ANDROID DEVELOPMENT

Interesting Web Sites

http://developer.android.com
http://www.frandroid.com
http://source.android.com
www.openhandsetalliance.com

Discover tools

Android SDK

- Android Software Development Kit
- Available on Windows, Mac OS X and Linux
- Used in order to develop Android Application
- Contains the emulator, APIs, tools and more
- Provided by Google

Test

Create a Project

- Use Android Studio
- Create a new Android Project named HelloDroid with :
  - Android 2.2 selected in Build Target list
  - Application name : **“Hello Droid !”**
  - Package name : **“com.android.hellodroid”**
  - Activity name : **“Main”**

- Look at the project contents

Test

Execute it in an Emulator

- Setup an Android Virtual Device
  - Return to **Android SDK and AVD Manager**
  - Select **Virtual Devices**
  - Click **New…** and fill the form to create the new AVD
- Execute your application in your emulator
  - Right click on your project
    - Run As…
      - Android Application

Test

Execute it in an Emulator

Next topics covered

- The Android Platform
- Android Project Tree
- Activities
- User Interface
- Intents

The Android Platform

Discover Android.

The Android Platform

Presentation

- Android Platform Key points :
  - Innovative : Integrates all the latest phone technology
    - Touchscreen, GPS, Camera, Accelerometer…
  - Accessible :
    - No need to buy specific Software or Hardware to develop
    - Java is a very widely used language, no need to learn another language
  - Open Source
    - Source code available

The Android Platform

Presentation

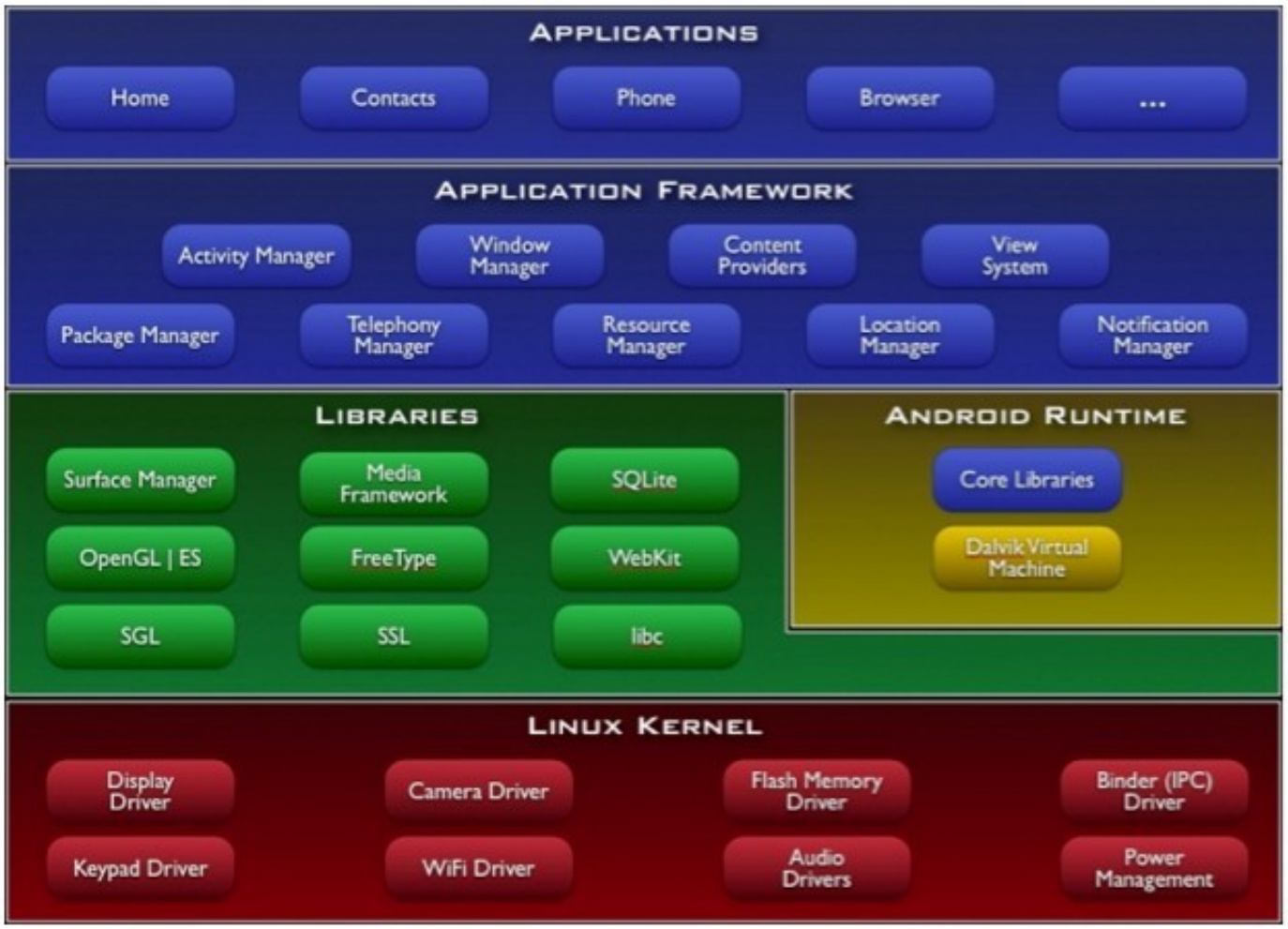
- Android is designed for mobile devices in a broad sense :
  - Phones
  - Tablets
  - Televisions
  - Headphones
  - Microwaves
  - ...

