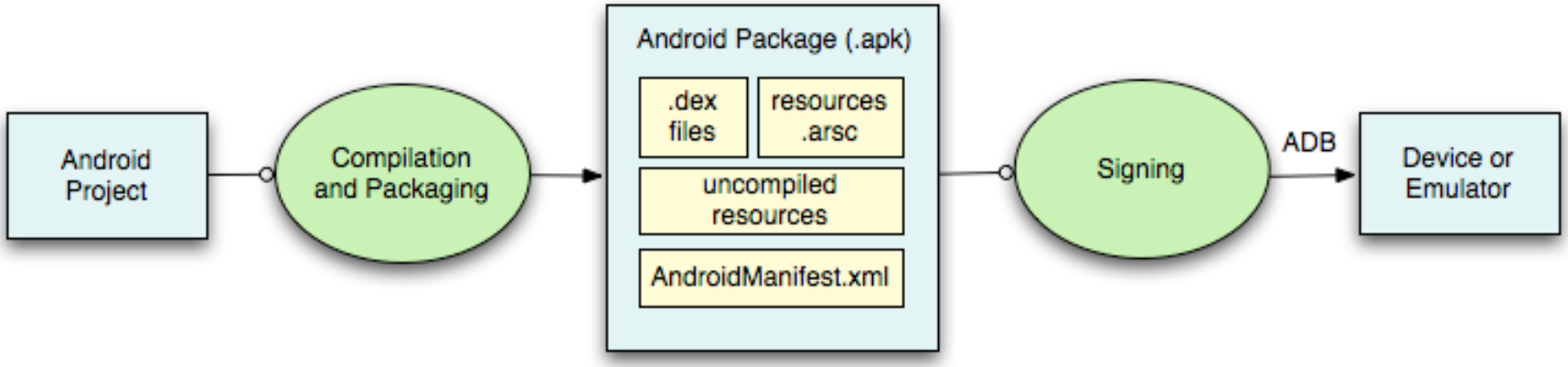
Application Fundamentals

Programming the Android Platform

CS 282
Principles of Operating Systems II
Systems Programming for Android

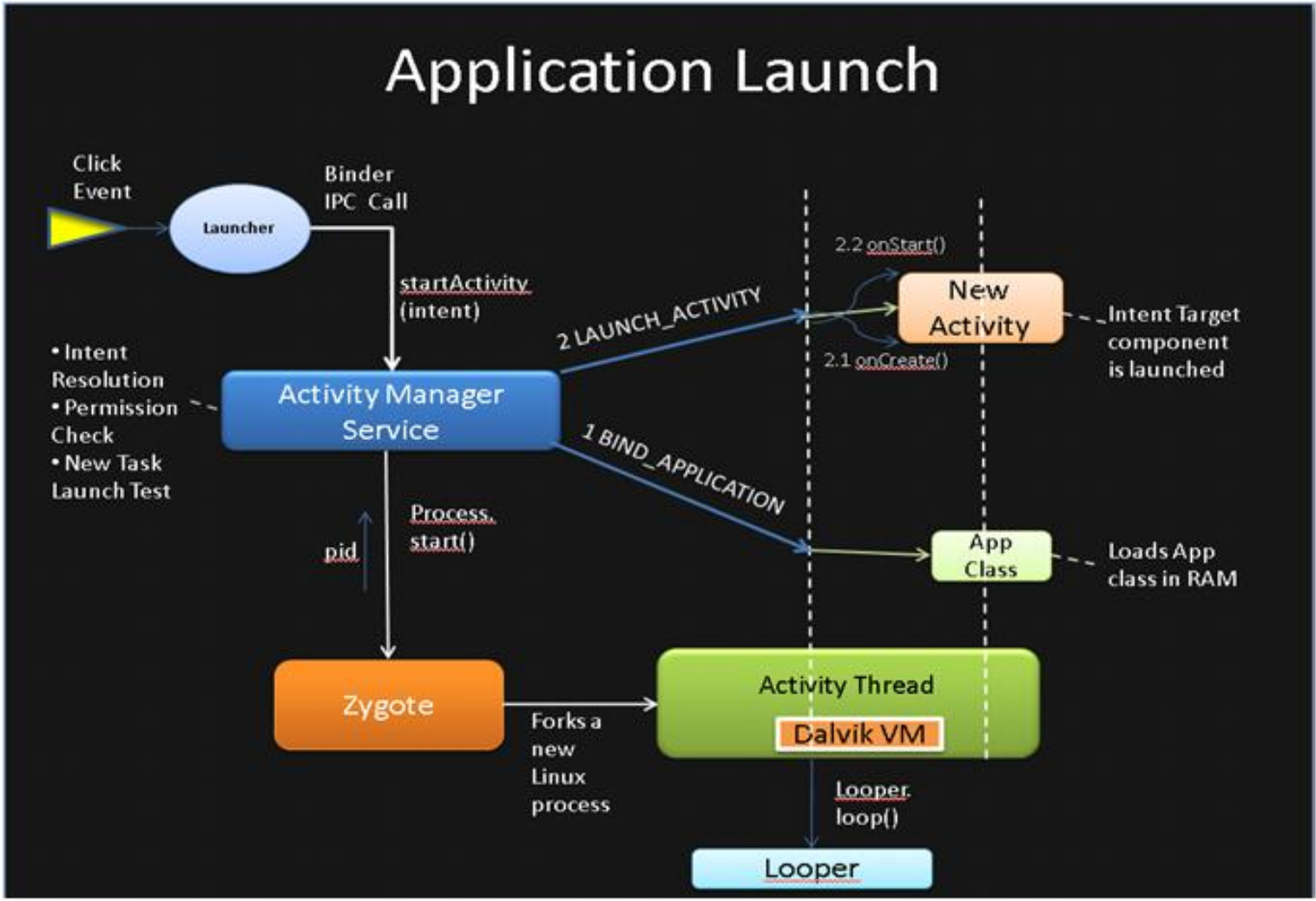
Application Development Steps



