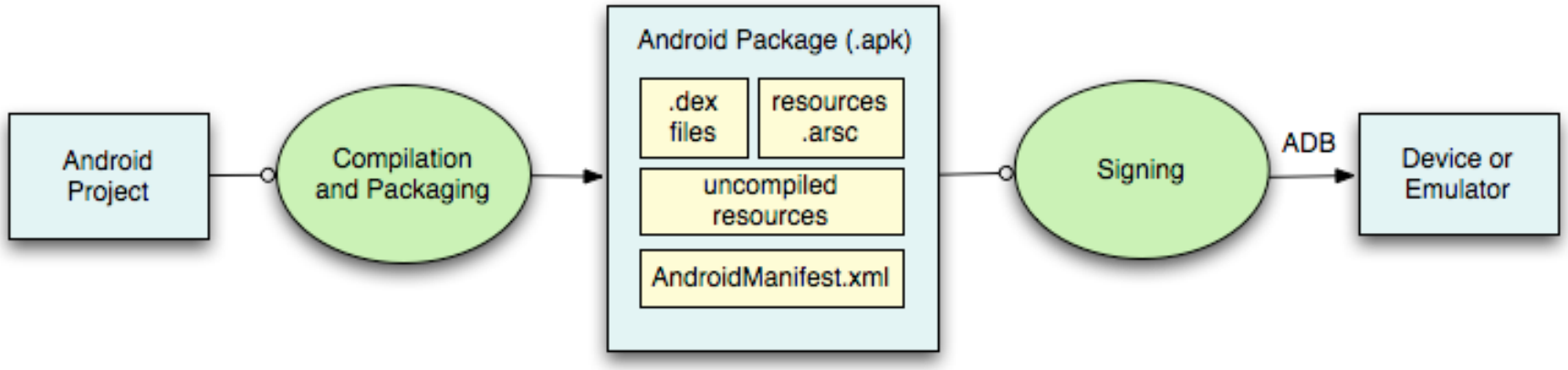
Application Fundamentals

Programming the Android Platform

Building an Application



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

Running an Application

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

Application Components

- An App can have multiple entry points
  - i.e., not just main() method
- App comprises components that the system can instantiate and run as needed
- Key component classes include:
  - Activities
  - Services
  - Broadcast receivers
  - Content providers

Activity

- Primary class for interacting with user
  - Usually implements a focused task
  - Usually Involves one screenful of data
- Example:
  - Calculator

Service

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

BroadcastReceiver

- Component that listens for broadcast announcements (events)
  - Events implemented as Intent instances
- Does not have a visual user interface
- Example
  - Messaging (on SMS receipt)

Content Providers

- Store & retrieve data across applications
- Uses database-style interface
- Example
  - Contacts

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: developer.android.com/guide/topics/resources/index.html

Layout

- User interface layout specified in XML file
  - With Eclipse can also do layout visually
- Stored in res/layout/filename.xml
- Accessed from R.layout class

Layout Example

```
?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”   />
</LinearLayout>
```

Strings

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

R.java

- At compilation time, resources are used to generate the R.java class
- Applications access resources through the R class

R.java

```
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 code usually in onCreate()
  - Restore saved state
  - Set content view
  - Initialize UI elements
  - Link UI elements to code actions
  - Set other Activity parameters as desired

MapLocation.java

```
public class MapLocation extends Activity {
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);   // restore saved state
```

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
```

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);
```

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) {
            try {
               String address = addressText.getText().toString();
               address = address.replace(' ', '+');
               Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                        Uri.parse("geo:0,0?q=" + address));
               startActivity(geoIntent);
            } catch (Exception e) {}}});
   }
}
```

3. Package Application

- System packages application as a .apk file
- Developers specify application information in AndroidManifest.xml

AndroidManifest.xml

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

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

Example

```
?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>
```

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>

Source Code Examples

- MapLocation