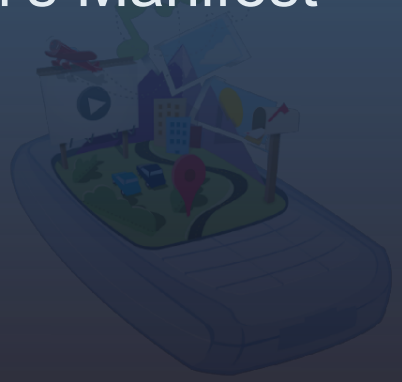
- Inter-activity communications (open an app when you get to a location, etc)

Intent most basic example…

```
public class NowRedux extends Activity
implements View.OnClickListener {
Button btn;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        setContentView(R.layout.main);

        btn=(Button)findViewById(R.id.button);
        btn.setOnClickListener(this);
        updateTime();
    }

    public void onClick(View view) {
        updateTime();
    }

    private void updateTime() {
        btn.setText(new Date().toString());
    }
}
```



The Application's Manifest



The manifest

- Declares the permissions the application will need (uses-permission)
- Declare permissions that activities or services might require to use your app (permission)
- Provides instrumentations elements (instrumentation)
- Defines the guts of the application
- Provides hooks to connect to optional Android components such as mapping (uses-library)

Default AndroidManifest.xml

- One application node. Application properties include icon and application label in home screen.
- One Activity node. Activity name is abbreviated to .Sample. Tells Android which Java class to load. The activity label is the title for that activity in the titlebar.
- Intent-filter tags specify which Intents launch the Activity. In order for an application to be available from the launcher it must include an Intent Filter listening for the MAIN action and the LAUNCHER category.

```
<manifest xmlns:android-http://schemas.android.com/apk/res/android
   package="com.motorola.Sample">

    <application android:icon="@drawable/icon"
                 android:label="@string/app_name">
        <activity android:name=".Sample"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Android Manifest (cont)

```
<manifest xmlns:android-http://schemas.android.com/apk/res/android
    package="com.mydomain.myapp">

    <application android:icon="@drawable/icon"
                 android:theme="@style/my_theme">

        <activity android:name=".MyActiv" android:label="@string/app_name">
                 <intent-filter> . . . </intent-filter>
        </activity>

        <service android:enables="true" android:name="MyService">
                 <intent-filter> . . . </intent-filter>
        </service>

        <provider android:permission="com.paad.MY_PERMISSION" . . .>
        </provider>

        <receiver android:enabled="true"
                 android:label="My Broadcast Receiver"
                 android:name=".MyBroadcastReceiver">
        </receiver>
    </application>
</manifest>
```

Android Manifest (cont)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bikerolas"
    android:versionCode="30"
    android:versionName="1.2">
  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
  <uses-permission android:name="android.permission.ACCESS_LOCATION />
  <uses-permission android:name="android.permission.ACCESS_GPS" />
  <uses-permission android:name="android.permission. ACCESS_CELL_ID />

  <application android:icon="@drawable/flingicn1" android:label="@string/app_name" android:debuggable="false">
    <activity android:name=".Fling"
          android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
<service android:name=".FlingService" />
<receiver android:name=".FlingServiceManager"
      android:permission="android.permission.RECEIVE_BOOT_COMPLETED">
<intent-filter>
  <action android:name="android.intent.action.BOOT_COMPLETED" />
</intent-filter>
</receiver>
</application>

<uses-sdk android:minSdkVersion="2"></uses-sdk>
</manifest>
```

Citation

- This presentation contains references from the following sources:
  - The Busy Coder's Guide to Android Development (by Mark L. Murphy)
  - Inside the Android Application Framework (by Dan Morrill) http://sites.google.com/site/io/inside-the-android-application-framework
  - Dalvik VM Internal (by Dan Bornstein) http://sites.google.com/site/io/dalvik-vm-internals