See: http://developer.android.com/guide/developing/building/index.html

Launching an Application

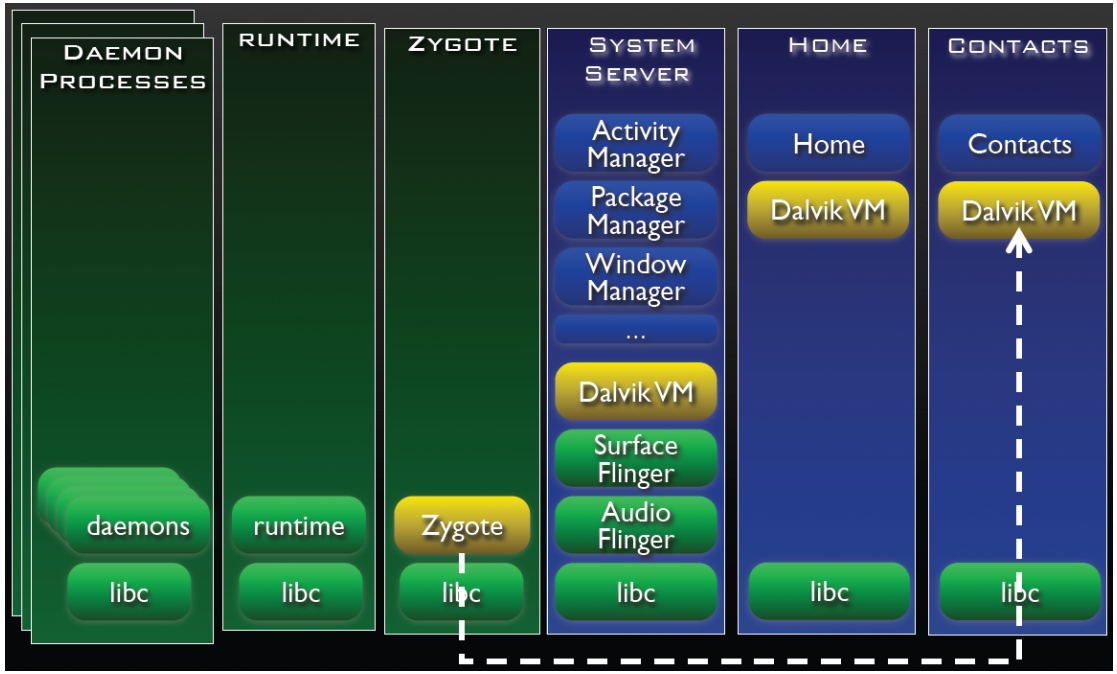
- All apps in Android are started via an Intent object
- The sole purpose of this intent is to notify the Activity Manager Service that the user wants something to happen



