FPT Software

ANDROID TRAINING LESSON 1

Version 0.1

Agenda

- References
- Tools
- Android SDK
- Creating a Project
- Project Components
  - XML
  - R Class
  - Layouts
  - Strings
  - Manifest File

References

- This lesson is a brief overview of some major concepts
- Developer's Guide
  - http://developer.android.com/guide/index.html
- API Reference
  - http://developer.android.com/reference/packages.html

Tools

- Phone
- Eclipse (http://www.eclipse.org/downloads/)
  - Android Plugin (ADT)
- Android SDK (http://developer.android.com/sdk/index.html)
  - Install everything except Additional SDK Platforms, unless you want to

Android SDK

- Once installed open the SDK Manager
- Install the desired packages
- Create an Android Virtual Device (AVD)

SDK Manager

Android SDK Manager

Packages Tools

SDK Path: C:\Users\FMC\android-sdks\

Packages

Name	API	Rev.	Status
Tools			
Android SDK Tools		19	Installed
Android SDK Platform-tools		11	Installed
Android 4.0.3 (API 15)			
Documentation for Android SDK	15	2	Installed
SDK Platform	15	3	Installed
Samples for SDK	15	2	Installed
ARM EABI v7a System Image	15	2	Installed
Google APIs	*15*	*2*	*Not installed*
Sources for Android SDK	15	2	Installed
Android 4.0 (API 14)			
SDK Platform	14	3	Installed
Samples for SDK	14	2	Installed
ARM EABI v7a System Image	*14*	*2*	*Not installed*
Google APIs	*14*	*2*	*Not installed*
Sources for Android SDK	14	1	Installed

Show: ☑ Updates/New ☑ Installed ☐ Obsolete Select New or Updates

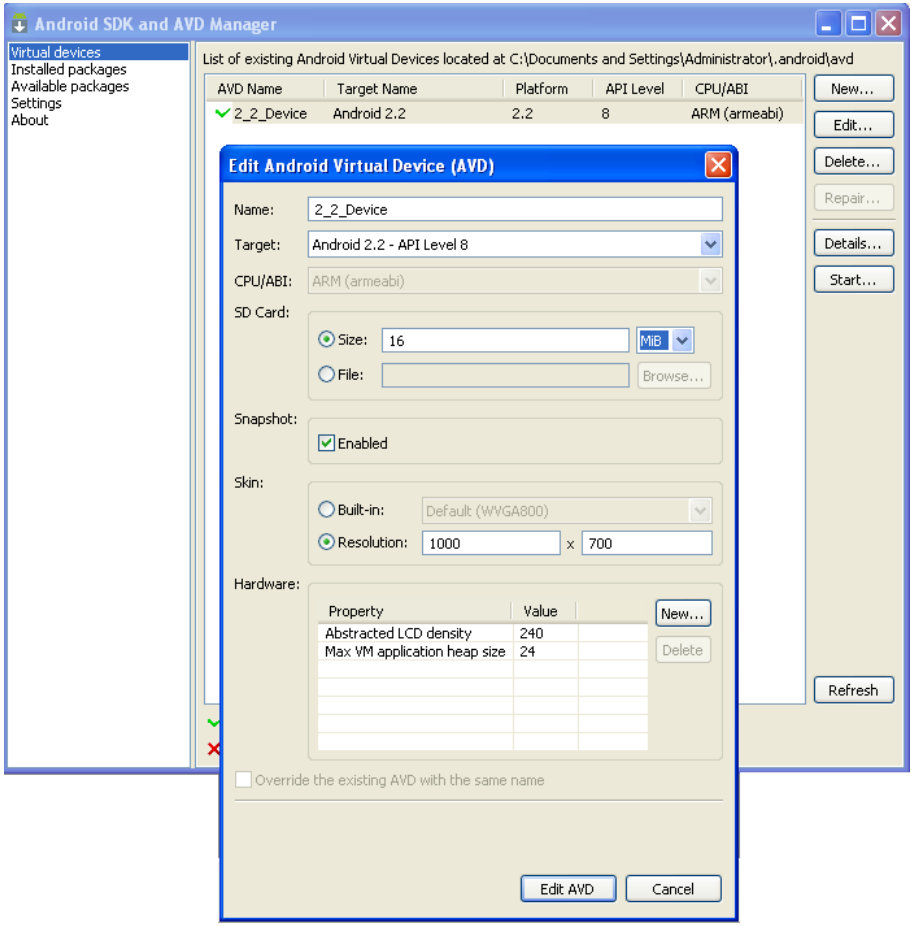
Install packages...

