

Rechner

②  
Android

02.02.14



# PROGRAMMING HANDHELD SYSTEMS

ADAM PORTER

# APPLICATION FUNDAMENTALS

# APPLICATION COMPONENTS

ACTIVITY

SERVICE

BROADCASTRECEIVER

CONTENTPROVIDER

# APPLICATIONS

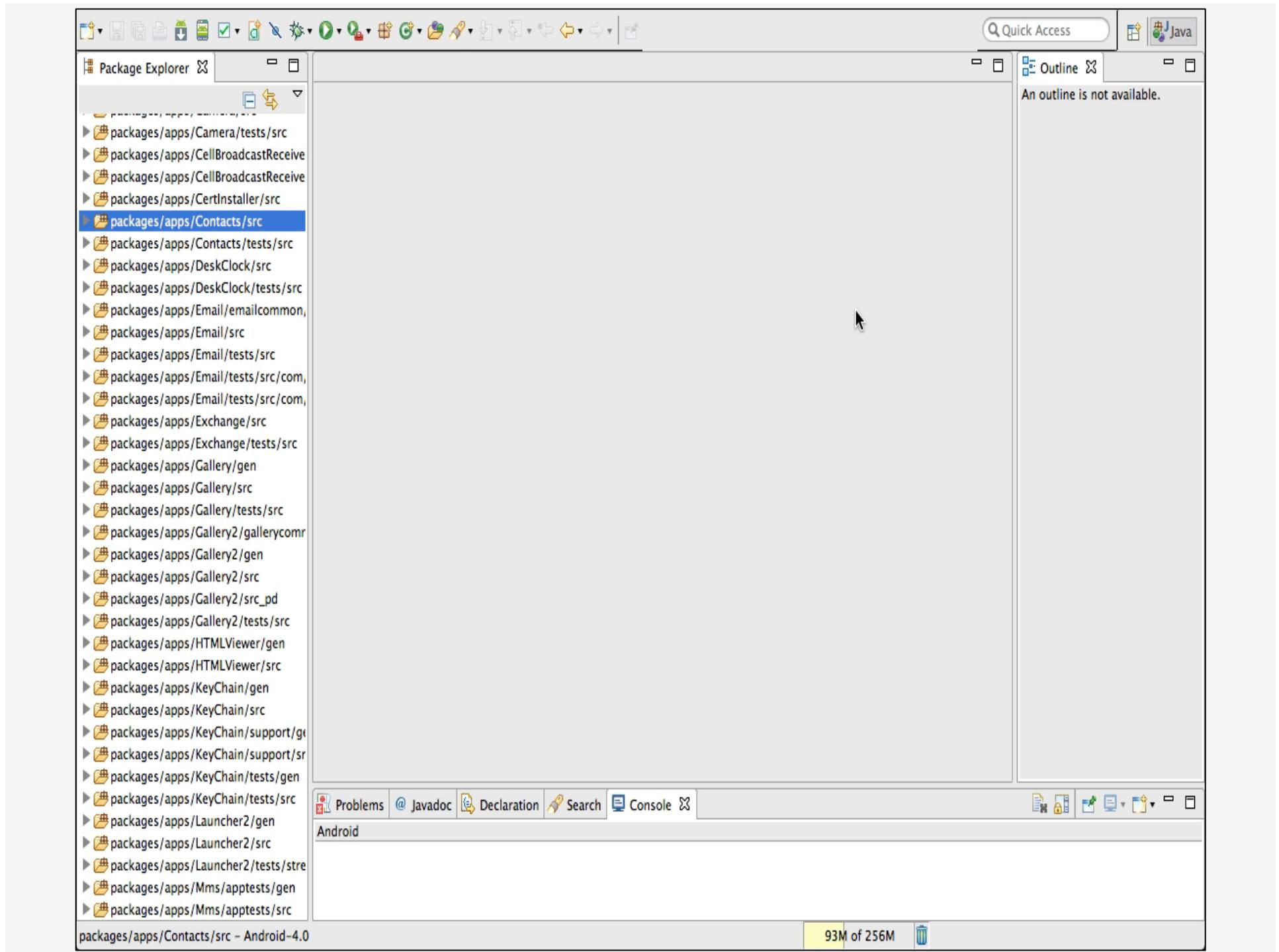
APPS ARE MADE FROM COMPONENTS

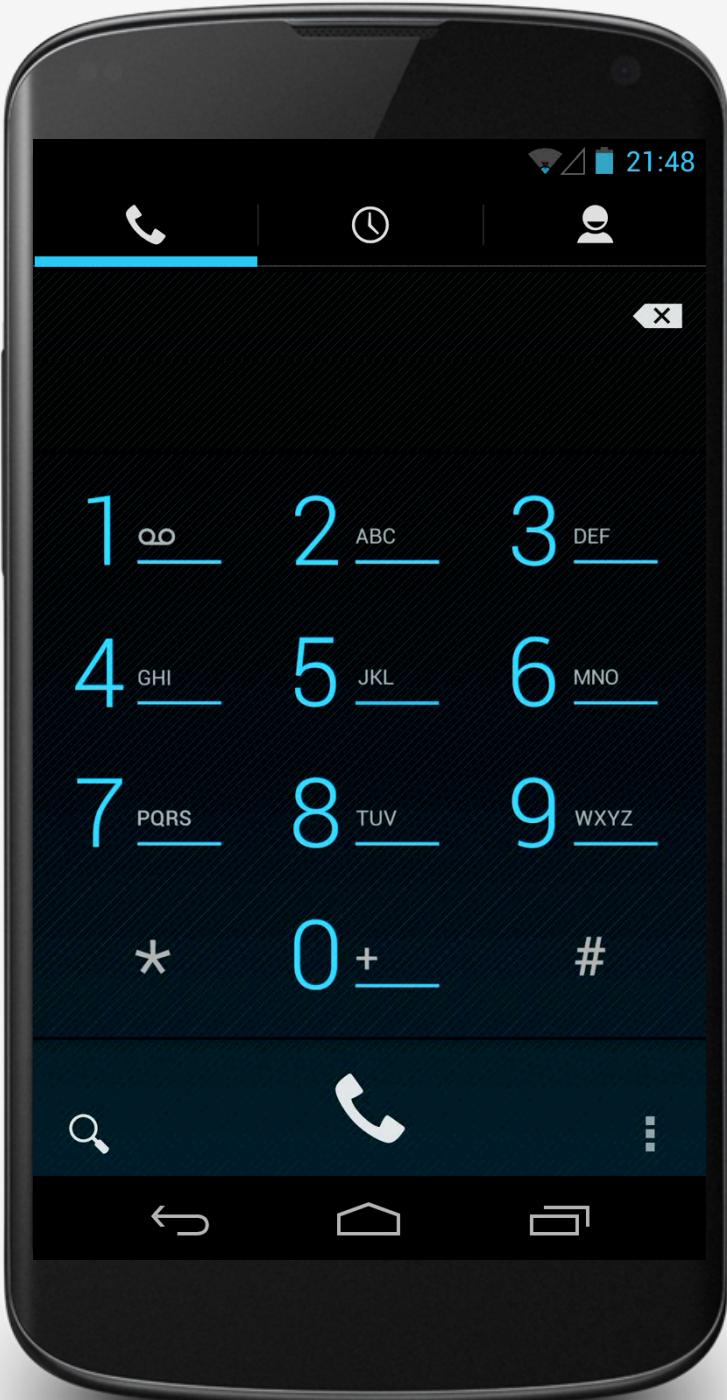
ANDROID INSTANTIATES AND RUNS THEM AS  
NEEDED

EACH COMPONENT HAS ITS OWN PURPOSE  