http://coltf.blogspot.com/p/android-os-processes-and-zygote.html

Running an Application

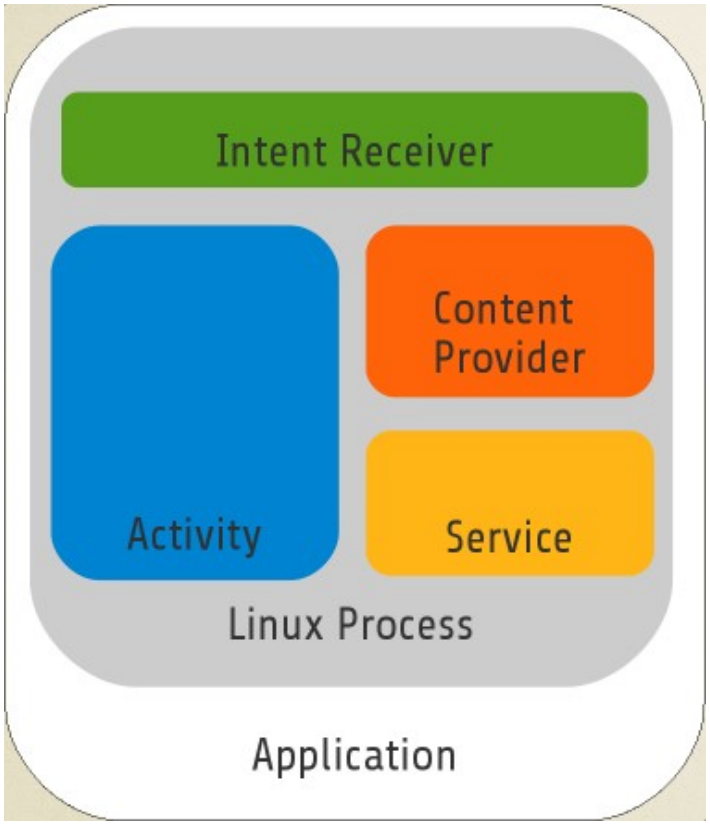
- By default, each application is assigned a unique Linux user ID & executes in its own Linux process
- By default, each process runs its own Dalvik VM
- Android manages process creation & shutdown
  - Starts process when any of the application's code needs to be run
  - Shuts down when process is no longer needed & system resources are required by other apps



https://sites.google.com/site/io/anatomy--physiology-of-an-android

Application Components

- Apps can have multiple entry points specified in their AndroidManifest.xml
  - i.e., not just main() method
- App comprises components that the system can instantiate & run as needed
- Key component classes include:
  - **Activities**
    - represents a single screen with a user interface
  - **Services**
    - runs in the background to perform long-running operations or to perform work for remote processes
  - **Content Providers**
    - manages a shared set of application data
  - **Broadcast Receivers**
    - a component that responds to system-wide broadcast announcements



