Introduction to Android Development

Architecture
Activities, Views, Intents
Installing Android Studio & SDK
Walkthrough, MVC in Android
• What is Android?
  – Introduction
  – Architecture
    – Activities, Views, Intents
• Android Developer Tools
  – Installing the Android SDK
  – Setting up an AVD
• Demo
  – Creating a project, project structure
  – Building and deploying to an AVD
  – Using MVC
• Wrap-up
  – A4 Settings
  – Tips & Tricks
Get Started with Android Studio

Everything you need to build incredible app experiences on phones and tablets, Wear, TV, and Auto.

› Set up Android Studio
› Build your first app
› Learn about Android
› Sample projects

http://developer.android.com/
What is Android?

• Pervasive Mobile Platform
  – Runs on hundreds of millions of mobile phones, tablets
  – World’s most popular & frequently installed mobile OS
  – Open Source (minimum-definition).

• Developing on Android
  – Multi-platform tools
  – Based on Java
  – Uses Android Studio IDE
  – Excellent third-party tools support

• See http://developer.android.com/tools/index.html
Design Considerations

• Mobile OS with limited resources
  – Limited memory, processing power
  – Battery life is critical!

• Architectural tradeoffs
  – The OS aggressively flushes memory
  – Background computation is limited

• Unusual device characteristics
  – Small screen
  – Multiple orientations, dynamic layout
  – Single window/application running
  – Multi-touch input
• Layered environment
  – **Bottom tier**: Linux 2.6 kernel
  – **Mid tier**: Android libraries/services
  – **Top tier**: Java applications

• Applications run in virtual machines (VM)
  – Each application has its own VM & address space
  – Restrictions on sharing resources and data
  – Dalvik virtual machine process
Design Goals

- Applications provide multiple entry points; they’re distinct components that can be invoked individually.
  - **Activity**: a single screen in an application
  - **Service**: runs in a separate background thread
  - **Intent**: a way for a component to start another component.

- Applications are dynamic
  - You should provide different layouts (XML layouts, and resources) based on device characteristics.
  - Applications should dynamically adjust based on device screen size and orientation.

Overview

• What is Android?
  – Introduction
  – Architecture
  – Activities, Views, Intents

• Android Developer Tools
  – Installing the Android SDK
  – Setting up an AVD

• Demo
  – Creating a project, project structure
  – Building and deploying to an AVD
  – Using MVC

• Wrap-up
  – A4 Settings
  – Tips & Tricks
Activities

- Different types of components that we can build in Android: applications, services, etc. (see chart below)
- A standard application component is an **Activity**
  - Typically represents a single screen
  - Main entry point (equiv. to main() method)
  - For most purposes, this is your application class!

<table>
<thead>
<tr>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>Single-screen of an application</td>
</tr>
<tr>
<td>Service</td>
<td>Long-running background application</td>
</tr>
<tr>
<td>Content provider</td>
<td>Provides shared set of application data</td>
</tr>
<tr>
<td>Broadcast receiver</td>
<td>Responds to system broadcast events</td>
</tr>
</tbody>
</table>
Activities have an explicit lifecycle
- One activity runs at a time, others are paused in the background.
- As users navigate through your application, they switch activities.

Every activity has a state: run, paused, or stopped.
- Changing state fires a corresponding activity method.
• Common to switch between different activities/screens.
  – Activities can create other activities ("Back stack")
  – Navigation forward/back through activities is typically triggered by user actions
• Applications can stop at any time (i.e. user quits, OS kills it).
  – Each activity needs to manage its own state.
  – Activities have methods for saving and restoring state

http://developer.android.com/training/basics/activity-lifecycle/pausing.html
Intents

- We use **intents** to pass data between activities/screens.
  - Data structure holding an abstract description of an action.
  - Use Activity `startActivity()` method to launch with intent.
    - **Explicit** (named activity) vs. **implicit** (capabilities, e.g. camera)

[Diagram of intents flow]

UI Components

- **android.view.ViewGroup**
  - Abstract base class, replaces LayoutManagers in Java
  - Subclasses: AbsoluteLayout, RelativeLayout

- **android.view.View**
  - Abstract base class, drawing and event handling
  - Subclasses: ImageView, TextView, ProgressBar

- Views are managed by each activity (screen)
  1. Typically use XML layouts (dynamically).
  2. Can also load in code (manual).
Fragments can be thought of as *portions* of a UI

An activity can contain (and manage) multiple fragments.