AND APIs

# ACTIVITY

PRIMARY CLASS FOR USER INTERACTION  
USUALLY IMPLEMENTS A SINGLE, FOCUSED  
TASK THAT THE USER CAN DO



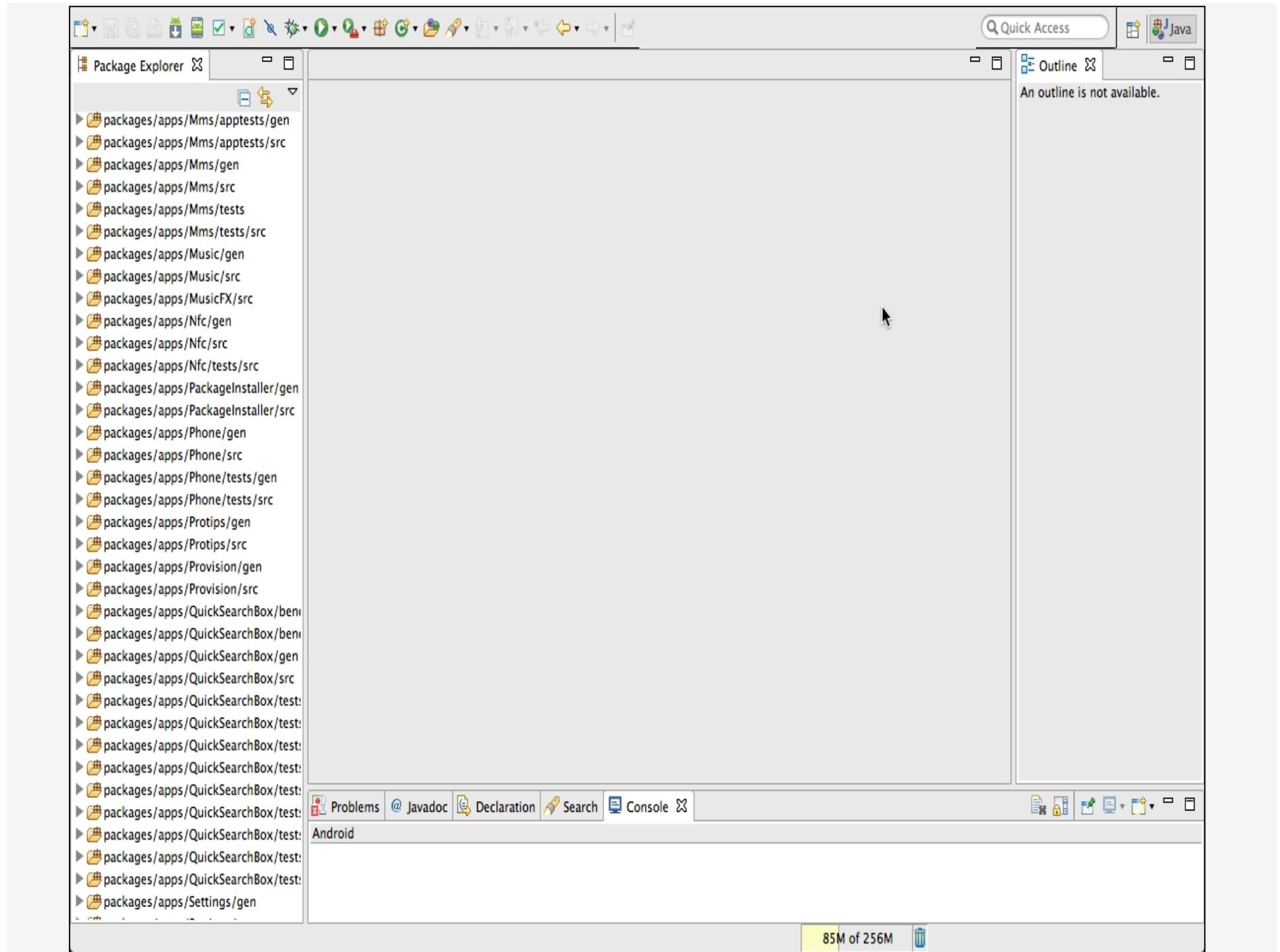


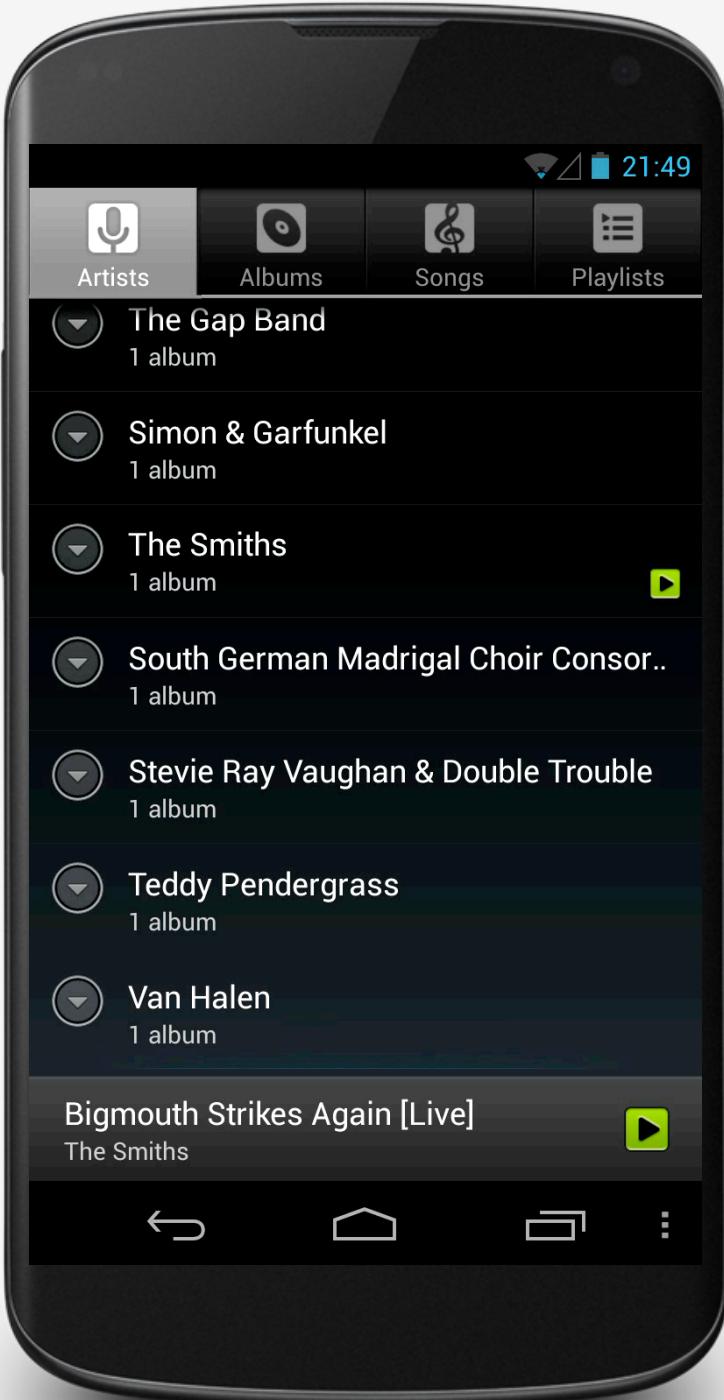
# SERVICE

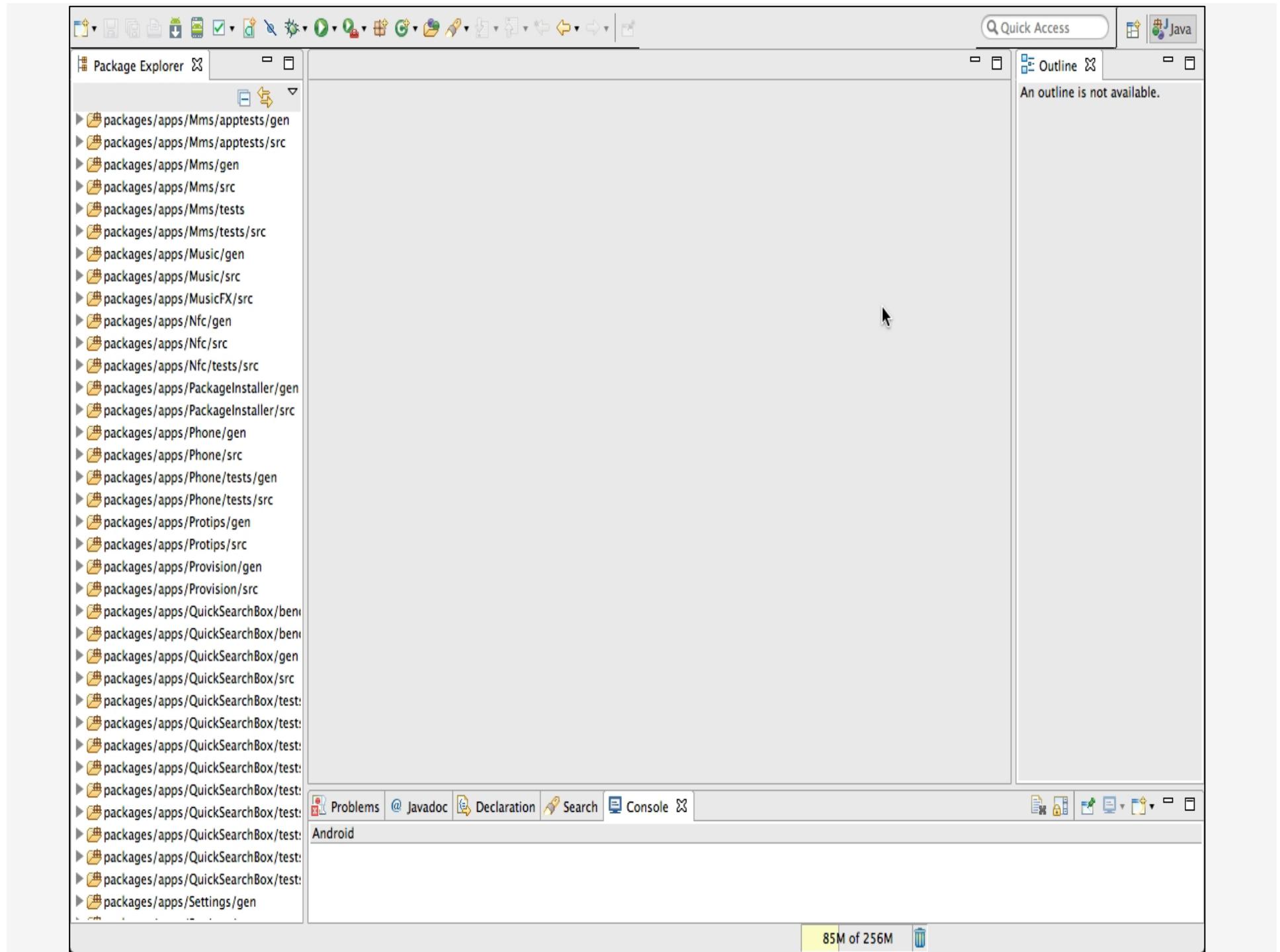
RUNS IN THE BACKGROUND

TO PERFORM LONG-RUNNING OPERATIONS

TO SUPPORT INTERACTION WITH REMOTE  
PROCESSES







