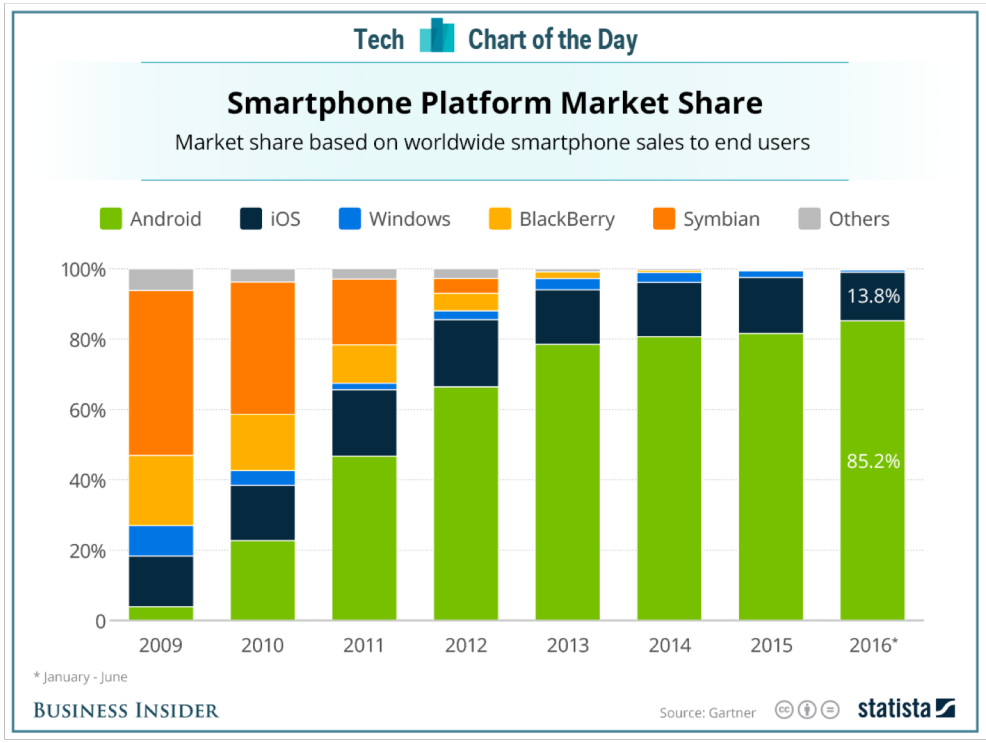
EECS 448 Smartphone Security
Android Programming

Xusheng Xiao

Electrical Engineering and Computer Science
Case Western Reserve University

Last Lecture: Android Basics

- Architecture
- App Components
- Activity Lifecycle
- Inter-Component Communications



Environment Setup

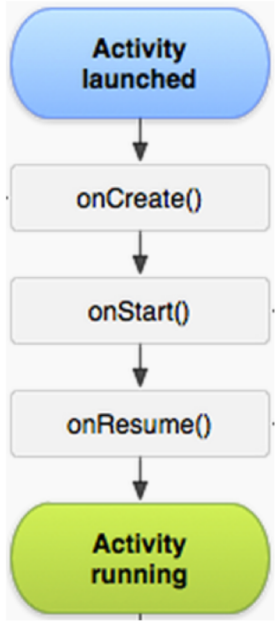
- Android Studio
  - https://developer.android.com/sdk/index.html
- Oracle Java 7 JDK
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Verify Android Studio is Installed
  - Open Android Studio Check for Updates
  - Download and install the latest updates

App Components

- Essential building blocks of an Android app
  - An entry point for the system or a user to enter the app
- Four different types of app components:
  - Activities: a single screen with UI
  - Services: long-running operations in the background or services for remote processes
  - Broadcast receivers: responding to system-wide broadcast events
  - Content providers: shared data accesses

Activity

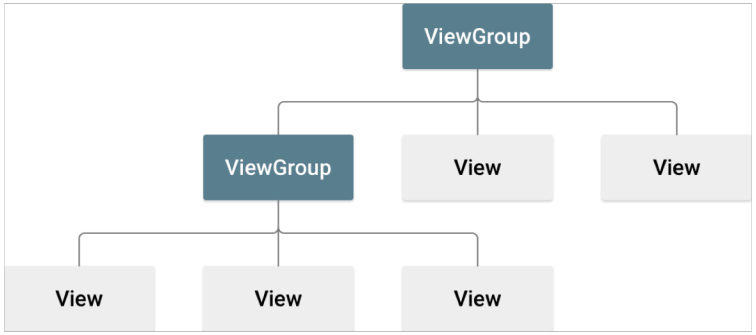
- Responsible for managing user interaction with a screen of information
- Write subclasses of Activity
  - Extend Activity or a subclass of Activity
- Contains a lifecycle
- Activities can start other activities