Platform Components

- Android is composed of different layers :
  - A Linux Kernel
  - Libraries for UI, Multimedia, Persistence, etc…
  - A specific Java Virtual Machine called **Dalvik**
  - An Application Framework with many features
  - Applications

The Android Platform

Platform Components

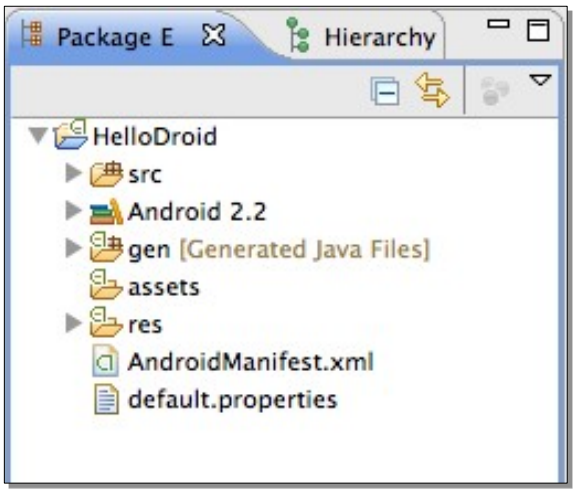


Android Project Tree

Android Project Tree

Overview

- When ADT creates a project, it generates a set of folders and files :
  - **src** folder
  - **res** folder
  - **gen** folder
  - **assets** folder
  - **AndroidManifest.xml** file
  - **default.properties**