Activities

- Primary class for interacting with user
  - Usually implements a focused task
  - Usually Involves one screenful of data
- You can use inherited methods on your Activity to get access to Android's system services:
  - GPS
  - Accelerometer
  - Phone
  - Launch other Activities
  - Camera
  - Media Framework
  - Etc.

```
<application android:name="PhoneApp"
      android:label="@string/phoneAppLabel"
      ...
    <activity android:name="EmergencyDialer"
     android:label="@string/emergencyDialerIconLabel"
     <intent-filter>
       <action android:name=
          "com.android.phone.EmergencyDialer.DIAL" />
       <category android:name=
            "android.intent.category.DEFAULT" />
     </intent-filter>
    </activity>
    <activity android:name="OutgoingCallBroadcaster"
      android:permission=
           "android.permission.CALL_PHONE"
     <intent-filter>
       <action android:name=
          "android.intent.action.CALL" />
       <category android:name=
          "android.intent.category.DEFAULT" />
       <data android:scheme="tel" />
     </intent-filter>
```

PhoneApp Activities

Service

- Runs in the background to perform long-running or remote operations
- Does not have a visual user interface
- Examples
  - MMS/SMS
  - Music player

MMS Services

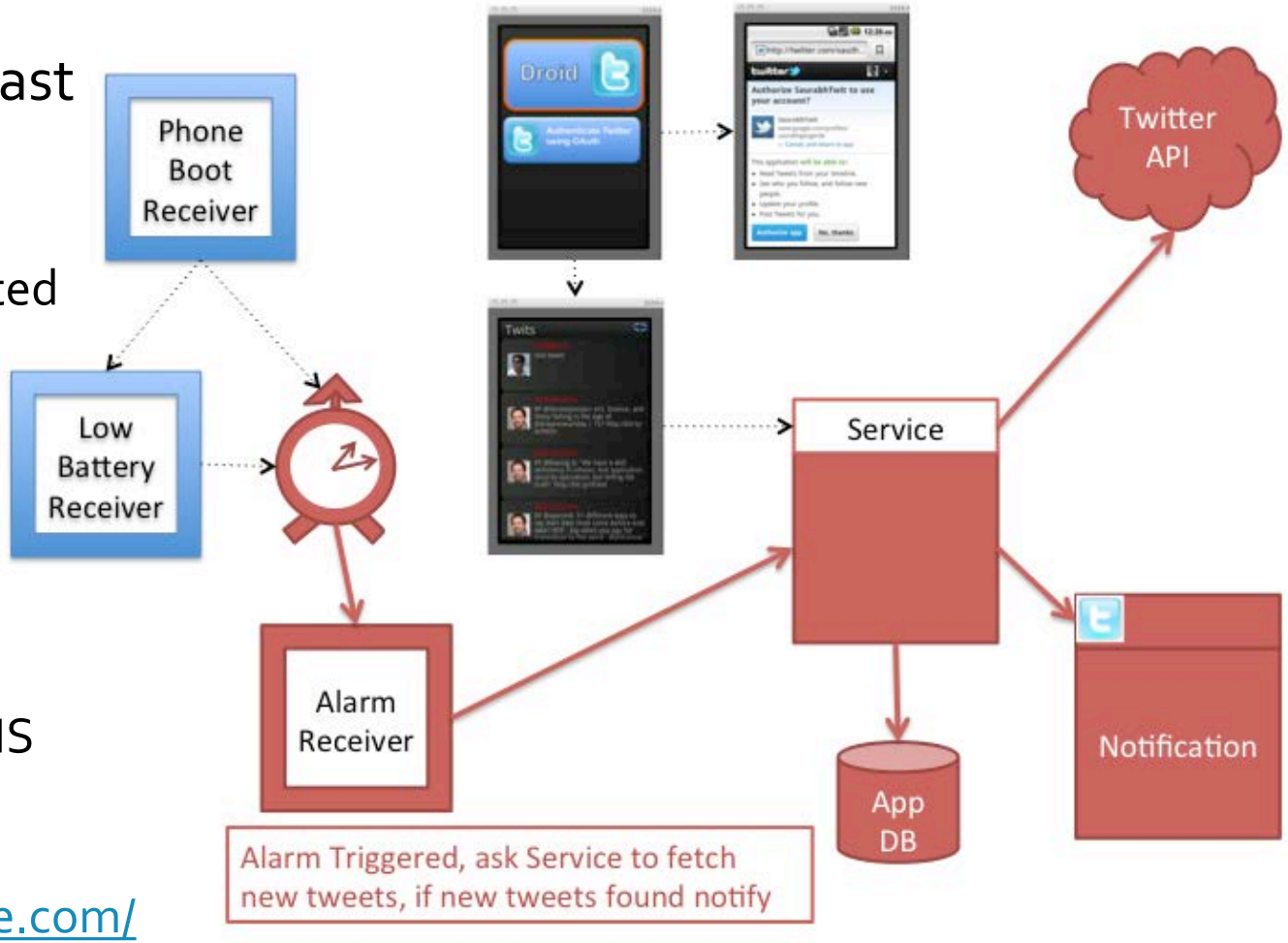
```
<service android:name=
        ".transaction.TransactionService"
         android:exported="true" />

    <service android:name=
         ".transaction.SmsReceiverService"
         android:exported="true" />
```