- Launcher Activity - activity started on launch when a user opens your app

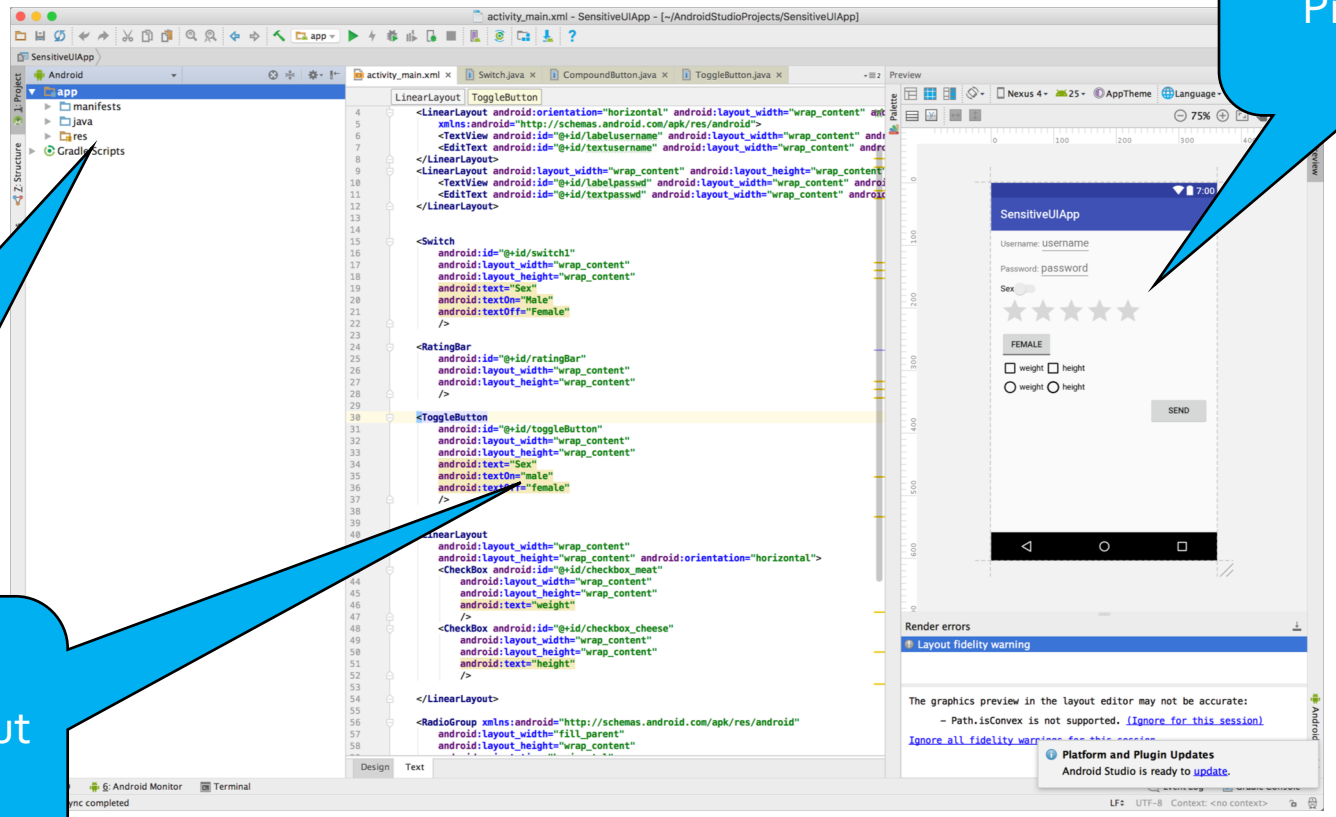
UI Layout for an Activity

- Defines a set of user interface objects and their position on the screen.
- Made up of definitions written in XML
  - Extensible Markup Language
- Definitions are used to create objects that appear on a screen



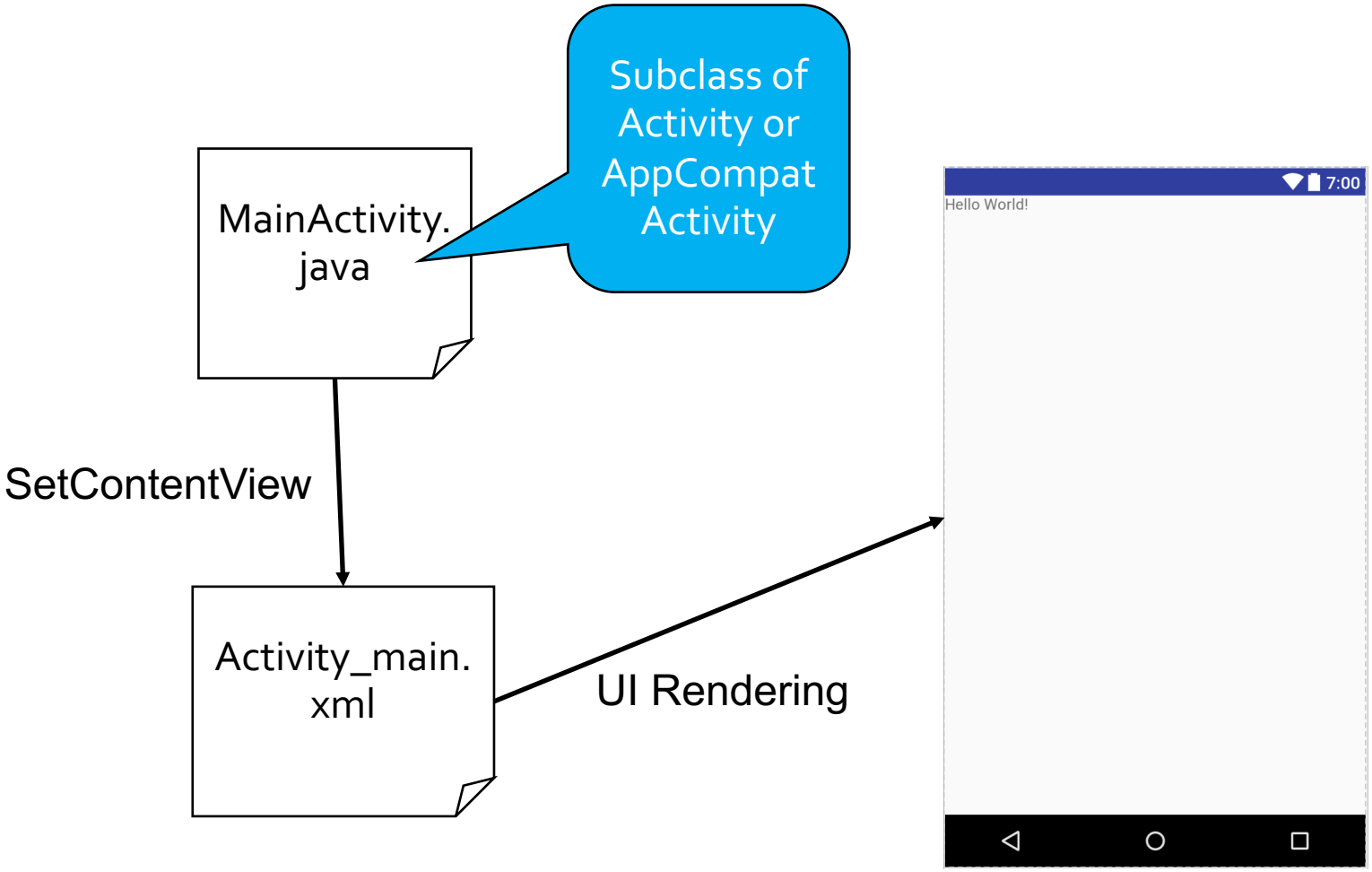
- Viewgroups and views/widgets
  - Viewgroups hold/layout the other views