Android Project Tree

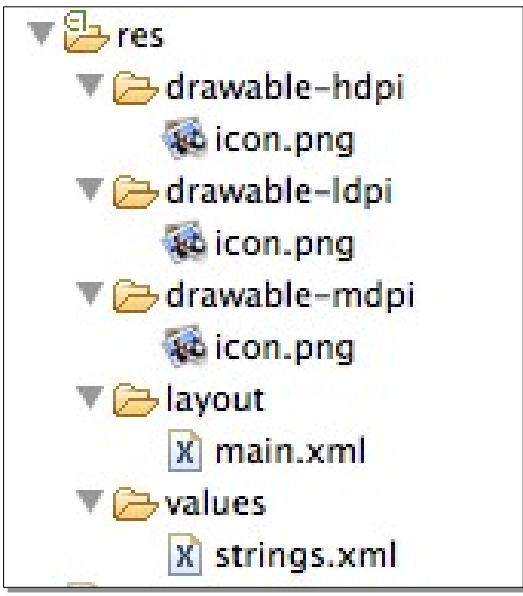
Folder : src

- The same folder as in Java Projects
- Contains packages and Java code files

Android Project Tree

Folder : res

- Contains all the resources needed by the application
- Three main types :
  - **drawable** : images and animations
  - **layout** : XML layout files used to construct static user interfaces
  - **values** : resources such as strings and integer constants



Android Project Tree

Folder : gen

- Contains Java files auto-generated by ADT (Android Development Tools)
- Contains the class **R** :
  - Special static class
  - References the data contained in resource files
  - Contains one static inner class per resource type

```
public final class R{
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    ... //code omitted
}
```

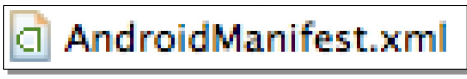
Android Project Tree

Folder : assets

- Contains asset files
  - Quite similar to resources
  - Accessed in a classic file manipulation style
    - With stream of bytes
  - Use AssetManager class to open them
  - Not to use extensively as resources

Android Project Tree

File: AndroidManifest.xml



- Mandatory file in every Android project
- Contains information needed by Android to run the application
  - Package name of the application
  - List of Activities, Services, Broadcast Receivers, …
  - Permissions needed by the application
  - etc…

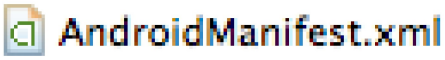
Android Project Tree

File: AndroidManifest.xml

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.hellodroid"
    android:versionCode="1"
    android:versionName="1.0">

  <application android:icon="@drawable/icon"
               android:label="@string/app_name">
    <activity android:name=".HelloDroid"
              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 Project Tree

File: default.properties



- Contains all the project settings :
  - The build target
  - If the project is a library or not
  - Library references
  - etc…

- Never edit this file manually :
  - Edit these properties using the Properties Project window of Eclipse

Activities

Presentation layer

Activities

Preview

This is the content we'll see :

- Presentation
- Example
- Life Cycle

Activities

Presentation

- An activity is a sort of screen composed of several views and controls :
  - For instance an add contact form or a personalized Google Map
- As many activities as application screens
- Presentation layer of an application

Presentation

- Composed of two parts :
  - **The Activity Logic** :
    - Defined in Java inside a class extending **android.app.Activity**
  - **The User Interface** :
    - Defined either in Java inside the Activity class or inside an XML file (in the folder **/res/layout/**)

Example

- Activity class simple example :

```
package com.android.hellodroid;

import android.app.Activity;
import android.os.Bundle;

public class Main extends Activity {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

}
```

Activities

Example

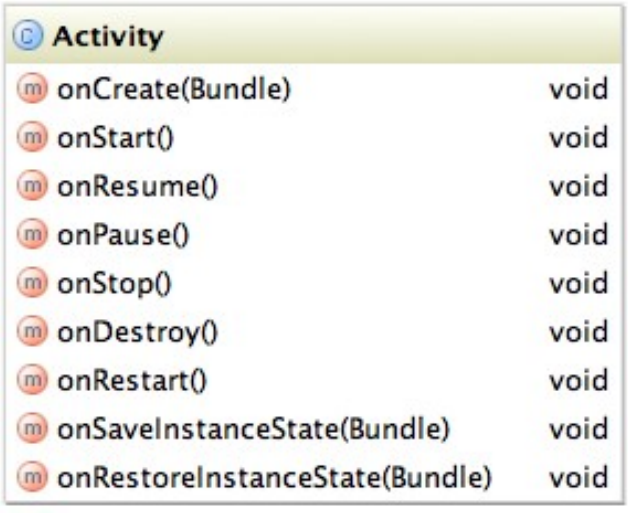
- Layout file simple example :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >

  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />

</LinearLayout>
```

Activities

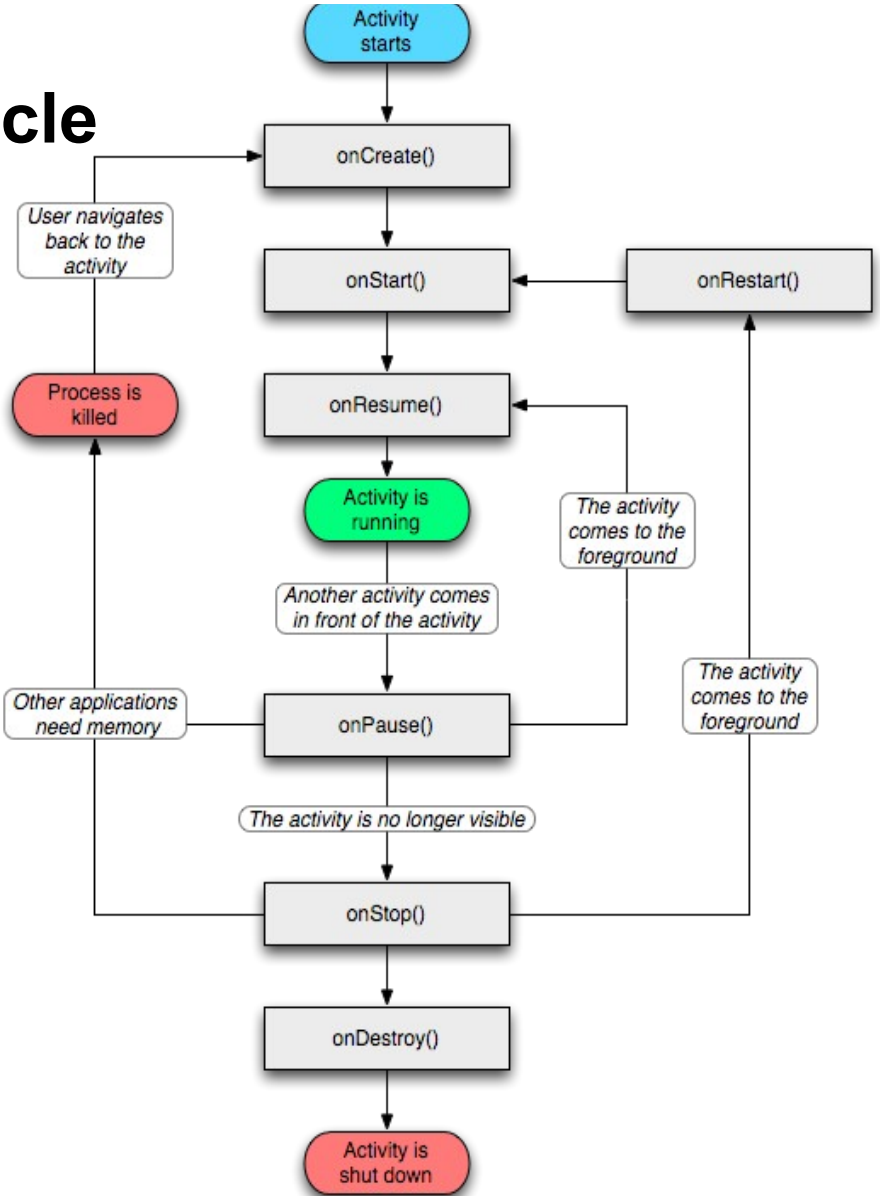
Life Cycle

- An activity can be in three different states :
  - **Active**
    - The activity is visible and has the user focus
  - **Paused**
    - The activity is at least partly visible but doesn't have the focus
  - **Stopped**
    - The activity is not visible
- Activity class defines methods to manage life cycle



Activities

Life Cycle



Activities

Declare an Activity

- To be usable, an activity must be declared
  - Inside the **AndroidManifest.xml** file :

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

Main Activity declaration

Activity declarations

Resources

How to use them.

Resources

Preview

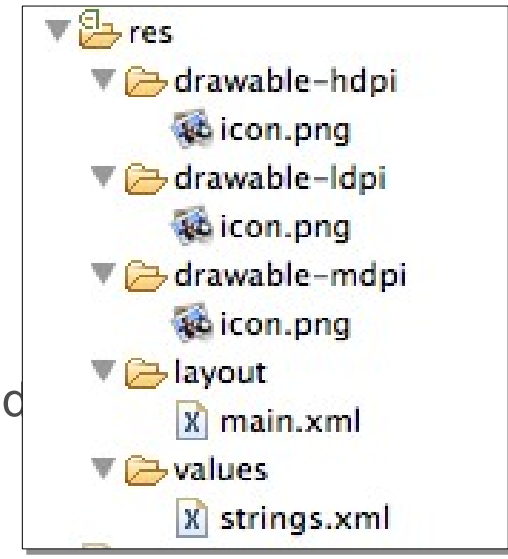
This is the content we'll see :

- Presentation
- Use Resources
- System Resources
- Simples Values
- Images

Resources

Presentation

- Android externalizes resources like :
  - Images
  - Strings
  - User Interface description
  - …
- Appropriate for static “data” determined execution of the aplication
- Easier to manage and to maintain
- Contained inside the **res** folder



Use the resources

- Resources are accessible inside the code thanks to the static class : **R**
  - This class is automatically generated by ADT
  - When you add a resource inside the **res** folder, ADT adds a reference to it inside the R class
- The syntax to retrieve a resource reference is :

