

Android: Resources and Intents

31 Mar 2010

CMPT166

Dr. Sean Ho

Trinity Western University

XML layout

- Laying out widgets can be **complex** in code
- You may use an **XML config** file for your layouts:
 - Create a file under **res/layout/*.xml**
 - XML is like HTML: **<tag> ... </tag>**
- Specify **layouts**, **widgets**, font/colour/text/etc.
 - Eclipse ADT has a WYSIWYG **layout editor!**
- The XML layout gets **compiled** into an object in the **R class** (auto-generated; don't edit directly!)
 - Refer to **R.layout.myLayout**
(follows **name** of the XML file)

Layout editor

```
HelloAndroid.java *main.xml strings.xml dimens.xml
<?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" />

    <DigitalClock android:text="@+id/DigitalClock01" android:id="@+id/DigitalClock01" android:layout_height="wrap_content" and
    </DigitalClock>
    <EditText android:id="@+id/EditText01" android:layout_height="wrap_content" and
        android:textSize="@dimen/text_size" and
        android:textStyle="italic"></EditText>
</LinearLayout>
```

HelloAndroid.java *main.xml strings.xml dime

Editing config: default

Devices ADP1 Config Portrait Locale A Theme

Layouts

- AbsoluteLay...
- DialerFilter
- Expandable...
- FrameLayout
- GridView
- HorizontalSc...
- ImageSwitc...
- LinearLayout

Views

- SurfaceView
- View
- ViewStub
- AnalogClock
- AutoComple...
- Button
- CheckBox
- CheckedText...

Hello, Android user! The time is:
10:21:39 am

Type a message here!

XML view

WYSIWYG view

Referring to resources

- In the XML layout, the first TextView widget has a default ID: `@+id/TextView01`
 - `@`: resource ID (instead of literal value)
 - `+`: create this resource ID if it doesn't exist
- Change the widget's ID by editing **Property/ID**:
 - e.g., `@+id/top_label`
- Refer to this widget in the **code** using its ID:
 - `final TextView label = (TextView) findViewById(R.id.top_label);`
 - `label.setText("This text was set by code!");`

Text resources and i18n

- “i18n”: **Internationalization**: single software that can be deployed in many countries
- “L10n”: **Localization**: adapting the software for local language, formats, etc.
- Separate all **localizable strings** into separate **file**
 - Dialogue text, labels, etc.
 - **Default** strings file: `res/values/strings.xml`
- String **resources**: **name/value** pairs
 - Refer to `@string/name`
 - Use a **string resource** as the **text** of a **widget**

Drawable resources

- Drawables include images, icons, animation sequences, etc.
- Store PNGs, etc. under `drawable/` directory
- Refer to via `@drawable/filename` (w/o ext)
 - In properties: `@drawable/filename`
 - From code, use `getResourceById()` to get a reference to the object (cast as needed):
 - ◆ `getResourceById(R.drawable.filename);`
- All resources are packaged together with your compiled code: one distributable application

Alternate resources

- **Alternate** resource directories may be used depending on the device's **locale**, screen **res**, supported **hardware**, etc.:
- **res/values-fr/strings.xml**: **French** strings
- **res/drawable-hdpi/**: high-**pixel-density** images
- **Qualifiers**: Cell **network** (MCC/MNC), **language**, **region** (**en-CA**), phys. screen **size**, **orientation**, **pixel density**, **touchscreen** type, etc.

Adding event listeners

■ Buttons have `OnClickListeners`:

- `import android.view.View.OnClickListener;`

- `import android.widget.Button;`

- ◆ (**Resource ID** need not be same as **var name**)

- `final Button clickMe = (Button)
 findViewById(R.id.clickMe);`

- ◆ Anon. **inner** class, anon object:

- `clickMe.setOnClickListener(new OnClickListener() {
 public void onClick(View v) {
 // do stuff when button is clicked
 }
});`