Music Service

```
<service android:name=
      "com.android.music.MediaPlaybackService"
       android:exported="false" />
```

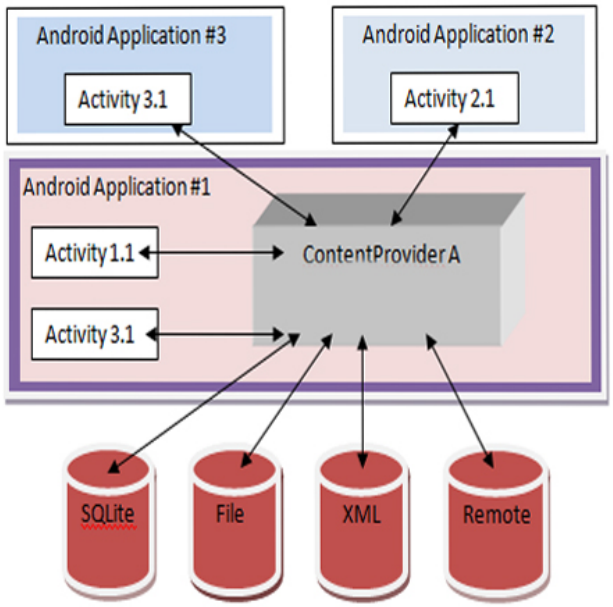
Broadcast Receiver

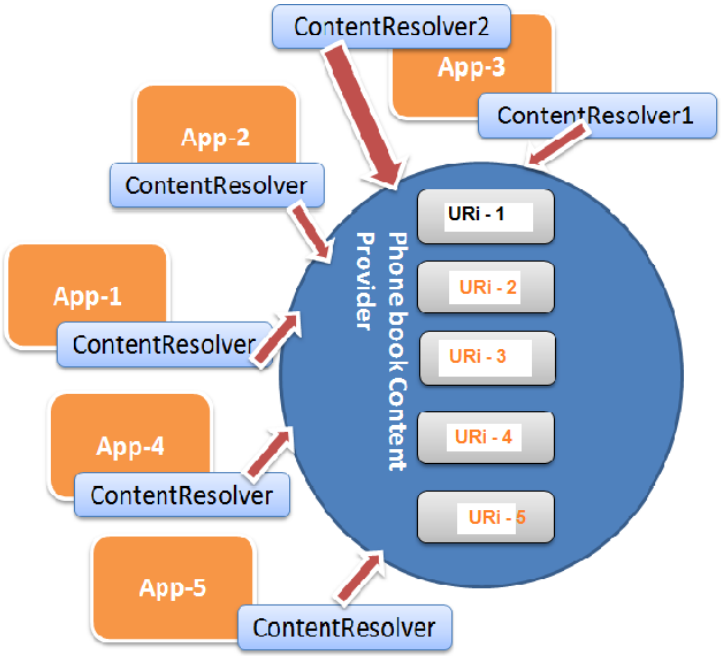
- Component that listens for broadcast announcements (events)
  - Events implemented as Intent instances
- Does not have a visual user interface
- Example
  - Messaging (on SMS receipt)
  - Droidtwit http://code.google.com/p/droidtwit/