Sort by: ◉ API level ○ Repository Deselect All

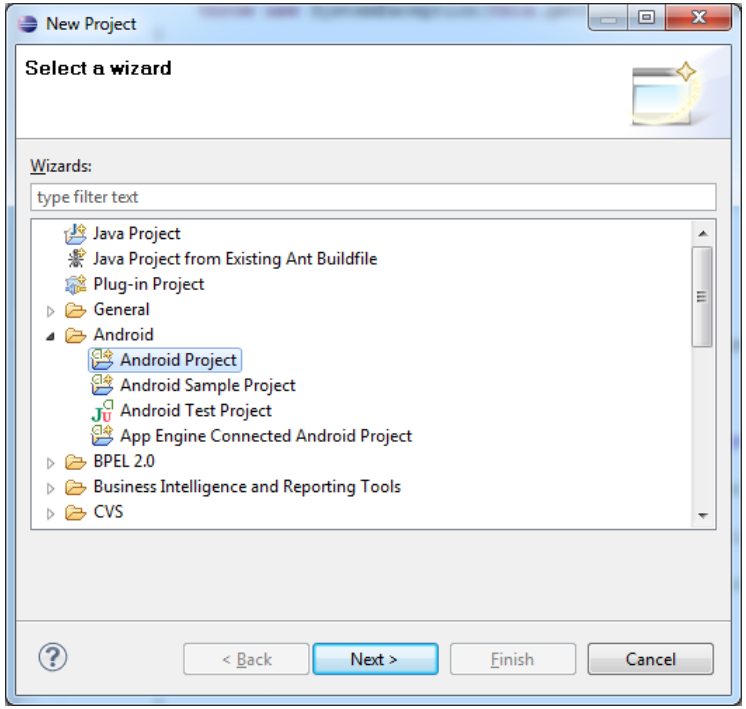
Delete packages...

Done loading packages.