- Allows for dynamic adjustments to the UI
- Fragments have their own lifecycle, and their own layout.
- Alternative to multiple activities

Your application will consist of:

- **Activities**
  - MainActivity as your main window (and entry point)
  - Possibly other activities

- **Views** within each Activity
  - View Groups containing Views (i.e. components)
  - Possibly fragments, managed by a Activity
  - Components using `android.widget`.

- **Intents** tying Activities together
Overview

• What is Android?
  – Introduction
  – Architecture
  – Activities, Views, Intents

• Android Developer Tools
  – Installing the Android SDK
  – Setting up an AVD

• Demo
  – Creating a project, project structure
  – Building and deploying to an AVD
  – Using MVC

• Wrap-up
  – A4 Settings
  – Tips & Tricks
• Android used to support Eclipse, and a standalone SDK
• Google recently standardized on Android Studio
  • Custom version of IntelliJ.
  • Includes IDE, SDK, supporting tools, documentation, sample code.
  • Runs on Mac, Windows, Linux

• Install from the Android Developer Portal
  • Develop -> Tools
• Android SDK is included, with command-line tools, libraries etc.
  – e.g. /Users/<user>/Library/Android/sdk/
• During installation, it will also install HAXM (Intel HW Acceleration).
• To update the SDK later: Tools -> Android -> SDK Manager
• Emulator that can be run on your machine for development.
• Can emulate a range of devices / capabilities / OS versions.
  – Android Studio -> Tools -> Android -> AVD Manager

**Android Virtual Device (AVD)**
1. **Create the project**
   - Normally use the New Project wizard in Android Studio.
   - Do NOT attempt to build a project by-hand.

2. **Setup the Android Virtual Device (AVD)**
   - Configure the emulator for a specific target device.
   - This is the profile that you’ll use for testing.
   - Make sure that HAXM is enabled!

3. **Setup the Run Configuration**
   - Specify the starting Activity that will be used, and the target (typically an AVD).
   - Test with an empty project first.

4. **Commit *Everything* to SVN.**
   - Commit early, and often.
   - Save your entire project, not just code!
Overview

• What is Android?
  – Introduction
  – Architecture
  – Activities, Views, Intents
• Android Developer Tools
  – Installing the Android SDK
  – Setting up an AVD
• Demo
  – Creating a project, project structure
  – Building and deploying to an AVD
  – Using MVC
• Wrap-up
  – A4 Settings
  – Tips & Tricks
• The New-Project wizard generates “Hello World” starter code.
• Great demonstration of how the project fits together.
• Recall the application lifecycle. Event states map directly to methods.
package com.example.helloworld;

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

public class MainActivity extends Activity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    public void onStart() { super.onStart(); }

    public void onPause() { super.onPause(); }

    public void onResume() { super.onResume(); }

    public void onStop() { super.onStop(); }
}
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}

- **OnCreate()**
  - Maps to the OnCreate() state
  - executed when the app runs (i.e. entry point!)
- **SetContentView(R.layout.activity_main)**
  - Default view
  - Example of Android using XML for view widgets
  - Look under `res/layout/activity_main.xml`
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
• To implement MVC, we can instantiate our model, view and controller classes directly in code.
  - Define views using resource files, but use code to instantiate them and bind them together.
Overview

• What is Android?
  – Introduction
  – Architecture
  – Activities, Views, Intents

• Android Developer Tools
  – Installing the Android SDK
  – Setting up an AVD

• Demo
  – Creating a project, project structure
  – Building and deploying to an AVD
  – Using MVC

• Wrap-up
  – A4 Settings
  – Tips & Tricks
A4 Settings

• Tools
  – Android Studio 1.5 (or later) on Mac, Windows or Linux
  – Android 6.0 (API 23) (installed by default)

• Project
  – New Android Application Project wizard
    • Minimum SDK: API 21: Android 5.0 (Lollipop)
    • Blank Activity (accept defaults)

• AVD
  – Create a Nexus 6 device, “Nexus 6 API 22“, with defaults:
    • Release Name: Lollipop
    • API Level: 22
    • ABI: x86
    • Target: Android 5.1
  – AVD Name: Nexus 6 API 22
    • Emulated Performance: Use Host GPU (checked)
    • Device Frame: (checked)
• AVD
  – The AVD is slow to launch, so keep it running in the background while you’re programming / debugging.
• Use the debugging tools in Android Studio
  – You can set breakpoints etc. as usual
  – *logcat* shows device output and you can write to it using the `android.util.Log` class (sim. to `printf`).

---

Tips & Tricks

[Image of Android Monitor]
Other Resources

• Android Developer Channel @YouTube
  – [http://www.youtube.com/user/androiddevelopers](http://www.youtube.com/user/androiddevelopers)
• Recommended (Optional!) Books
  – Mednieks et al., *Programming Android*