Content Providers

- Store & retrieve data across applications
- Uses database-style interface
  - Can be implemented various ways (typically SQLite)





- Examples
  - Contacts
  - Browser history & bookmarks
  - App Launcher

A Simple Application

- MapLocation
  - User enters an address
  - App displays a map showing address

App Development

1. Define resources
2. Implement application classes
3. Package application
4. Install & run application

1. Defining Resources

- Several types of resources can be defined
  - Layout
  - Strings
  - Images
  - Menus
  - etc.
- See: http://developer.android.com/guide/topics/resources/index.html

Layout

- User interface layout specified in XML file
  - With Eclipse can also do layout visually (but beware of limitations)
- Stored in res/layout/ <filename>.xml
- Accessed from R.layout class

```
?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="wrap_content”
     android:layout_height="wrap_content”
  android:text="Enter Location”/>

<EditText android:id="@+id/location”
    android:layout_width="fill_parent”
     android:layout_height="wrap_content” />

<Button android:id="@+id/mapButton”
android:layout_width="wrap_content”
     android:layout_height="wrap_content”    android:text="Show Map”
     android:onClick=“showMap”  />
</LinearLayout>
```

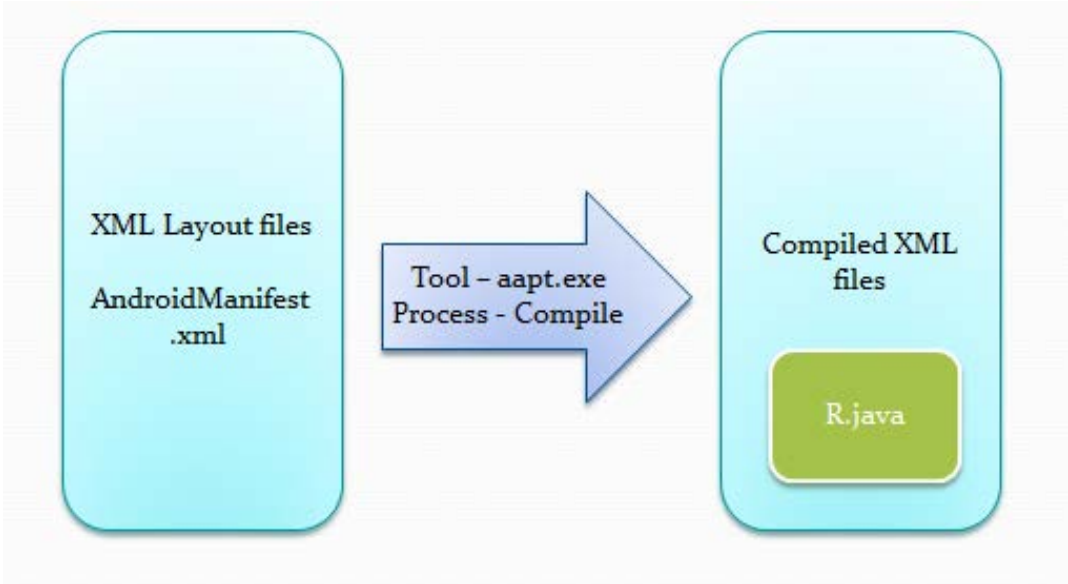
Strings

- Types
  - String
  - String Array
  - Plurals
- Can include style & formatting
- Stored in res/values/<filename>.xml
  - Each string specified as @string/string_name
- Accessed as R.string.string_name

```
<resources>

   <string name="app_name">Map Location</string>
   <string name="menu_settings">Settings</string>
   <string name=
     "title_activity_map_demo">MapLocation</string>
   <string name="location">Location:</string>
   <string name=
      "show_location">Show Location</string>
   <string name="lat">Latitude</string>
   <string name="lon">Longitude</string>

</resources>
```