AVD

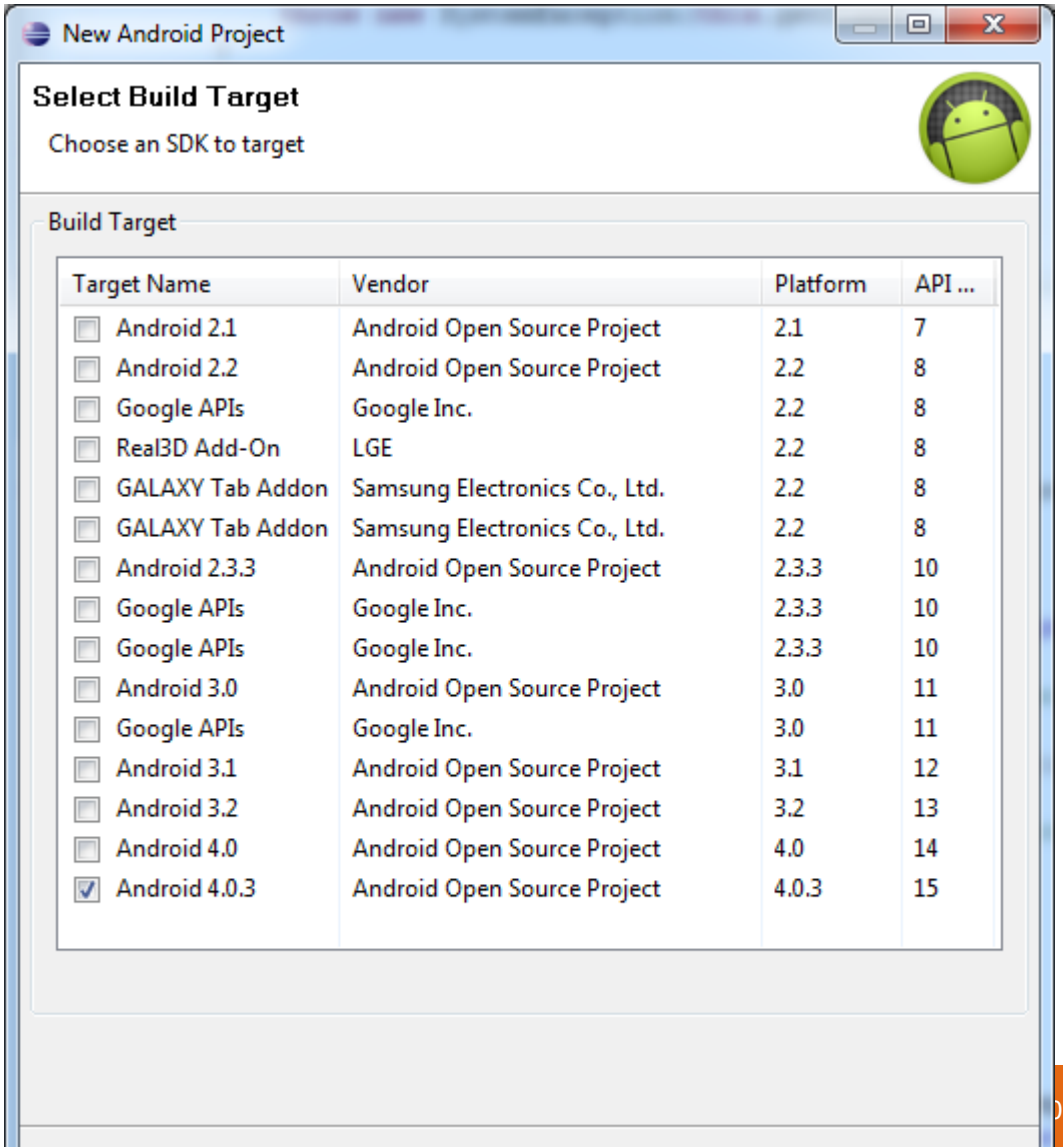


Creating a Project



- In Eclipse, select “File -> New -> Project….”, “Android Project”, and input your application detail. Eclipse will create all the necessary Android project files and configuration.

Creating a Project



Target Name	Vendor	Platform	API ...
Android 2.1	Android Open Source Project	2.1	7
Android 2.2	Android Open Source Project	2.2	8
Google APIs	Google Inc.	2.2	8
Real3D Add-On	LGE	2.2	8
GALAXY Tab Addon	Samsung Electronics Co., Ltd.	2.2	8
GALAXY Tab Addon	Samsung Electronics Co., Ltd.	2.2	8
Android 2.3.3	Android Open Source Project	2.3.3	10
Google APIs	Google Inc.	2.3.3	10
Google APIs	Google Inc.	2.3.3	10
Android 3.0	Android Open Source Project	3.0	11
Google APIs	Google Inc.	3.0	11
Android 3.1	Android Open Source Project	3.1	12
Android 3.2	Android Open Source Project	3.2	13
Android 4.0	Android Open Source Project	4.0	14
Android 4.0.3	Android Open Source Project	4.0.3	15

Project Components

- src – your source code
- gen – auto-generated code (usually just R.java)
- Included libraries
- Resources
  - Drawables (like .png images)
  - Layouts
  - Values (like strings)
- Manifest file

XML

- Used to define some of the resources
  - Layouts (UI)
  - Strings
- Manifest file
- Shouldn’t usually have to edit it directly, Eclipse can do that for you
- Preferred way of creating UIs
  - Separates the description of the layout from any actual code that controls it
  - Can easily take a UI from one platform to another

R Class

- Auto-generated: you shouldn't edit it
- Contains IDs of the project resources
- Use findViewById and Resources object to get access to the resources
  - Ex. Button b = (Button)findViewById(R.id.button1)
  - Ex. getResources().getString(R.string.*hello*));

Layouts (1)

- Eclipse has a great UI creator
  - Generates the XML for you
- Composed of *View* objects
- Can be specified for portrait and landscape mode
  - Use same file name, so can make completely different UIs for the orientations without modifying any code

Layouts (2)