Android Studio



https://developer.android.com/studio/index.html

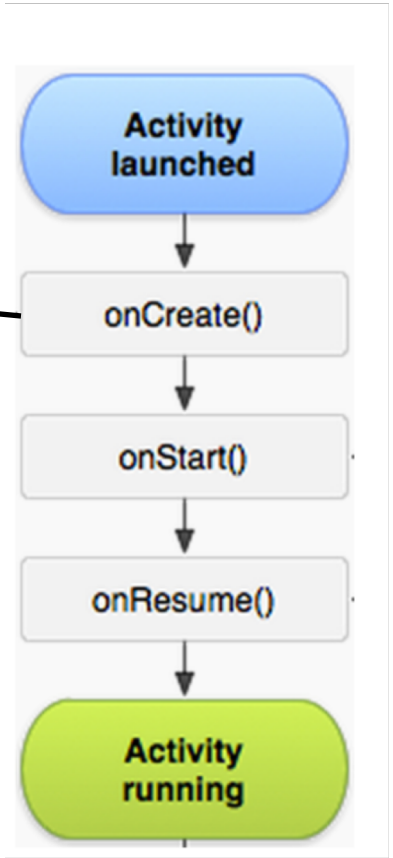
Relationship btw Code and UI



Hello World Class

```
public class MainActivity extends AppCompatActivity {

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

Layout file resource: activity_main.xml



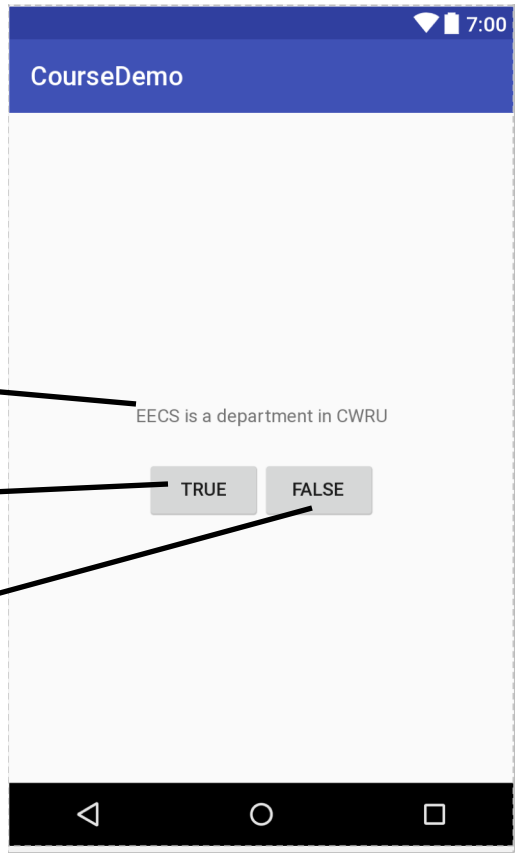
Hello World UI Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!"
       />

</RelativeLayout>
```