R.java

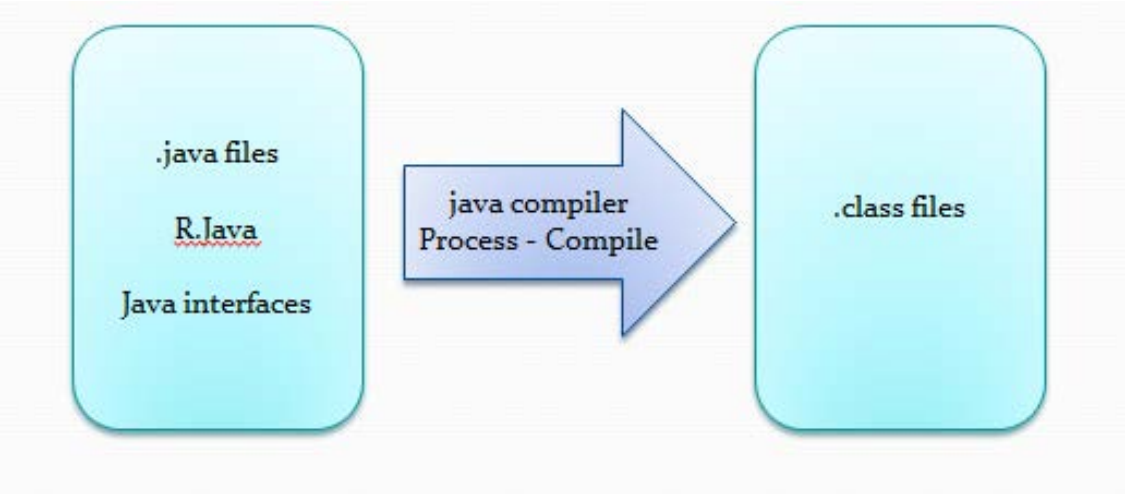
- At compilation time, resources are used to generate the R.java class
- Applications access resources through the R class
- Implements Resource Files & Packed Data patterns from http://smallmemory.com




```
public final class R {
   public static final class attr {
     }

   public static final class id {
      public static final int
     location=0x7f040000;
      public static final int
     mapButton=0x7f040001;
   }

   public static final class layout
     {
      public static final int
     main=0x7f030000;
   }
}
```

2. Implement Classes

- Usually involves at least one Activity
- Initialization usually in onCreate()
  - Restore saved state
  - Set content view
  - Initialize UI elements
  - Link UI elements to code actions
  - Set other Activity parameters as desired