Intents

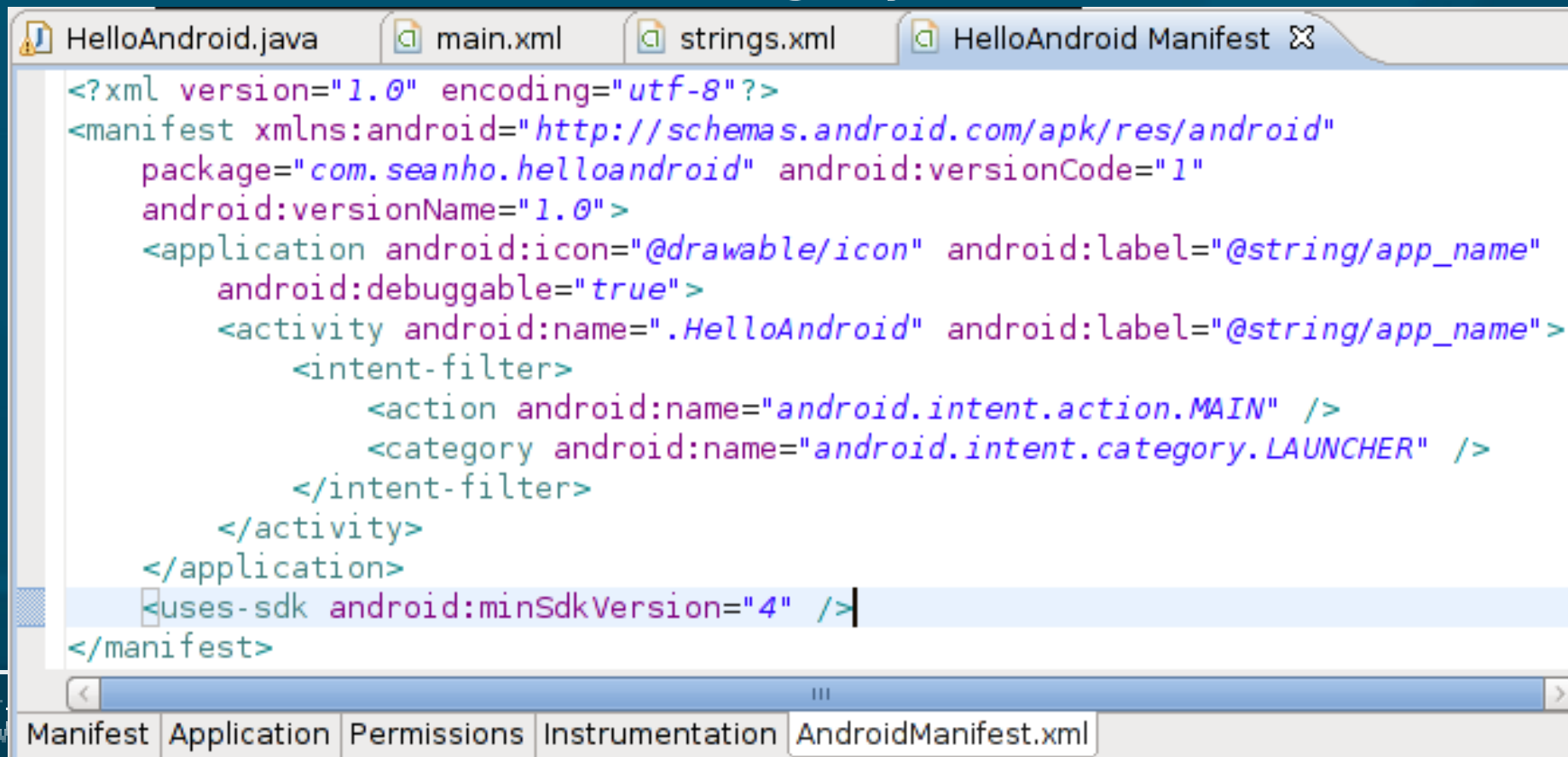
- Activities (and Services, etc.) are triggered by Intents: system-wide messages/events
- The “glue” that connects together components
- An Intent may include:
 - Target: package and component (Activity)
 - Action: what the target should do
 - Data: URI and MIME type
 - Category: home, launcher, preference, etc.

Implicit intents and filters

- An **implicit** intent does not specify a particular **target** component (Activity)
- Android matches **action**, **data**, and **category** against a component's **intent filter** to figure out which component to send the message to
- e.g., to launch **web browser**:
 - Sets **action** and **data**:
 - **Intent browse = new Intent(Intent.VIEW_ACTION, Uri.parse("http://www.google.com/"));**
 - **startActivity(browse);**

AndroidManifest.xml

- Declare an activity's intent filters in **manifest**:
- e.g., make it **launchable** from the home screen:
 - Action: **MAIN**. Category: **LAUNCHER**



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.seanho.helloandroid" android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name"
        android:debuggable="true">
        <activity android:name=".HelloAndroid" 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>
    <uses-sdk android:minSdkVersion="4" />
</manifest>
```