```
R.resource_type.resource_name
```

Resources

Example

```
public final class R{
    public static final class string {
        public static final int app_name=0x7f020000;
    }
    public static final class layout {
        public static final int my_screen=0x7f030000;
    }
    ... //code omitted
}
```

```
// Define the layout of an activity
setContentView(R.layout.my_screen);

// Retrieve the application name
Resources resources = getResources();
String appName = resources.getString(R.string.app_name);
```

Resources

Use the resources

- You can also use resources inside XML files
- Very much used in layouts
- Syntax :

```
"@[package_name:]resource_type/resource_identifier"
```

```
...
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
...
```

Resources

System Resources

- Android already includes a number of resources
  - Predefined Colors
  - Predefined Strings
  - Predefined Images
- Examples :

```
String cancel =
       resources.getString(android.R.string.cancel);
```

```
...
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@android:color/darker_gray"
    android:text="@string/hello"
    />
...
```

Resources

Simple values

- Simple values are stored in XML files inside the **/res/values** folder
- You can declare
  - **Strings**
    - You can use the HTML tags **<b>**, **<i>** and **<u>**
  - **Colors**
    - Accepts #RGB, #ARGB, #RRGGBB and #AARRGGBB format
  - **Dimensions**
    - In pixels (px), inches (in), millimeters (mm), points (pt), density-independent pixel (dp) or scale-independent pixel (sp)
  - **Arrays**
    - Of Integers or Strings.

Resources

Simple values

- Example :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="Cyan">#00FFFF</color>
  <string name="first_name">Brice</string>

  <string-array name="my_array">
    <item>A string</item>
    <item>Another String</item>
  </string-array>

  <integer-array name="my_other_array">
    <item>123</item>
    <item>456</item>
  </integer-array>

  <dimen name="my_dimension">4dp</dimen>
  <dimen name="text_size">4px</dimen>
</resources>
```

Resources

Images

- Android accepts different bitmap formats for resources :
  - **PNG** (advised by the documentation)
  - **JPEG**
  - **GIF** (deprecated)

- Starting Android 1.6, three folders :
  - **drawable-hdpi** : resource for high-resolution screens
  - **drawable-mdpi** : resources for medium-resolution screens
  - **drawable-ldpi** : resources for low-resolution screens