# BROADCAST RECEIVER

COMPONENT THAT LISTENS FOR AND RESPONDS TO EVENTS

THE SUBSCRIBER IN PUBLISH/SUBSCRIBE PATTERN

EVENTS REPRESENTED BY THE INTENT CLASS AND THEN BROADCAST

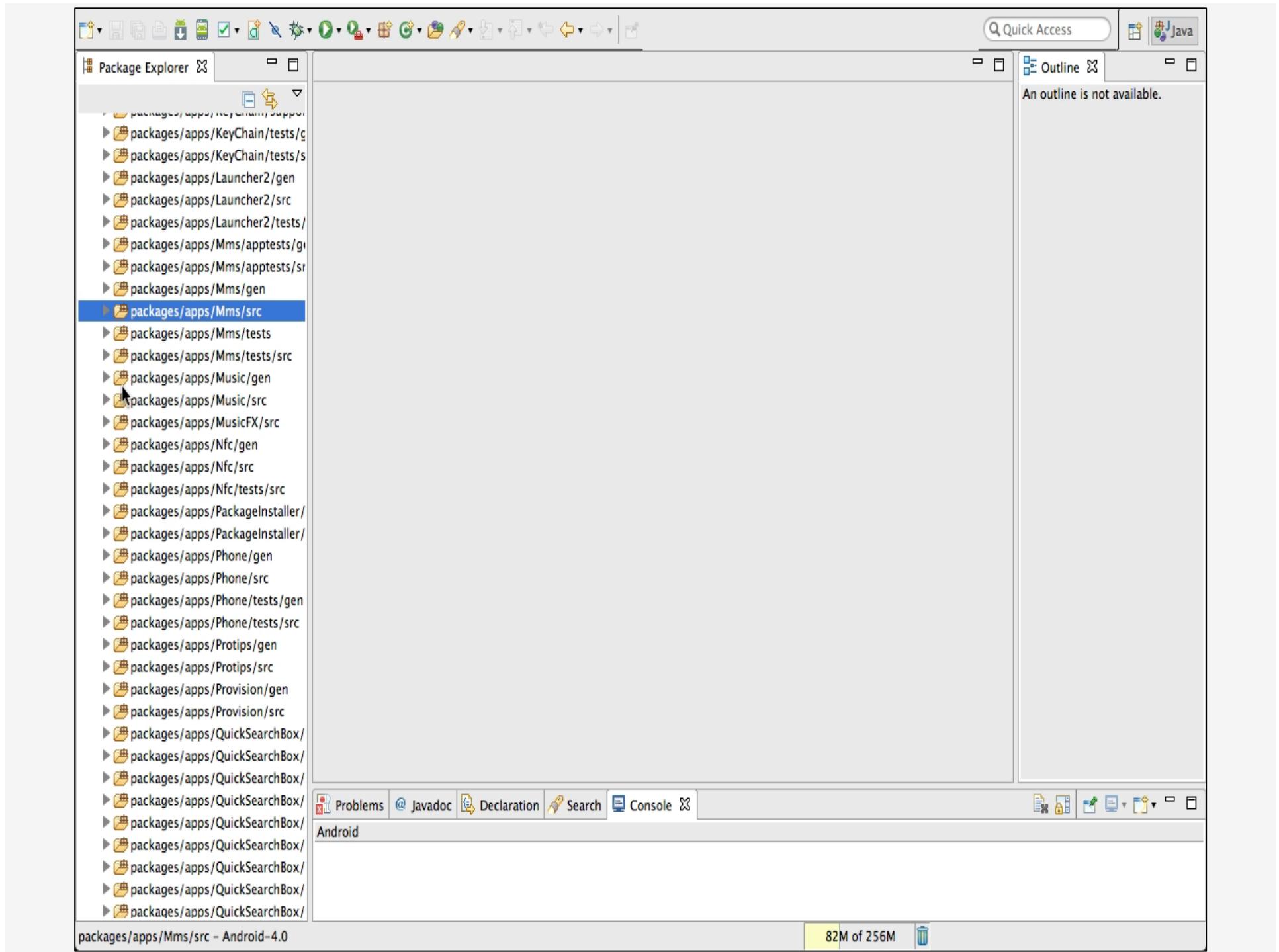
# BROADCASTRECEIVER

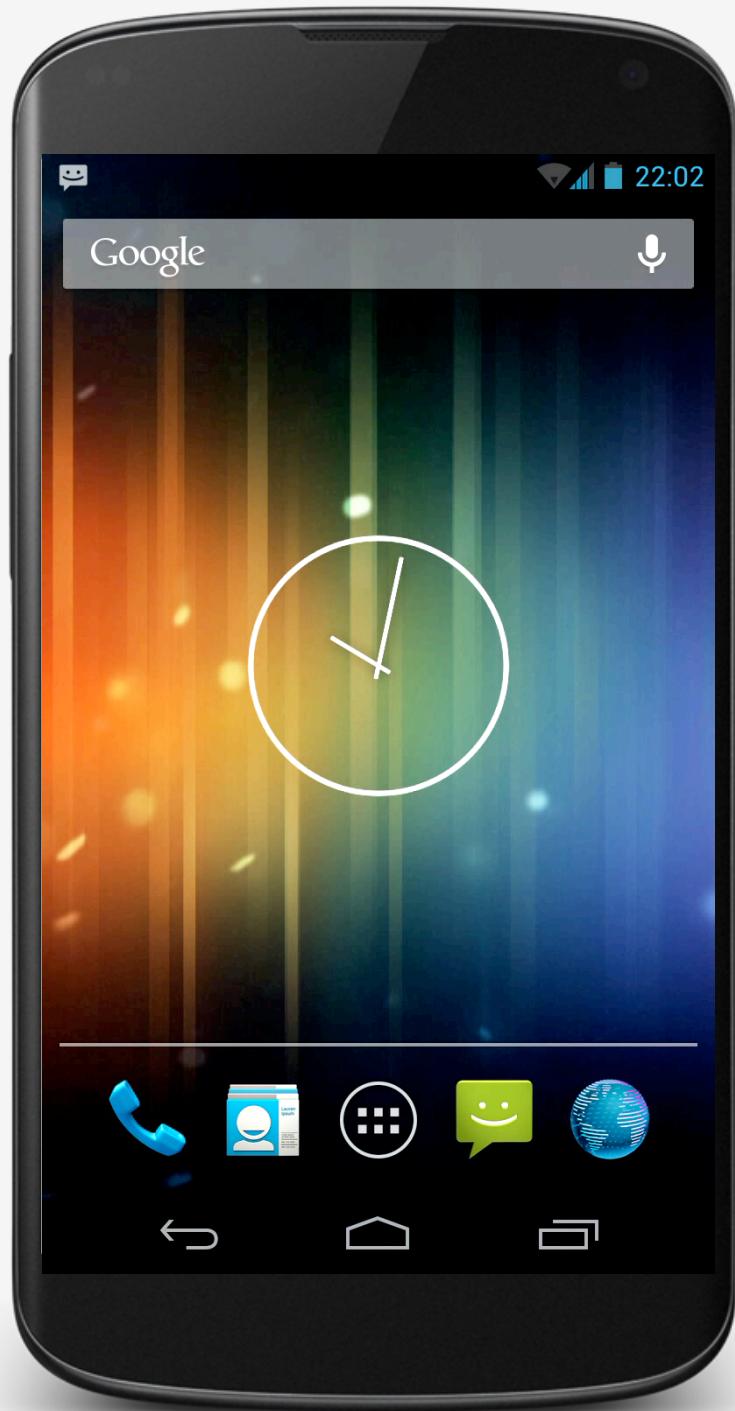
COMPONENT THAT LISTENS FOR AND RESPONDS TO EVENTS

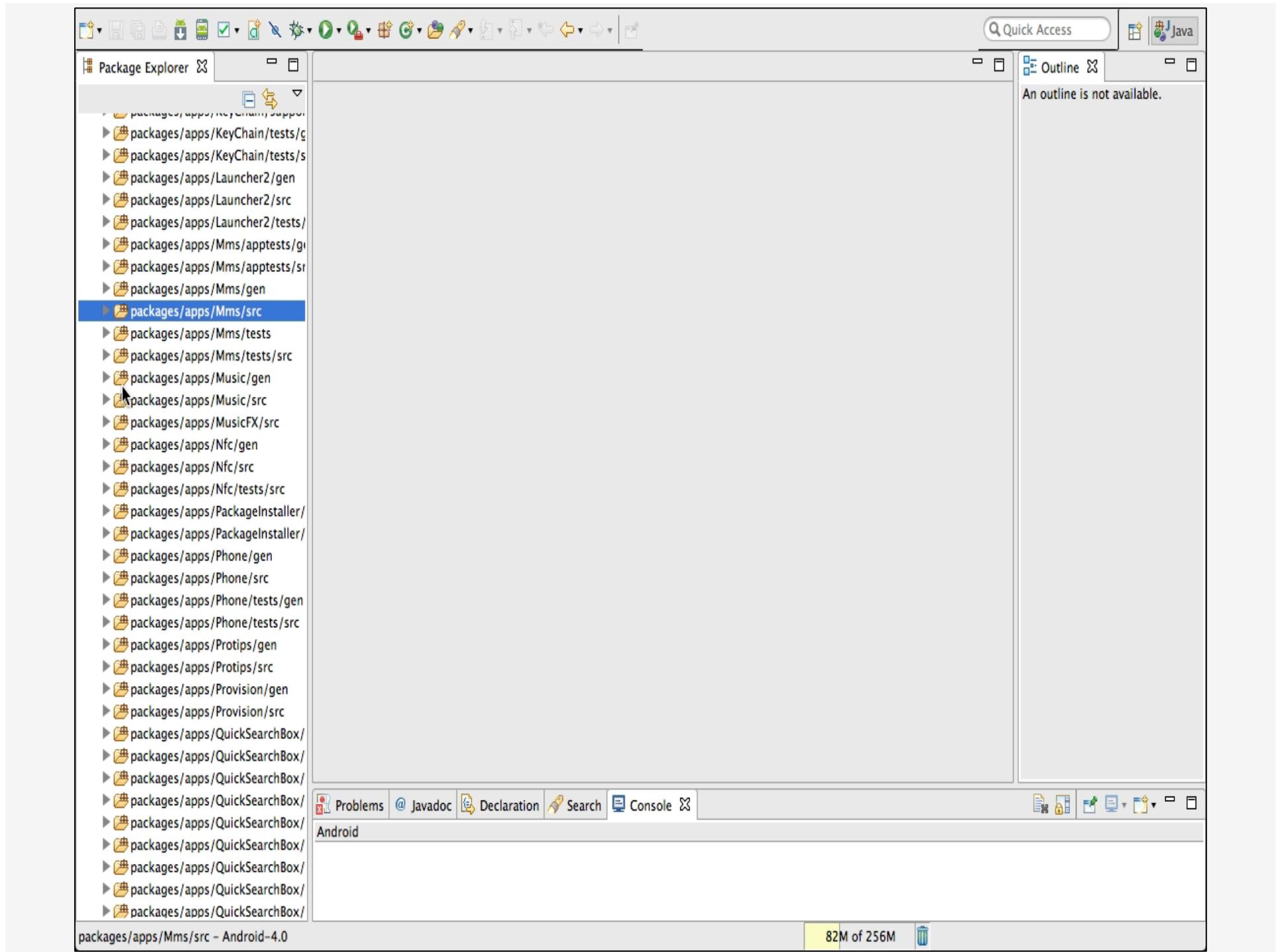
THE SUBSCRIBER IN PUBLISH/SUBSCRIBE PATTERN