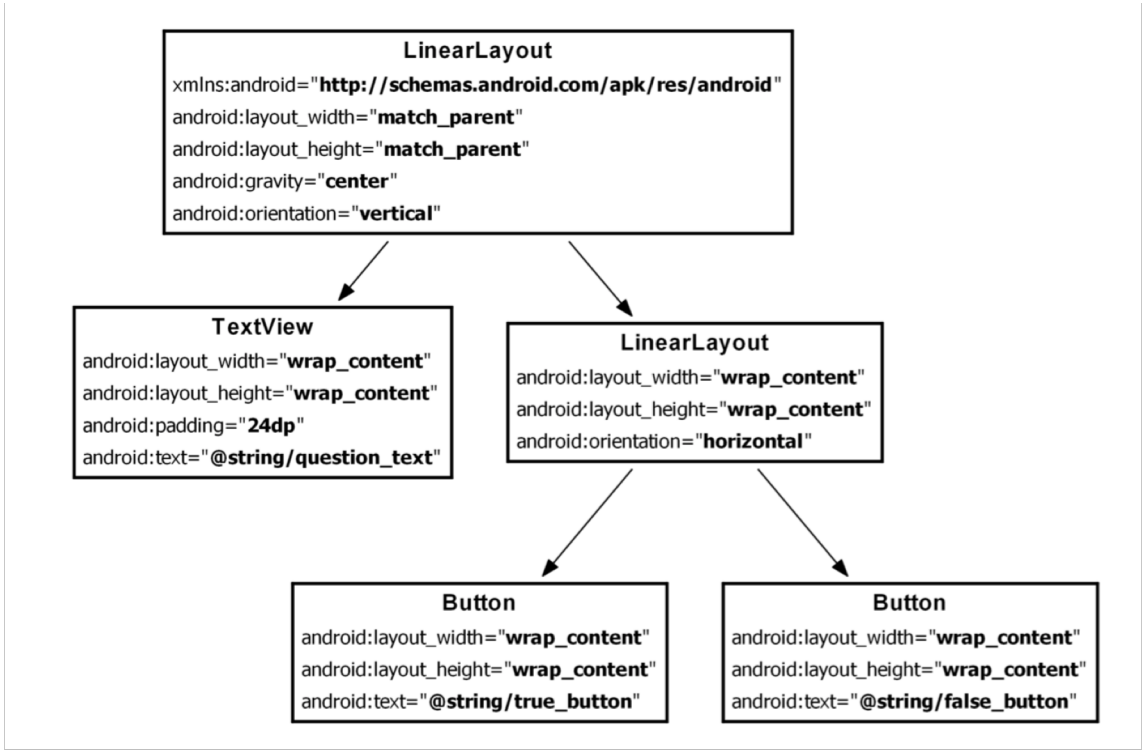
- **Layouts**: arranging UI objects automatically
- **Widgets**: building blocks to compose a user interface.
    - Show text or graphics
    - Buttons, text input controls, and checkboxes

An Updated UI Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="center"
  android:orientation="vertical" >
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/question_text" />
  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/true_button" />
    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/false_button" />
  </LinearLayout>
</LinearLayout>
```



View Hierarchy



- LinearLayout: vertical and horizontal, match_parent, center
- TextView: wrap_content, @string/question_text
- Button: @string/true_button, @string/false_button

String Resources

- A string that lives in a XML file
  - Location: app/res/values/strings.xml

```
<resources>
    <string name="app_name">CourseDemo</string>
    <string name="action_settings">Settings</string>
    <string name="question_text">EECS is a department in CWRU</string>
    <string name="true_button">True</string>
    <string name="false_button">False</string>
</resources>
```

- Widget can have a hard-coded string
  - E.g., android:text="True"

Resources and Resource IDs

- Types of resources
  - Image files, audio files, and XML files
  - Location: res

- Resource ID to access specific resources
  - Providing a way to access the resource in code
    - E.g., R.layout.activity_main
  - Mapping a resource file to an integer value
    - Automatically generated in a R.java

Example R File

```
public final class R {
        public static final class anim { … }
        …
        public static final class id { … }
        public static final class layout { …
                public static final int activity_main=0x7f04001c;
        }
        …
}
```

```
public class MainActivity extends AppCompatActivity {

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

Resource IDs for Widgets

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="center"
  android:orientation="vertical" >
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/question_text" />
  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/true_button"
      android:id="@+id/true_button"
/>
    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/false_button"
      android:id="@+id/false_button"
/>
  </LinearLayout>
</LinearLayout>
```

```
public static final class id {
        …
        public static final int true_button=0x7f0d007a;

        public static final int false_button=0x7f0d007b;
        …
}
```

- android:id attribute
  - Unique value
  - "+" to automatically generate new ids

Manipulating Widgets

```
public class GeoQuizActivity extends AppCompatActivity {

    private Button mtruebtn;
    private Button mfalsebtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_geo_quiz);

        mtruebtn = (Button) findViewById(R.id.true_button);
        mfalsebtn = (Button) findViewById(R.id.false_button);


        mtruebtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // Does nothing
                }
        });