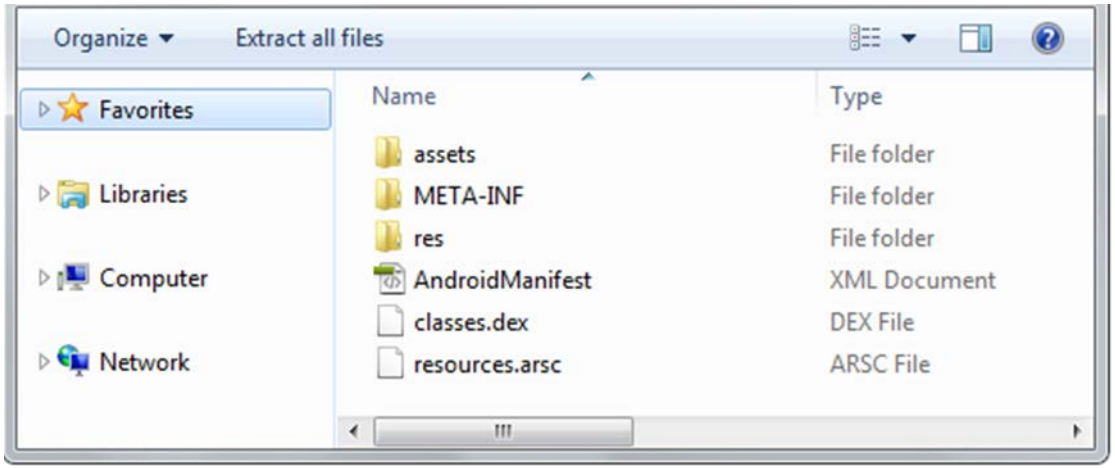

```
public class MapLocation extends
    Activity {
 public void onCreate(Bundle
    savedInstanceState) {

    super.onCreate(savedInstanceSt
    ate);        // restore saved state
     setContentView(R.layout.main);
    // set content view

    // initialize UI elements
     final EditText addressText =
      (EditText)
         findViewById(R.id.location);

     final Button button = (Button)

    findViewById(R.id.mapButton);
```

MapLocation.java

```
public class MapLocation extends Activity {
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);   // restore saved state
      setContentView(R.layout.main);              // set content view

     // initialize UI elements
      final EditText addressText = (EditText) findViewById(R.id.location);
      final Button button = (Button) findViewById(R.id.mapButton);

     // link UI elements to code actions
      button.setOnClickListener(new Button.OnClickListener() {
      public void onClick(View v) {
         String address = addressText.getText().toString();
           startActivity(new Intent(android.content.Intent.ACTION_VIEW,
                                          Uri.parse("geo:0,0?q=" + address)));
      }
}
```

3. Package Application

- Android development environment packages application as a .apk file
- Developers specify application information in AndroidManifest.xml



- **assets** folder contains documents in HTML format that inform about application, license information, FAQ etc.
- **META-INF** folder contains data that are used to ensure the integrity of the APK package and system security. There are several files in the META-INF folder namely: CERT.RSA, CERT.DSA, CERT.SF and MANIFEST.MF.
- **res** folder contains resource files, such as graphics, sounds, settings etc..
- **AndroidManifast.xml** file contains information about name, version, access rights, also references to library files and other.
- **classes.dex** is Dalvik virtual machine executable file. This file contains compiled Java source codes. DEX file can be executed only in Dalvik virtual machine and Java Runtime Environment cannot run DEX files.
- **resource.arsc** is binary resource file after compilation.

AndroidManifest.xml

- Information includes:
  - Application Name
  - Components
  - Required permissions
  - Application features
  - Minimum API level
  - Other

```
?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/
    android" package="”course.examples.SimpleActivity">
  <application>
    <activity android:name=".MapLocation"
    android:label="Map A Location">
      <intent-filter>
        <action
    android:name="android.intent.action.MAIN" />
        <category
    android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

- See: http://developer.android.com/guide/topics/fundamentals.html#Manifest

4. Install & Run

- From Eclipse run in the emulator or device
- From command line
  - Enable USB Debugging on the device
    - Settings > Applications > Development > USB debugging
  - % adb install <path_to_apk>

More on Intents

- Components can communicate by sending & receiving Intent events
- From AndroidManifest.xml (intent filter)
  - <action android:name="android.intent.action.MAIN" />
  - <category android:name="android.intent.category.LAUNCHER" />
- Specifies that MapLocation Activity is entry point for the application & will appear in launcher
- System sends this Intent to application when user clicks on application icon

Component Lifecycles

- Android can pause or terminate individual components
- For example when:
  - Task stack changes
  - Memory gets low
  - User stops interacting with the application
  - New application is launched
- At these times, Android notifies applications by calling their lifecycle methods
  - Each component type has its own lifecycle
  - Will discuss more in later classes