EVENTS REPRESENTED BY THE INTENT CLASS AND THEN BROADCAST

BROADCASTRECEIVER RECEIVES AND RESPONDS TO BROADCAST EVENT





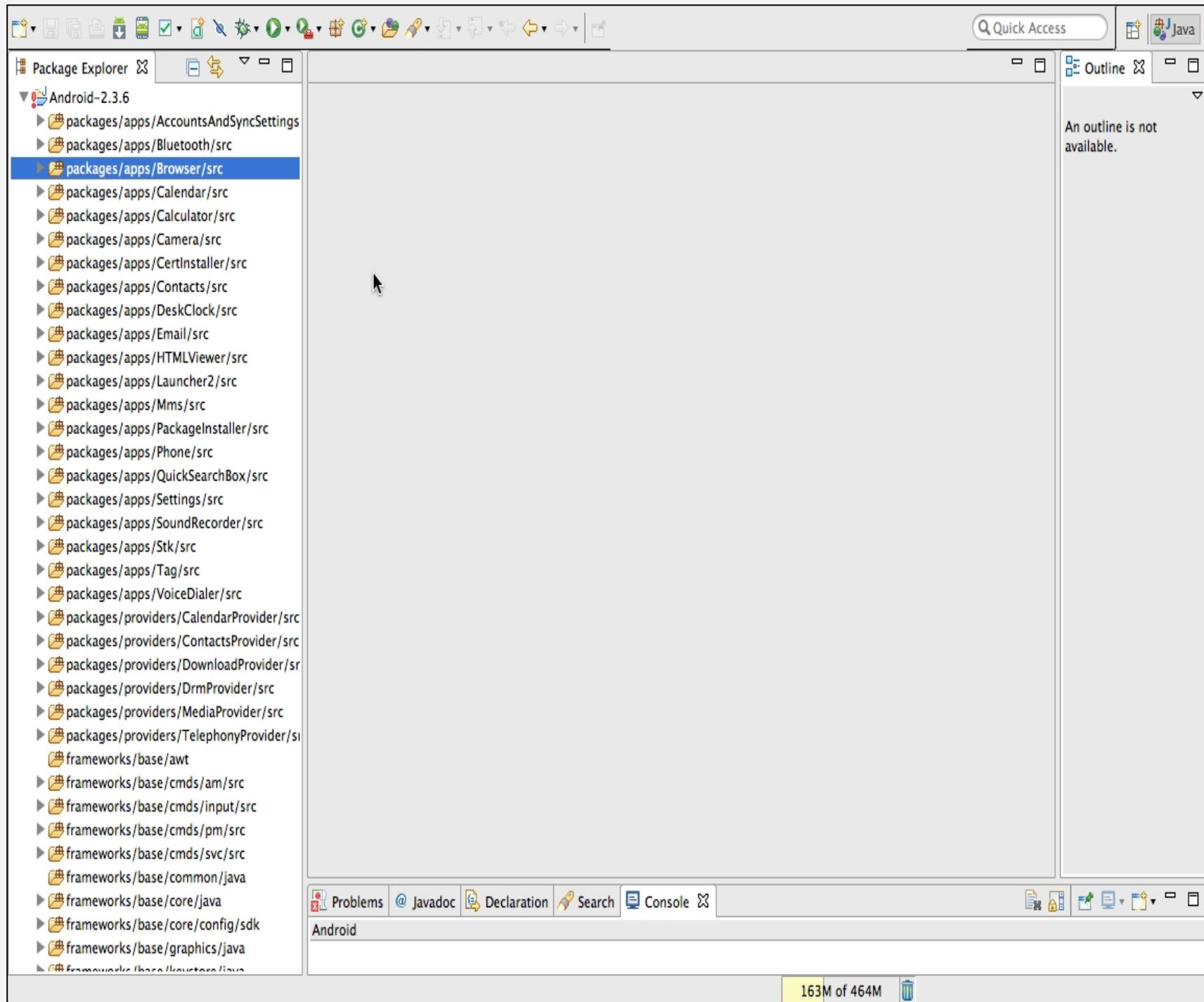


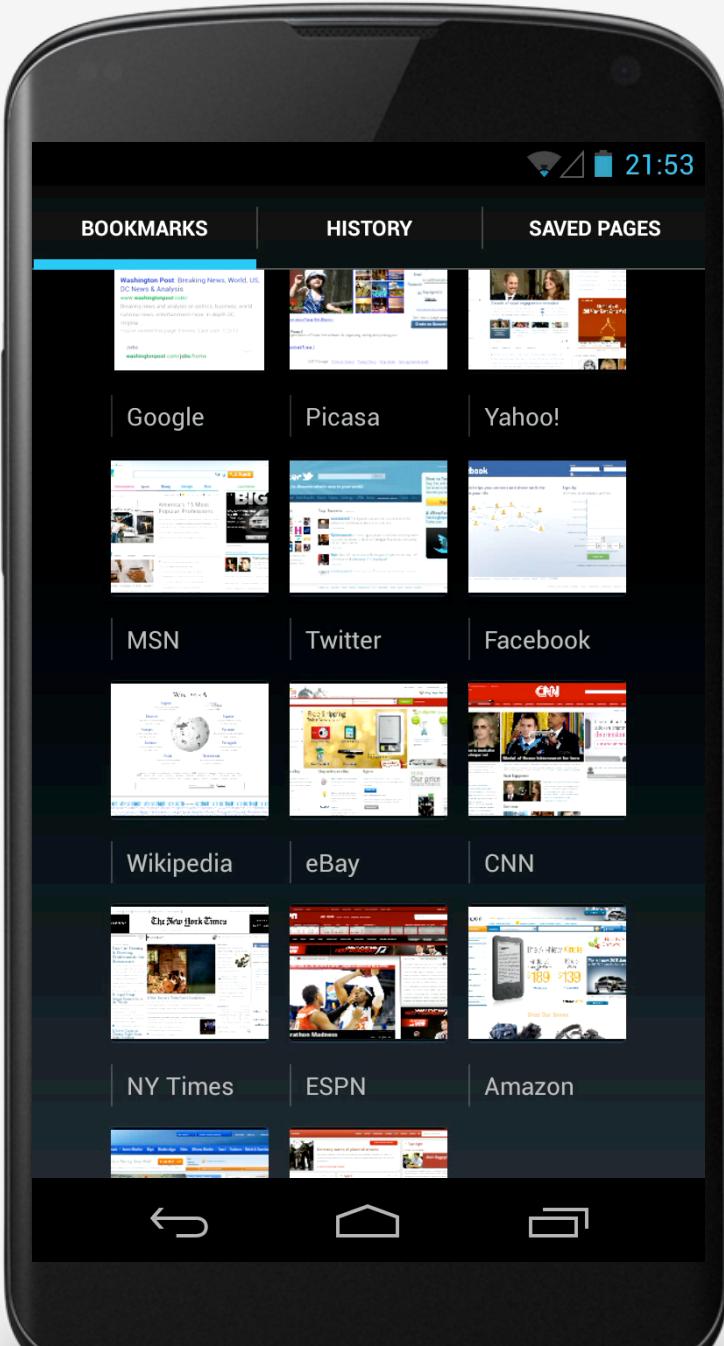
# CONTENT PROVIDERS

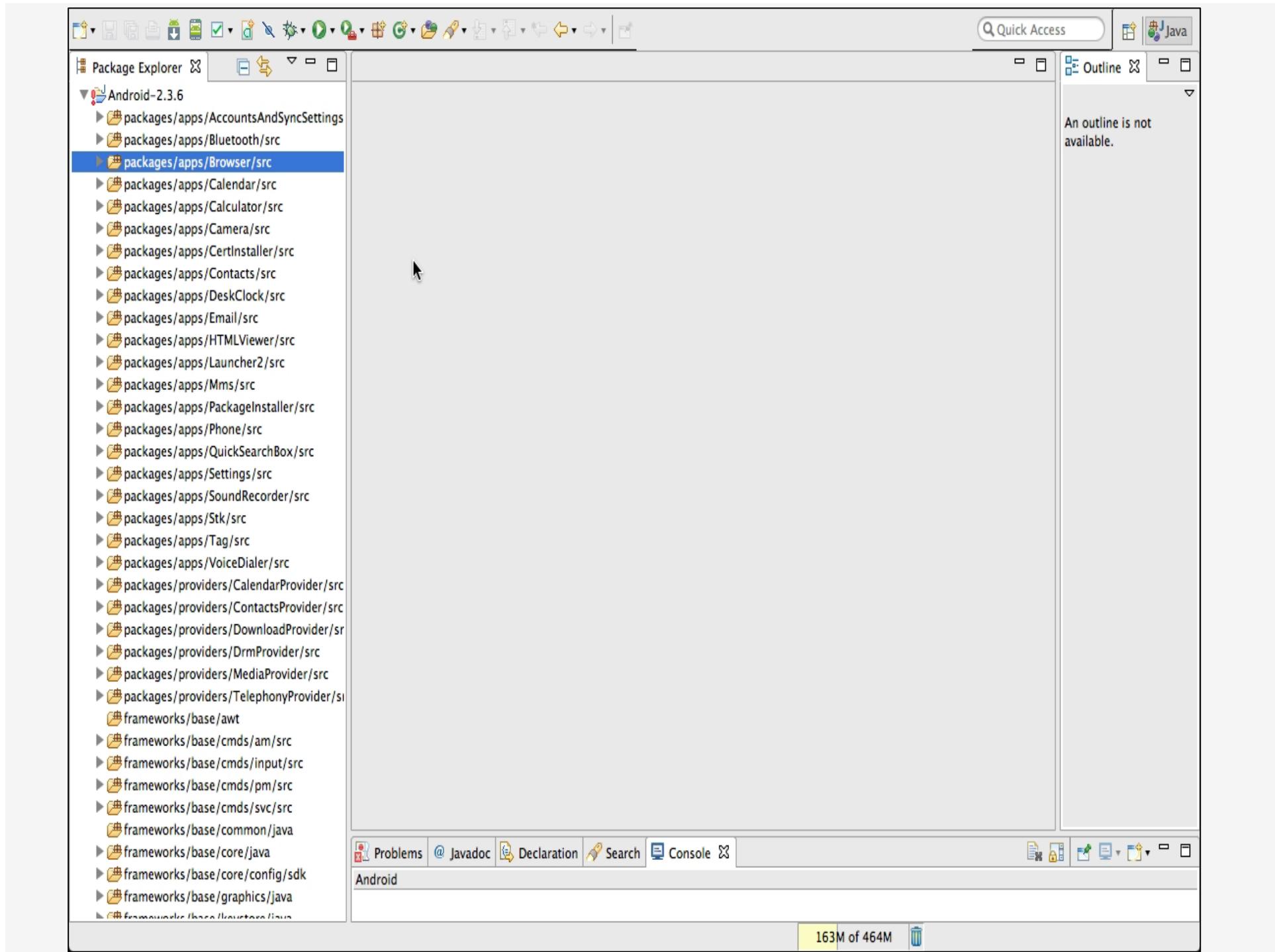
STORE & SHARE DATA ACROSS APPLICATIONS

USES DATABASE-STYLE INTERFACE

HANDLES INTERPROCESS COMMUNICATION





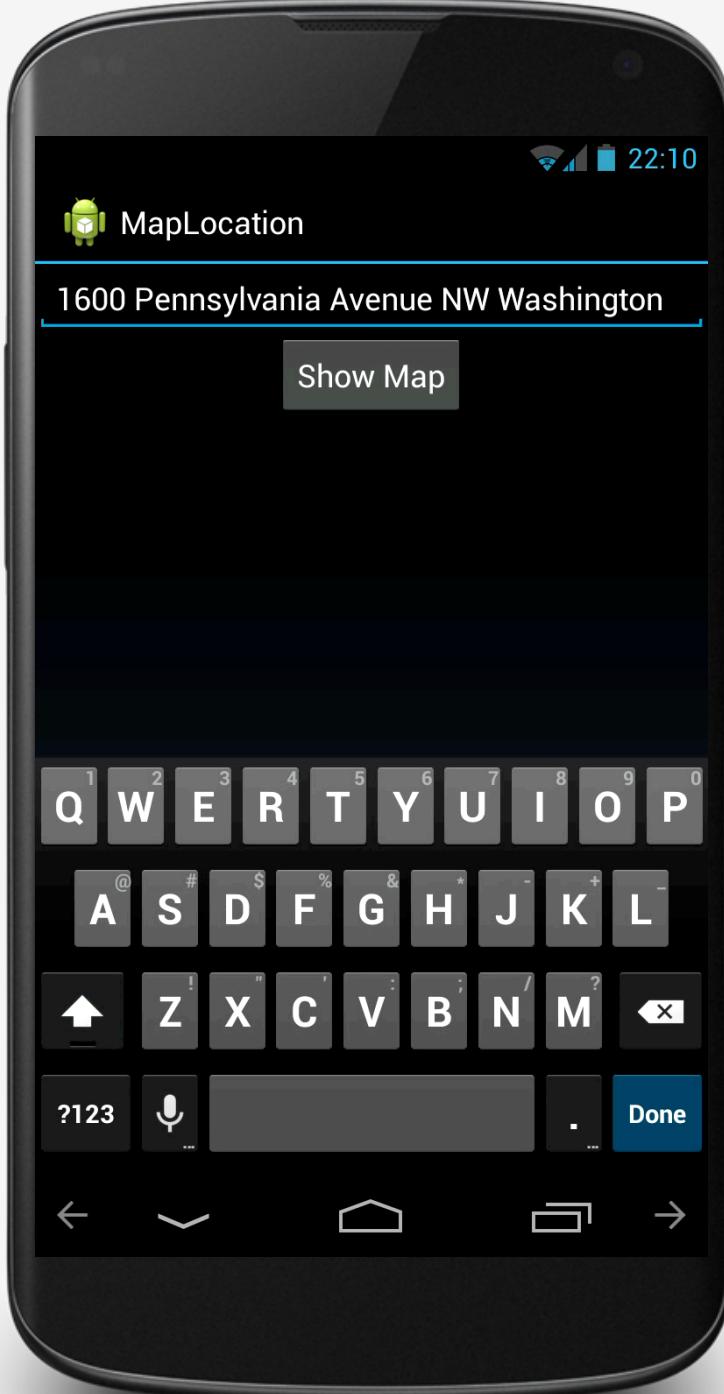




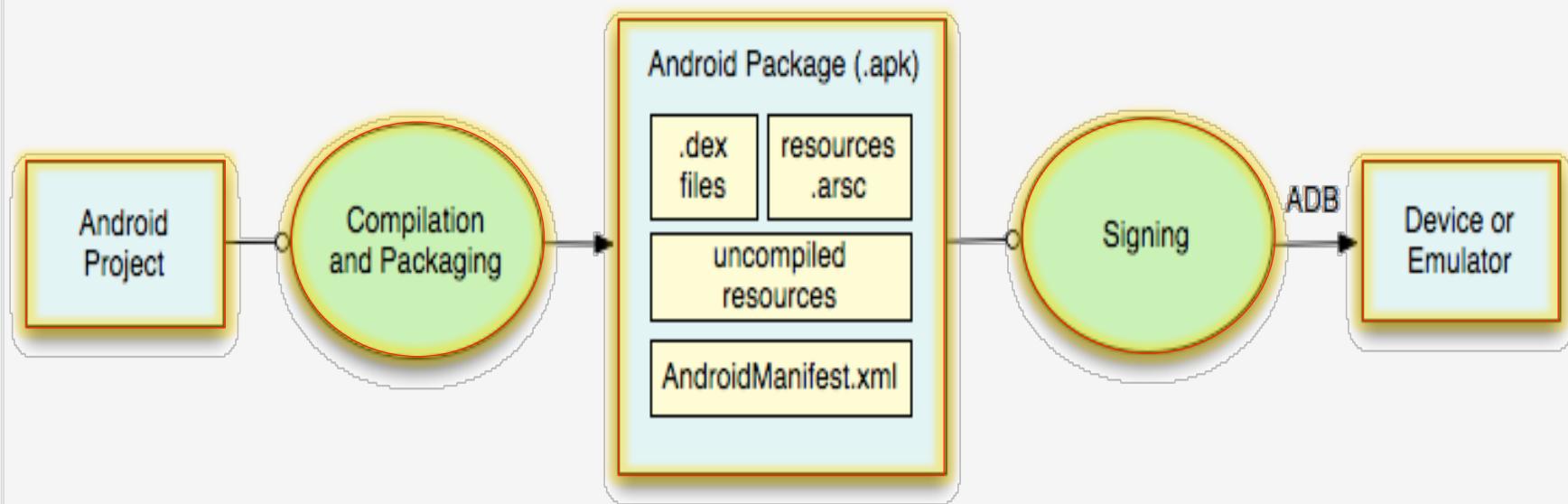
# MAPLOCATION

USER ENTERS AN ADDRESS

APP DISPLAYS A MAP OF AREA AROUND THE  
ADDRESS



# BUILDING AN APPLICATION



SEE:

<http://developer.android.com/guide/developing/building>

# CREATING AN ANDROID APP

1. DEFINE RESOURCES
2. IMPLEMENT APPLICATION CLASSES
3. PACKAGE APPLICATION
4. INSTALL & RUN APPLICATION

# 1. DEFINING RESOURCES

RESOURCES ARE NON-SOURCE CODE ENTITIES

MANY DIFFERENT RESOURCE TYPES, SUCH AS

LAYOUT, STRINGS, IMAGES, MENUS, & ANIMATIONS

ALLOWS APPS TO BE CUSTOMIZED FOR  
DIFFERENT DEVICES AND USERS

SEE:

[http://developer.android.com/guide/  
topics/resources](http://developer.android.com/guide/topics/resources)

# STRINGS

TYPES: STRING, STRING ARRAY, PLURALS

# STRINGS

TYPES: STRING, STRING ARRAY, PLURALS

TYPICALLY STORED IN RES/VALUES/\*.XML

SPECIFIED IN XML, e.g.,

```
<string name="hello">Hello World!</string>
```

CAN INCLUDE FORMATTING AND STYLING

# STRINGS

ACCESSED BY OTHER RESOURCES AS:

`@string/string_name`

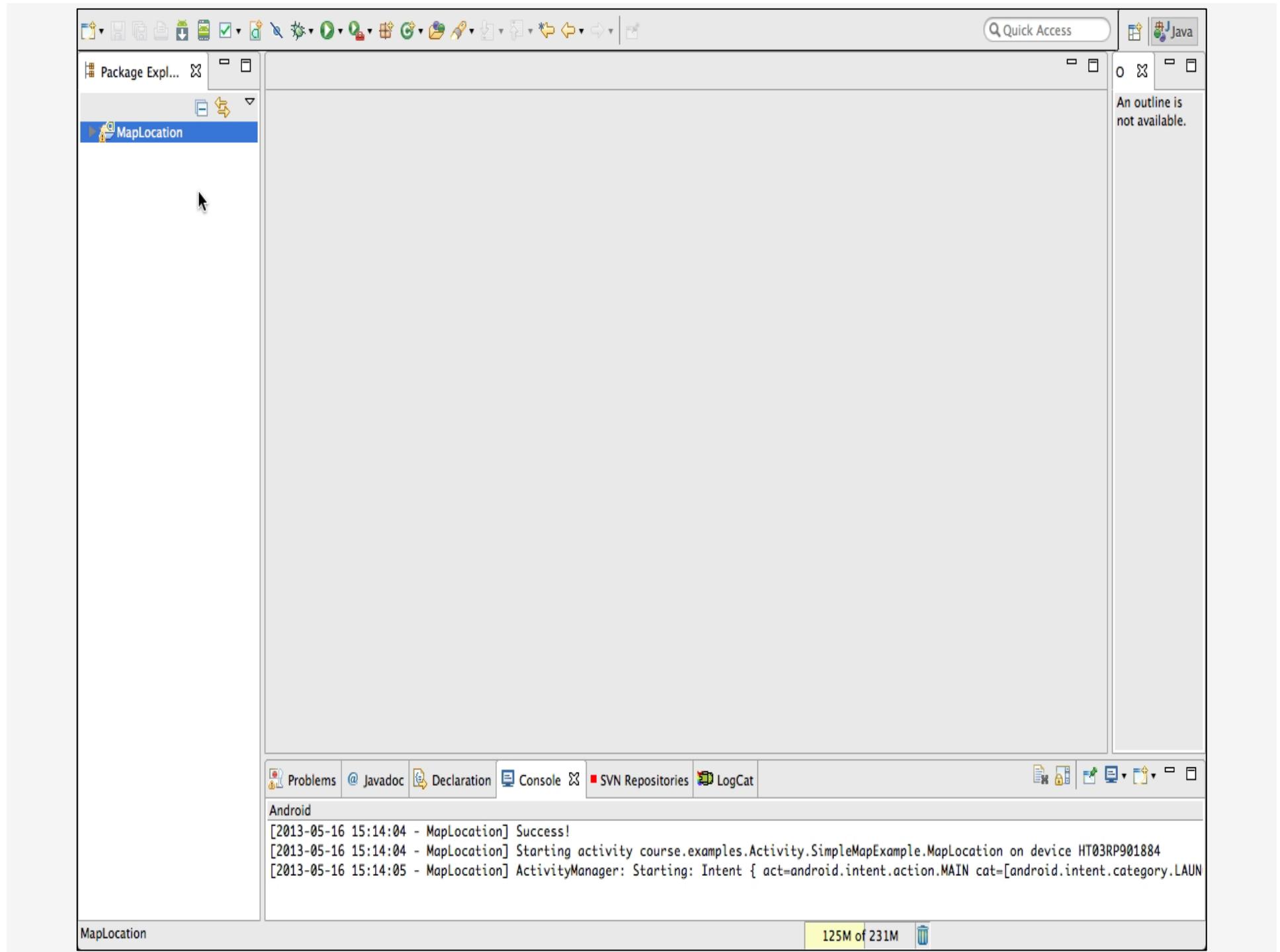
ACCESSED IN JAVA AS:

`R.string.string_name`

# MAPLOCATION

```
strings.xml ✘ strings.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="show_map_string">Show Map</string>
4     <string name="location_string">Enter Location</string>
5 </resources>
```

```
strings.xml ✘ strings.xml ✘
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="show_map_string">Mostra la mappa</string>
4     <string name="location_string">Digita l\'indirizzo</string>
5 </resources>
```



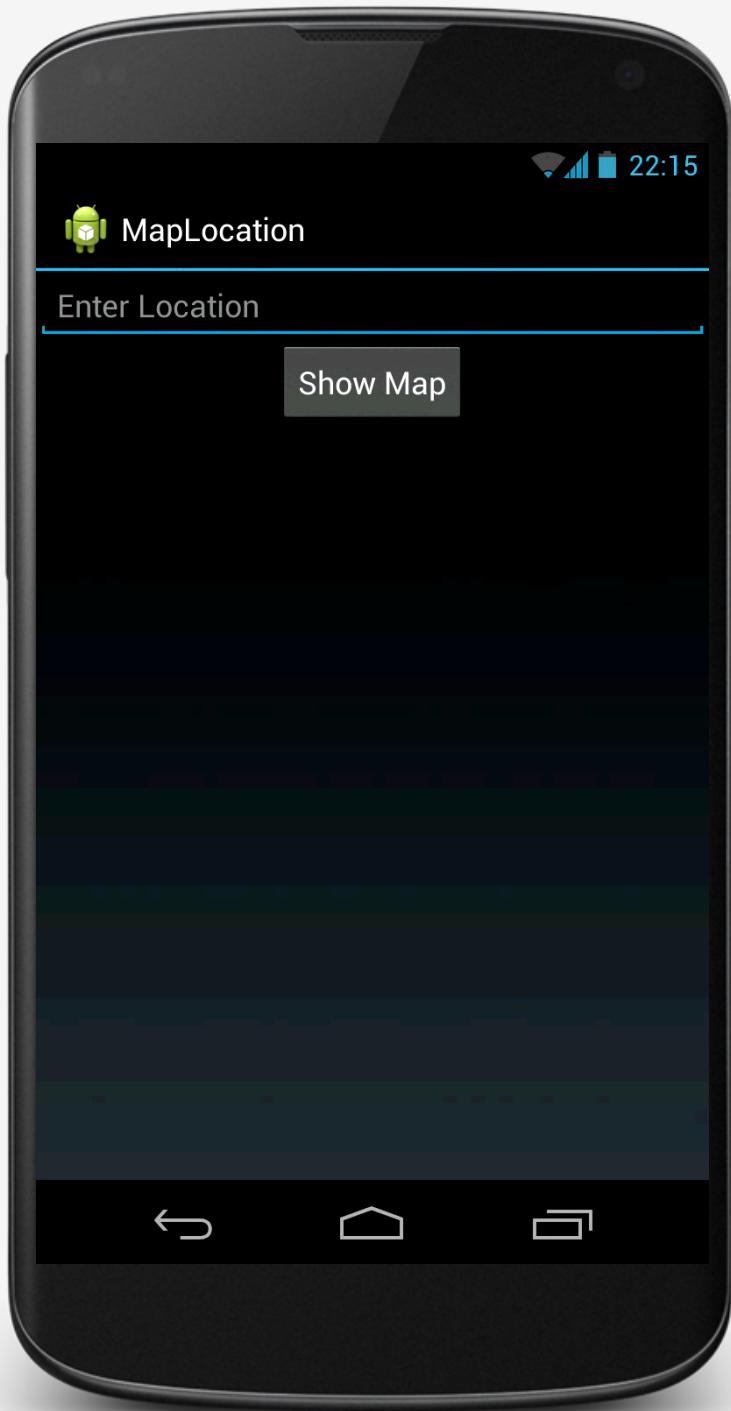
# CUSTOMIZING STRINGS

IF YOUR DEFAULT LANGUAGE IS ITALIAN,  
@string/location\_string IS

“DIGITA L’INDIRIZZO”

OTHERWISE,

“ENTER LOCATION”





# USER INTERFACE LAYOUT

UI LAYOUT SPECIFIED IN XML FILES

SOME TOOLS ALLOW VISUAL LAYOUT

XML FILES TYPICALLY STORED IN

RES/LAYOUT/\*.XML

ACCESSED IN JAVA AS:

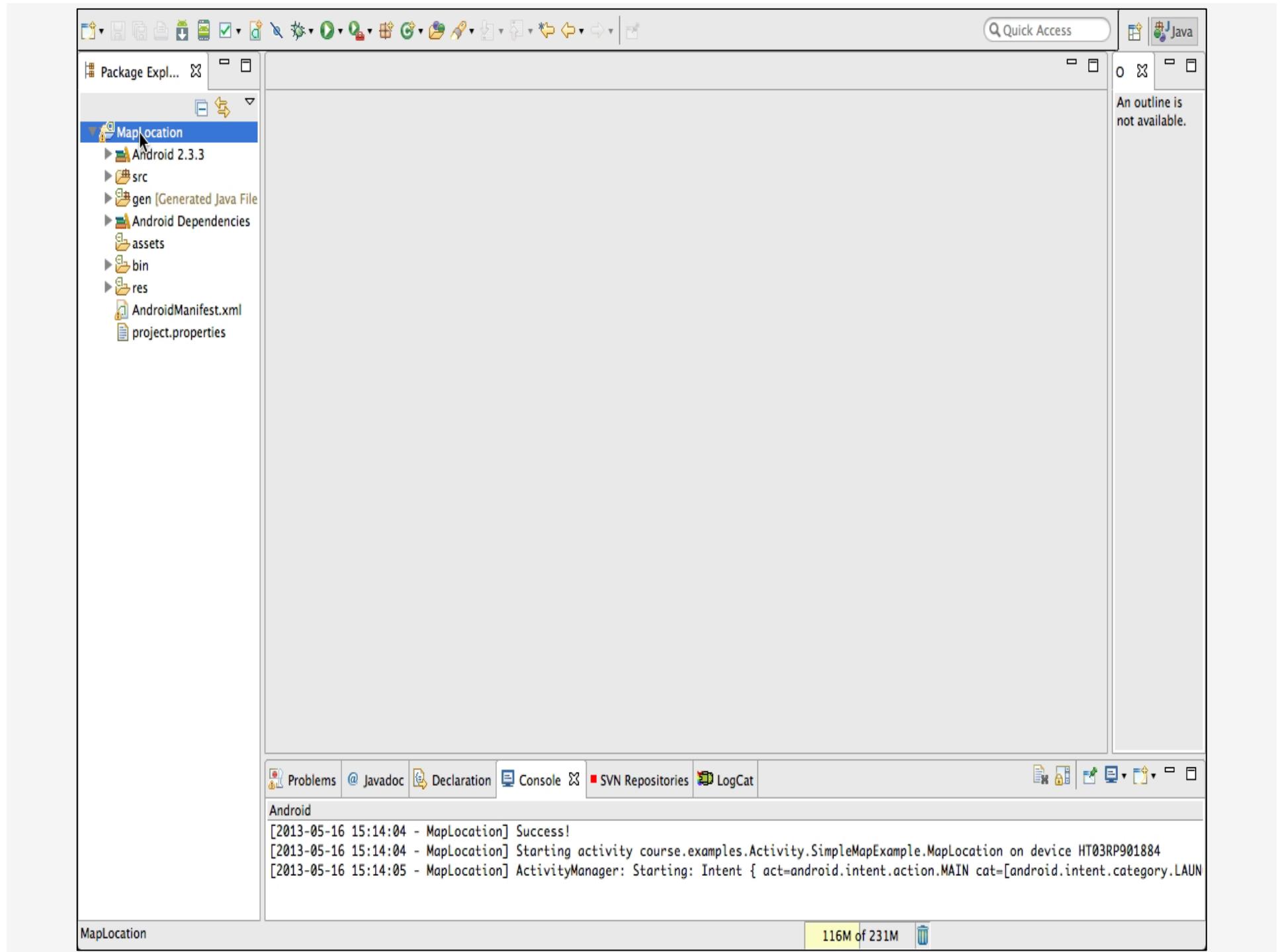
R.layout.layout\_name

ACCESSED BY OTHER RESOURCES AS:

@layout/layout\_name

# USING MULTIPLE LAYOUT FILES

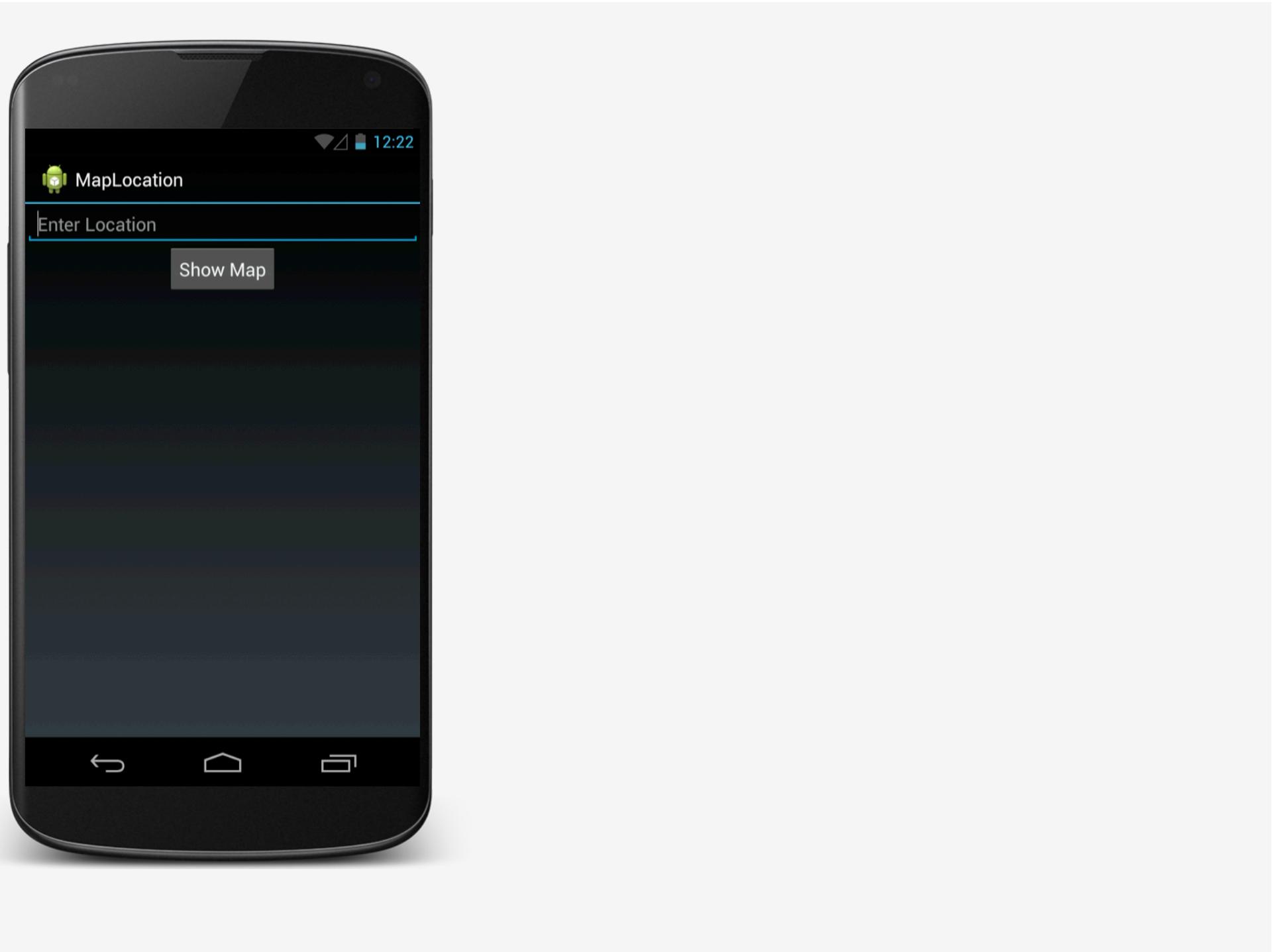
CAN SPECIFY DIFFERENT LAYOUT FILES BASED  
ON YOUR DEVICE'S ORIENTATION, SCREEN SIZE,  
ETC.