    }
}
```

- Accessing widgets using IDs
  - IDs automatically generated in R.java
  - findViewById

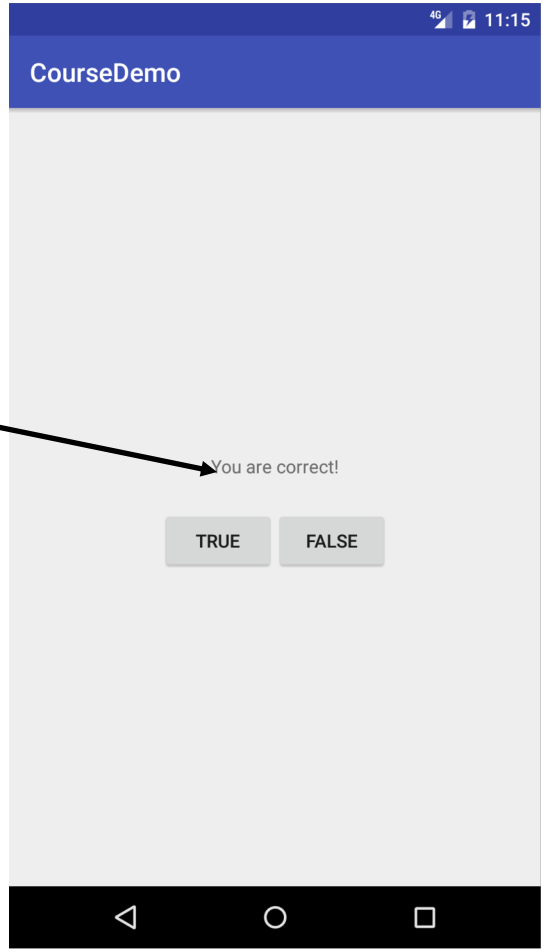
- Adding event handlers
  - Various events to add handlers
    - onClick, onTouch
    - Executed when the events occur

Handler Task: Making Toasts

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_geo_quiz);

    mtruebtn = (Button) findViewById(R.id.true_button);
    mfalsebtn = (Button) findViewById(R.id.false_button);

    mtruebtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast toast = Toast.makeText(GeoQuizActivity.this,
R.string.correct_toast, Toast.LENGTH_SHORT);
            toast.show();
        }
    });


    mfalsebtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast toast = Toast.makeText(GeoQuizActivity.this,
R.string.incorrect_toast, Toast.LENGTH_SHORT);
            toast.show();
        }
    });

}
```

Handler Task: Change Text View

```
mtruebtn = (Button) findViewById(R.id.true_button);
mfalsebtn = (Button) findViewById(R.id.false_button);
mquestiontext = (TextView) findViewById(R.id.question_text);

mtruebtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast toast = Toast.makeText(GeoQuizActivity.this,
R.string.correct_toast, Toast.LENGTH_SHORT);
        toast.show();
        mquestiontext.setText("You are correct!");
    }
});
```



- Model-View-Controller
  - Model: mquestiontext
  - View: TextView
  - Controller: OnClickListener()

Starting a New Activity

- Creating an other activity
  - Layout files and controller code (e.g., SecondActivity)
  - Registering the activity in the manifest file
- Specifying a new activity to start
  - Using an intent with the class name of the activity
  - Passing data via the intent

```
public void onclick(View view) {
    Intent intent = new Intent(this, SecondActivity.class);
    intent.putExtra("key","value");
    startActivity(intent);
}
```

Activities in Manifest File

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/androi
d"
   package="com.example.xushengxiao.coursedemo">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity
         android:name=".MainActivity"
         android:label="@string/app_name"
         android:theme="@style/AppTheme.NoActionBar">
         <intent-filter>
            <action
android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity android:name=".SecondActivity"></activity>
   </application>

</manifest>
```

- Default activity
  - The launching activity
- Second activity
  - The one to start

Services

- Service
  - Run in your application's main thread by default
  - May slow the performance of any activity
- IntentService
  - Run in a worker thread to handle all of the start requests, one at a time.

Toast Service

```
public class ToastService extends IntentService {

    public ToastService() {   super("ToastService");    }

    @Override
    protected void onHandleIntent(final Intent intent) {
        new Handler(Looper.getMainLooper()).post(new
Runnable() {
            @Override
            public void run() {
                Toast toast1 = Toast.makeText(ToastService.this,
intent.getStringExtra("msg"), Toast.LENGTH_LONG);
                toast1.show();
            }
        });
    }
}
```

- Implement the onHandleIntent() to handle received intent
  - Retrieve extra data from the intent using getXXXExtra

```
<manifest   package="…">
    <application>
        …
        <service
            android:name=".ToastService"
            android:exported="false"></service>
    </application>