Java - SampleAndroid/res/layout/main.xml - Eclipse

File Edit Run Navigate Search Project Refactor Window Help

Package Explorer

- SampleAndroid
  - src
    - oose.sample
      - ButtonListener.java
      - SampleAndroidActivity.java
      - SecondAct.java
  - gen [Generated Java Files]
    - oose.sample
      - R.java
  - Android 2.2
  - assets
  - res
    - drawable-hdpi
    - drawable-ldpi
    - drawable-mdpi
    - layout
    - layout-land
      - main.xml
      - secondact.xml
    - layout-port
      - main.xml
      - secondact.xml
    - values
  - AndroidManifest.xml
  - default.properties
  - proguard.cfg

SampleAndroidActivit | SecondAct.java | main.xml | *secondact.xml | ButtonListener.java

Editing config: default | Any locale | Android 2.2 | Create...

3.7in WVGA (Nexus One) | Portrait | Normal | Day time | Theme

Palette

- Form Widgets
- Text Fields
- Layouts
- Composite
- Images & Media
- Time & Date
- Transitions
- Advanced
- Custom & Library Views

Graphical Layout | main.xml

Problems | Javadoc | Declaration | Console

Android

Layouts (3)

- Click ‘Create’ to make layout modifications
- When in portrait mode can select ‘Portrait’ to make a res sub folder for portrait layouts
  - Likewise for Landscape layouts while in landscape mode
  - Will create folders titled ‘layout-port’ and ‘layout-land’
- Note: these ‘port’ and ‘land’ folders are examples of ‘alternate layouts’, see here for more info
  - http://developer.android.com/guide/topics/resources/providing-resources.html
- Avoid errors by making sure components have the same id in both orientations, and that you’ve tested each orientation thoroughly

Strings

- In res/values
  - strings.xml
- Application wide available strings
- UI components made in the UI editor should have text defined in strings.xml
- Strings are just one kind of ‘Value’ there are many others

Manifest File

- The application must declare all its components in this file, which must be at the root of the application project directory.
- The manifest does a number of things in addition to declaring the application's components, such as:
  - Identify any user permissions the application requires, such as Internet access or read-access to the user's contacts.
  - Declare the minimum API Level required by the application, based on which APIs the application uses.
  - Declare hardware and software features used or required by the application, such as a camera, bluetooth services, or a multitouch screen.
  - API libraries the application needs to be linked against (other than the Android framework APIs), such as the Google Maps library.
  - And more

Manifest File

- **Declaring components**
  - You must declare all application components this way:
  - <activity> elements for activities
  - <service> elements for services
  - <receiver> elements for broadcast receivers
  - <provider> elements for content providers

Manifest File – Adding an Activity

Java - SampleAndroid/AndroidManifest.xml - Eclipse

File Edit Run Navigate Search Project Refactor Window Help

Package Explorer

- SampleAndroid
  - src
    - oose.sample
      - ButtonListener.java
      - SampleAndroidActivity.java
      - SecondAct.java
  - gen [Generated Java Files]
    - oose.sample
  - Android 2.2
  - assets
  - res
    - drawable-hdpi
    - drawable-ldpi
    - drawable-mdpi
    - layout
    - layout-land
    - layout-port
    - values
  - AndroidManifest.xml
  - default.properties
  - proguard.cfg

SampleAndroidActivity.java | SampleAndroid Manifest | SecondAct.java

Label	@string/app_name	Browse...	Vm safe mode	
Icon	@drawable/icon	Browse...	Manage space activity	Browse...
Description		Browse...	Allow clear user data	
Permission			Test only	
Process		Browse...	Backup agent	Browse...
Task affinity		Browse...	Allow backup	
Allow task reparenting			Kill after restore	
Has code			Restore needs application	
Persistent			Restore any version	

Application Nodes

- .SampleAndroidActivity
- SecondAct (Activity)

Add... Remove... Up Down

Attributes for Activity

The activity tag declares an **android.app.Activity** class that is available as part of the package's application components, implementing a part of the application's user interface.

Name* SecondAct Browse...

Theme Browse...

Label Browse...