</manifest>
```

- Register the service in the manifest file

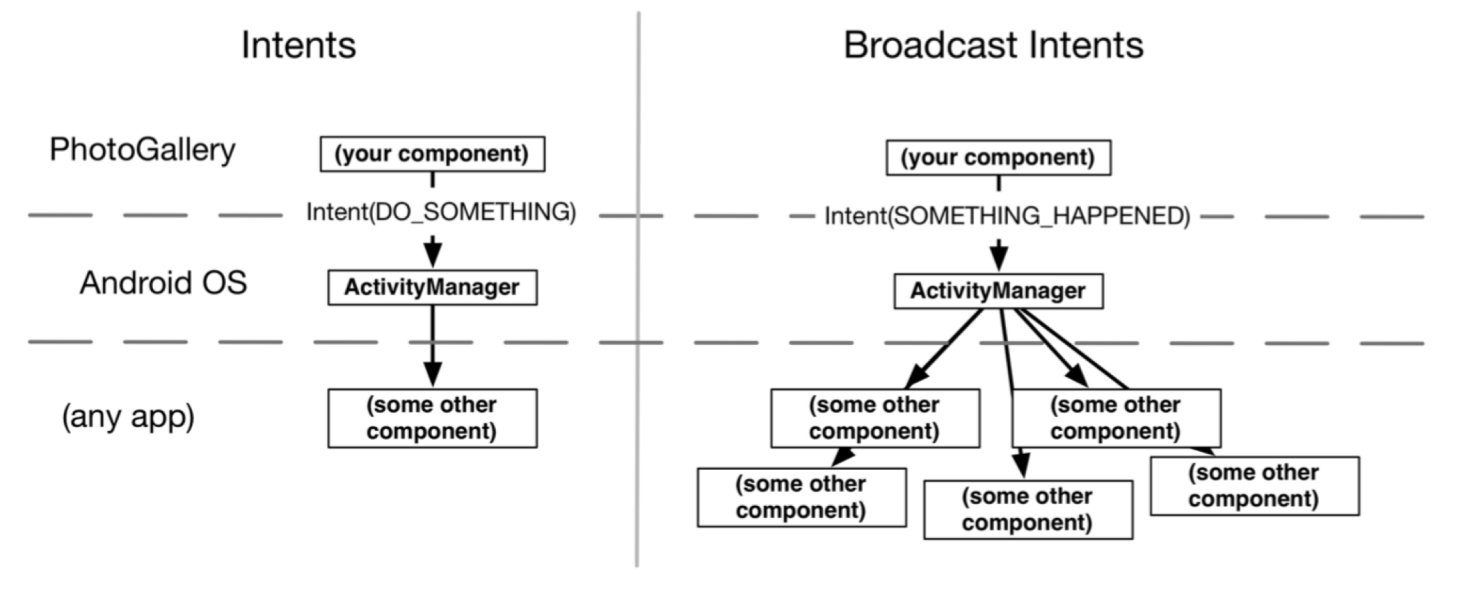
Starting a Service

```
Intent intent = new Intent(this,ToastService.class);
intent.putExtra("msg","Hello World!");
startService(intent);
```

- Create an intent with the service's class
- Add messages to the intent
- Invoke startService() to start the service
- IntentService stops after executing the onHandleIntent

Broadcast Intents

- Intents: only a component as the recipient
- Broadcast Intents: many components as recipients
  - Handled by broadcast receivers



Broadcast Receivers for System Events

```
public class StartupReceiver extends BroadcastReceiver{
    private static final String TAG = "StartupReceiver";
    @Override
    public void onReceive(final Context context, final Intent intent) {
        new Handler(Looper.getMainLooper()).post(new Runnable() {
            @Override
            public void run() {
                Toast toast1 = Toast.makeText(context, “BOOT!!”, Toast.LENGTH_LONG);
                toast1.show();
            }
        });
    }
}
```

```
<receiver
    android:name=".StartupReceiver"
    android:enabled="true"
    android:exported="true">

    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>

</receiver>
```

Sending Broadcast Intents

```
public class StartupReceiver extends BroadcastReceiver{
    private static final String TAG = "StartupReceiver";
    @Override
    public void onReceive(final Context context, final Intent intent) {
        new Handler(Looper.getMainLooper()).post(new Runnable() {
            @Override
            public void run() {
                Toast toast1 = Toast.makeText(context, intent.getStringExtra("msg"), Toast.LENGTH_LONG);
                toast1.show();
            }
        });
    }
}
```

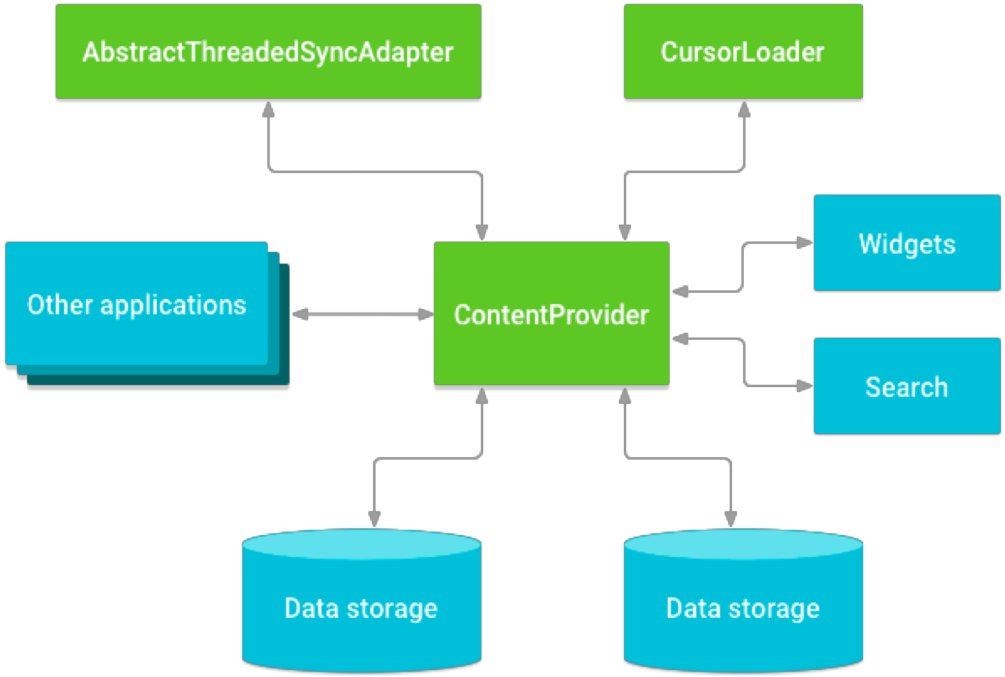
```
IntentFilter filter = new IntentFilter("test");
registerReceiver(new StartupReceiver(), filter);
```

```
public void onbroadcast(View view) {
    Intent intent = new Intent("test");
    intent.putExtra("msg", "broadcast");
    sendBroadcast(intent);
}
```

- Register a broadcast receiver to an action
- Send broadcasts using broadcast intent

Content Provider

- A content provider presents data to external applications as tables
  - Similar to the tables in a relational database



Creating a Content Provider

- URI Resolution: **"content://"** + ***AUTHORITY*** + **"/"** + BASE_PATH
  - Authority: internal name, e.g., com.example.app
  - Base path: table name, e.g., student
  - Patterns: matching URIs
    - content://com.example.app.provider/table1: A table
    - content://com.example.app.provider/*: any URI in the provider
    - content://com.example.app.provider/table3/6: row with id 6
- Data Manipulation Methods
  - Data sources: files, SQLite db, or Internet sources
  - CRUD: insert, query, update, delete

SQLite Database

- Lightweight relational database
  - Supports SQL
  - Model data as tables
  - Default DBMS supported by Android

word	app id	frequency	locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5

SQLite Helper

```
public class StudentDBHelper extends SQLiteOpenHelper {

    private static final String CREATE_TABLE_TUTORIALS = "create table " + TABLE + " ("
+ ID + " integer primary key autoincrement, " + NAME
            + " text, " + PHONE + " text );";

    private static final String DB_SCHEMA = CREATE_TABLE_TUTORIALS;


    public static final String[] ALL_COLUMNS =
            {ID, NAME, PHONE};

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DB_SCHEMA);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w("Student database", "Upgrading database. Existing contents will be lost. ["
                + oldVersion + "]->[" + newVersion + "]");
        db.execSQL("DROP TABLE IF EXISTS " + "student");
        onCreate(db);
    }

    public StudentDatabase(Context context) {
        super(context, DB_NAME, null, 1);
    }
}
```

SQL to Create Table

Create Table when helper is created

Delete table and recreate during upgrade

Constructor to specify db name and version

Content Provider using SQLite

```
public class StudentContentProvider extends ContentProvider {
   private static final String AUTHORITY = "com.example.xushengxiao.coursedemo";
   private static final String BASE_PATH = "student";
   public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/" + BASE_PATH);
   private static final UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
   static {
      uriMatcher.addURI(AUTHORITY, BASE_PATH, 1);
      uriMatcher.addURI(AUTHORITY, BASE_PATH + "/#", 2);
   }
   SQLiteDatabase db;
   …
   @Override
   public boolean onCreate() {
      db = new StudentDatabase(getContext()).getWritableDatabase();
      return true;
   }

   @Override
   public Uri insert(Uri uri, ContentValues values) {
      long id = db.insert(StudentDatabase.TABLE, null, values);

      if (id > 0) {
         Uri _uri = ContentUris.withAppendedId(CONTENT_URI, id);
         getContext().getContentResolver().notifyChange(_uri, null);
         return _uri;
      }
      throw new SQLException("Insertion Failed for URI :" + uri);
   }
   …
}
```

URI Patterns

SQLite DB

Insert data into SQLite DB

Notify changes

Using a Content Provider

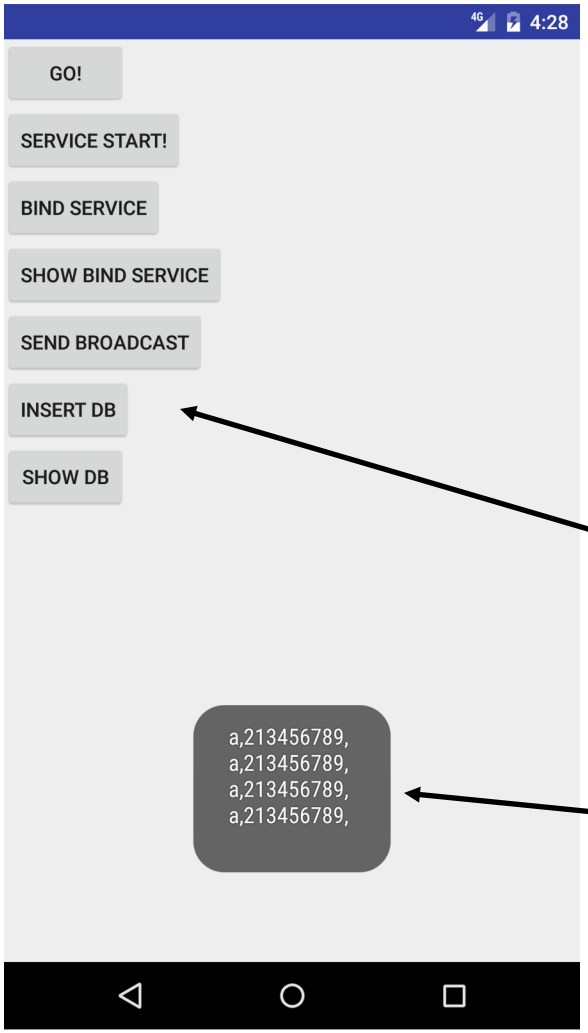
```
public void oninsertdb(View view) {
    ContentValues values  = new ContentValues();
    values.put(StudentDatabase.ID,new Random().nextInt(100));
    values.put(StudentDatabase.NAME,"a");
    values.put(StudentDatabase.PHONE,"213456789");

getApplicationContext().getContentResolver().insert(StudentCont
entProvider.CONTENT_URI,values);
}
```