Manifest | Application | Permissions | Instrumentation | AndroidManifest.xml

Problems | Javadoc | Declaration

0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				

Three Main Approaches to build GUI

- **Java-based**

  Use Java to define Strings, lay out window, create GUI controls, and assign event handlers. Like Swing programming.
- **XML- based**

  Use XML files to define Strings, lay out window, create GUI controls, and assign event handlers. The Java method will read the layout from XML file and pass it to setContentView.
- **Hybrid**

  Use an XML file to define Strings, lay out window and create GUI controls. Use Java to assign event handlers.

Java-Based Approach: Template

```
public class SomeName extends Activity {
    @Override
    public void onCreate(Bundle
        savedInstanceState) {
            super.onCreate(savedInstanceState);
            String message = "...";
            LinearLayout window = new LinearLayout(this);
            window.setVariousAttributes(…);
            Button b = new Button(this);
            b.setText("Button Label");
            b.setOnClickListener(new SomeHandler());
            mainWindow.addView(b);
            ...
            setContentView( window );
    }
    private class SomeHandler implements
        OnClickListener {
            @Override
            public void onClick(View clickedButton ){
                doSomething(...);
            }
    } }
```

XML-Based Approach: Template

Java

```
public class SomeClass extends Activity {
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
        public void handlerMethod (View clickedButton) {
                String someName = getString (R.string.some_name);
                doSomethingWith(someName);
        } }
```

XML-Based Approach: Template

XML

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="some_name">…</string>
…
</resources>
```

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout …>
<TextView … />
<Button …
    android:onClick="handlerMethod" />
</LinearLayout>
```

Hybrid Approach: Template

Java

```
public class SomeClass extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button b = (Button)findViewById(R.id.button_id);
        b.setOnClickListener(new SomeHandler());
    }
    private class SomeHandler implements OnClickListener {
        @Override
        public void onClick(Vie w clickedButton) {
            doSomething (...);
        }
    } }
```

XML

- Controls that need handlers are given IDs
- You ***do not*** use android:onClick to assign handler

Basic widgets

- **TextView**
  - TextView allocates an area of the screen to display text

```
<TextView android:id="@+id/counter“
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:padding="10dp"
    android:text="@string/sample_time"
    android:textSize="50sp" >
</TextView>
```

Basic widgets

- **EditText**
  - An EditText is a TextView that is configured to allow the user to edit the text inside it
  - inputType to change the behavior of the EditText:

```
<EditText android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="phone" />
```

Buttons

- Represents a push-button widget.
- Push-buttons can be pressed, or clicked, by the user to perform an action.

```
<Button android:id="@+id/finished"
    android:layout_width="match_parent“
    android:layout_height="wrap_content“
    android:text="@string/finished" >
</Button>
```

Boolean Buttons

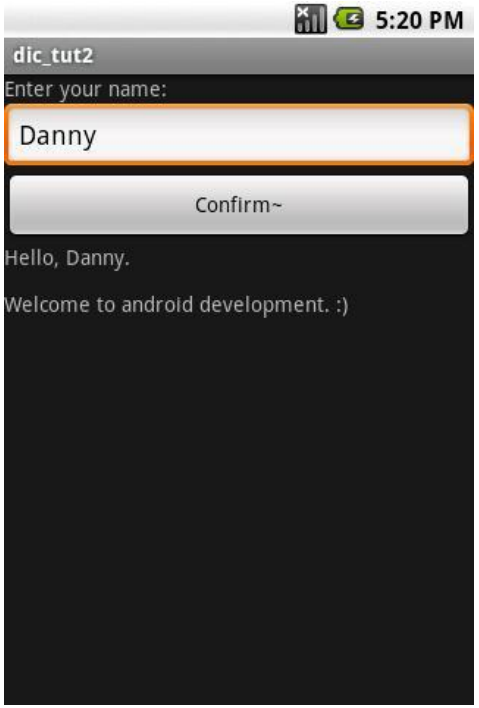
- Buttons are convenient for indicating on/off states.
- Android has a number of views, including toggle buttons, checkboxes, and radio buttons.

Basic widgets

- ImageView
  - The ImageView class can load images from various sources (such as resources or content providers)

Homework

- Create a application as following Picture by using XML based appoach and java based appoach



Thank you!