```
public void onshowdb(View view) {
    Uri uri = StudentContentProvider.CONTENT_URI;
    Cursor cursor =
this.getContentResolver().query(uri,null,null,null,null);
    StringBuilder sb = new StringBuilder();
    while (cursor.moveToNext()){
        sb.append(cursor.getString(1) + ",");
        sb.append(cursor.getString(2) + ",");
        sb.append("\n");
    }


Toast.makeText(this,sb.toString(),Toast.LENGTH_LONG).show(
);
}
```

```
<provider
    android:name=".StudentContentProvider"
    android:authorities="com.example.app"
    android:enabled="true"
    android:exported="true"></provider>
```

- Registering a content provider in the manifest file
- Specifying a URI to use the content provider
  - Adding data using ContentValues
  - Fetching data using cursor

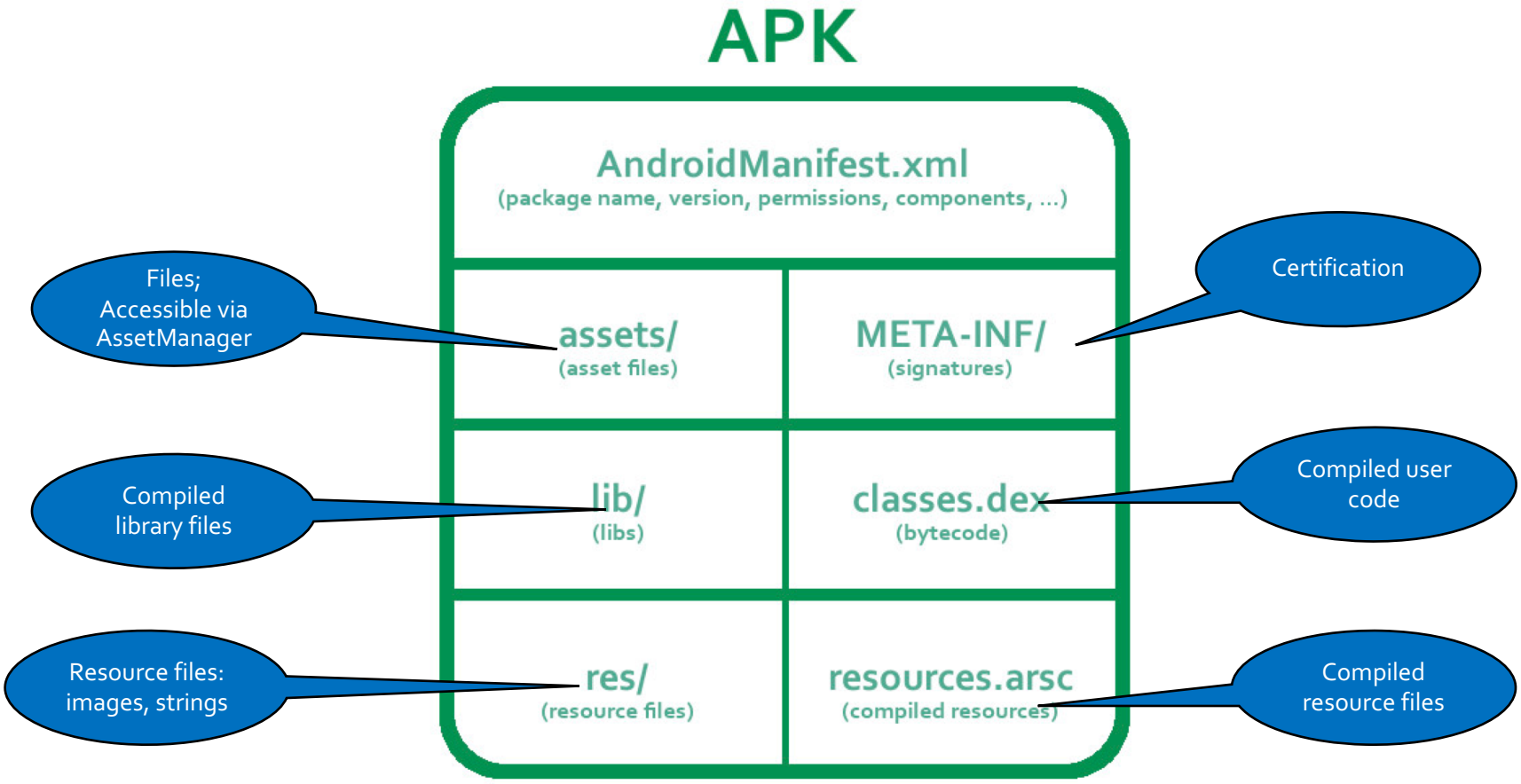
Example Content Provider




```
public void oninsertdb(View view) {
    ContentValues values  = new ContentValues();
    values.put(StudentDatabase.ID,new Random().nextInt(100));
    values.put(StudentDatabase.NAME,"a");
    values.put(StudentDatabase.PHONE,"213456789");

getApplicationContext().getContentResolver().insert(StudentCont
entProvider.CONTENT_URI,values);
}
```

- Insert record into db
- Show contents in db

Android APK Format



Thank You !

Questions ?