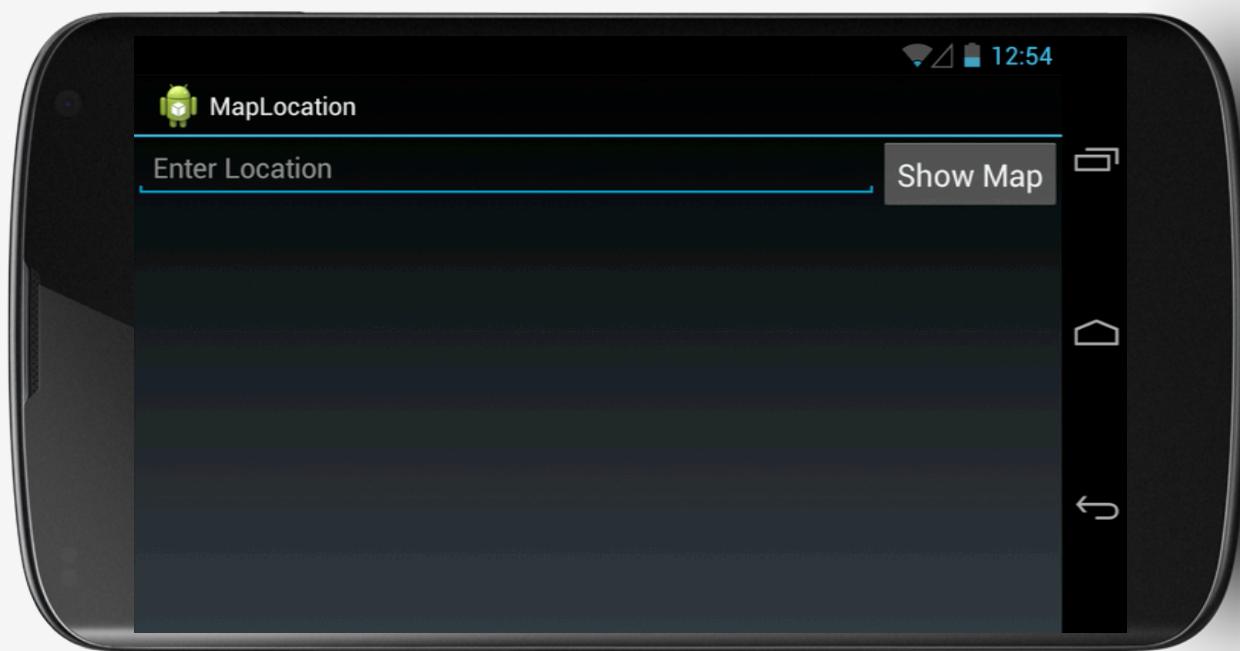


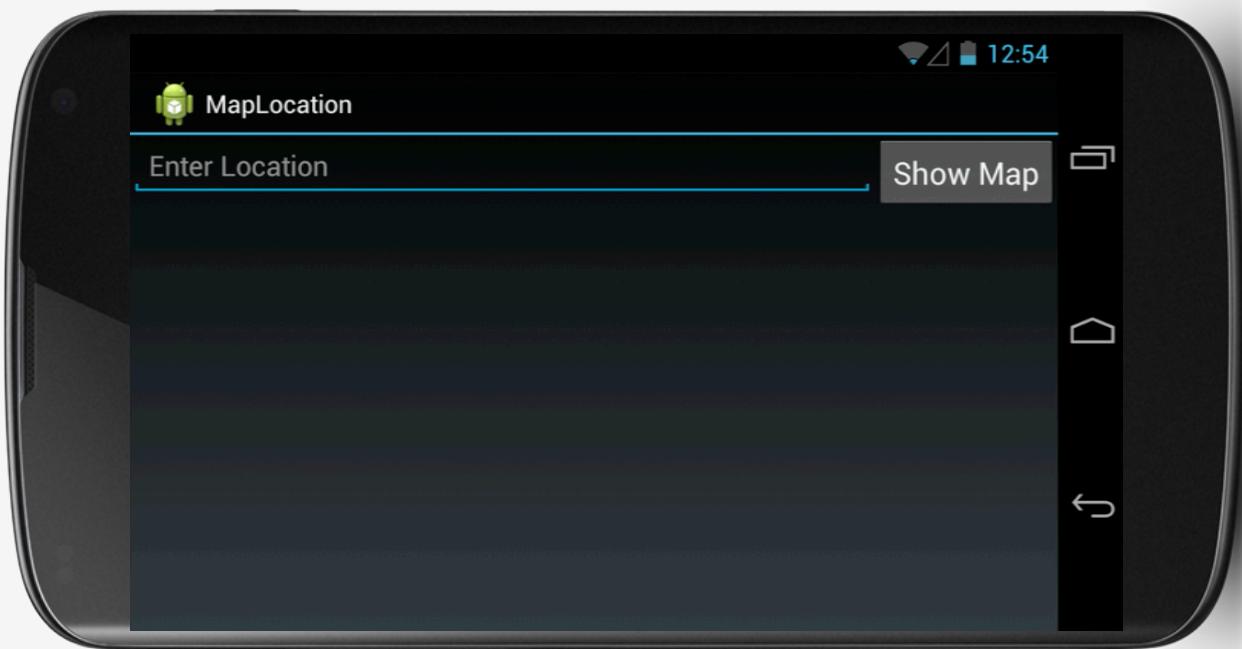
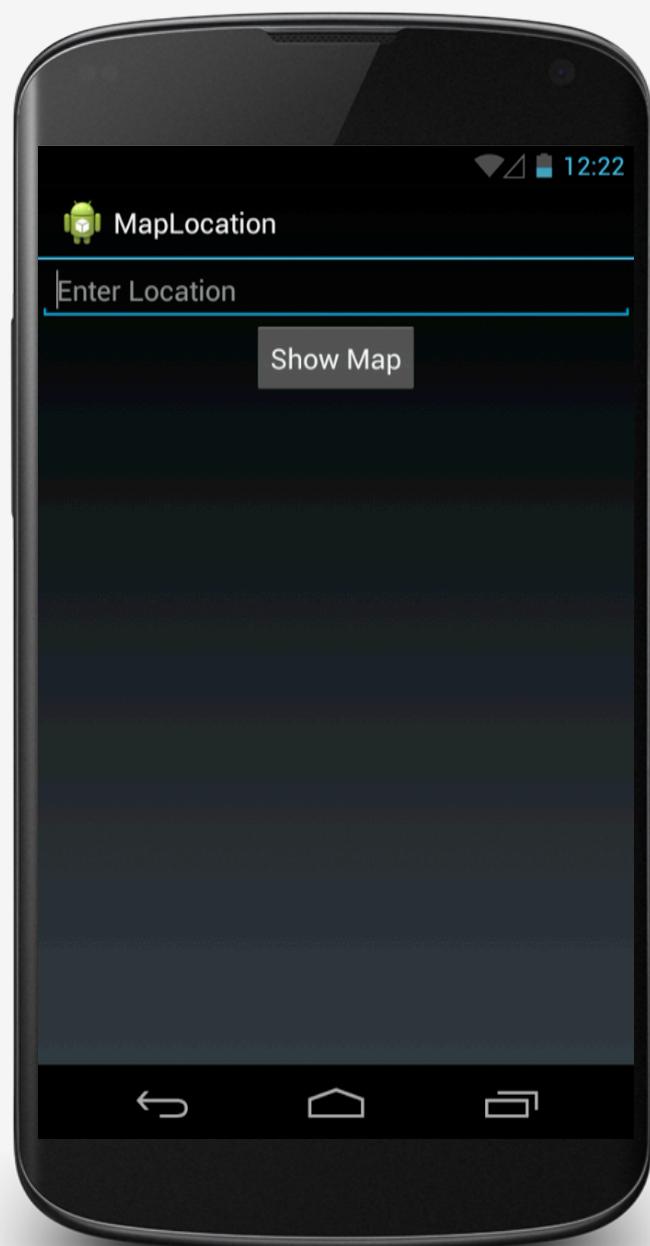
# MAPLOCATION

```
strings.xml ✘ strings.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="show_map_string">Show Map</string>
4     <string name="location_string">Enter Location</string>
5 </resources>
```

```
strings.xml ✘ strings.xml ✘
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="show_map_string">Mostra la mappa</string>
4     <string name="location_string">Digita l\'indirizzo</string>
5 </resources>
```



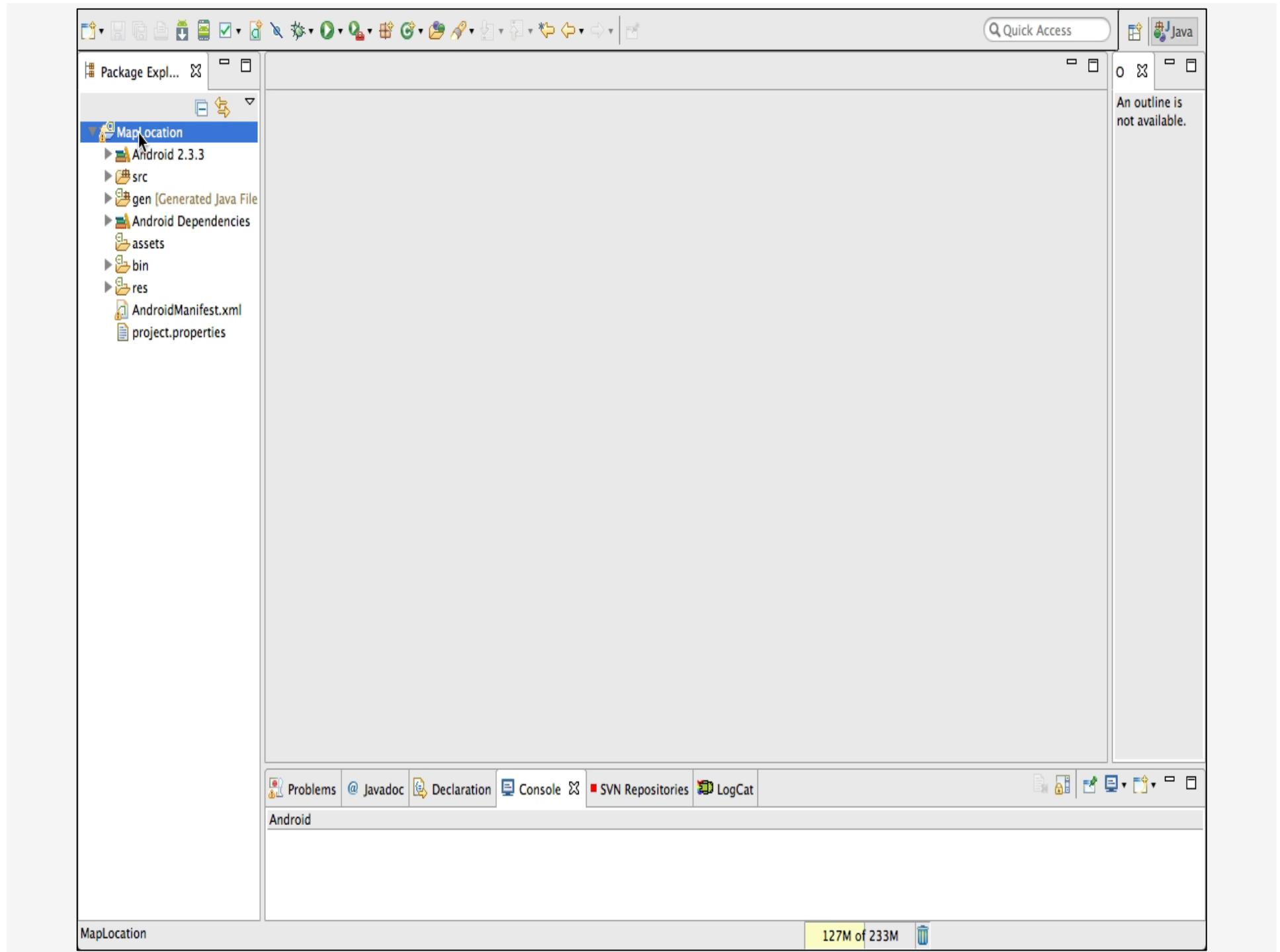




# R.JAVA

AT COMPILATION TIME, RESOURCES ARE USED  
TO GENERATE THE R.JAVA CLASS

JAVA CODE USES THE R CLASS TO ACCESS  
RESOURCES



# MAPLOCATION

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int RelativeLayout1=0x7f050000;
        public static final int location=0x7f050001;
        public static final int mapButton=0x7f050002;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int location_string=0x7f040001;
        public static final int show_map_string=0x7f040000;
    }
}
```

## 2. IMPLEMENT CLASSES

USUALLY INVOLVES AT LEAST ONE ACTIVITY  
ACTIVITY INITIALIZATION CODE USUALLY IN  
ONCREATE()

## 2. IMPLEMENT CLASSES

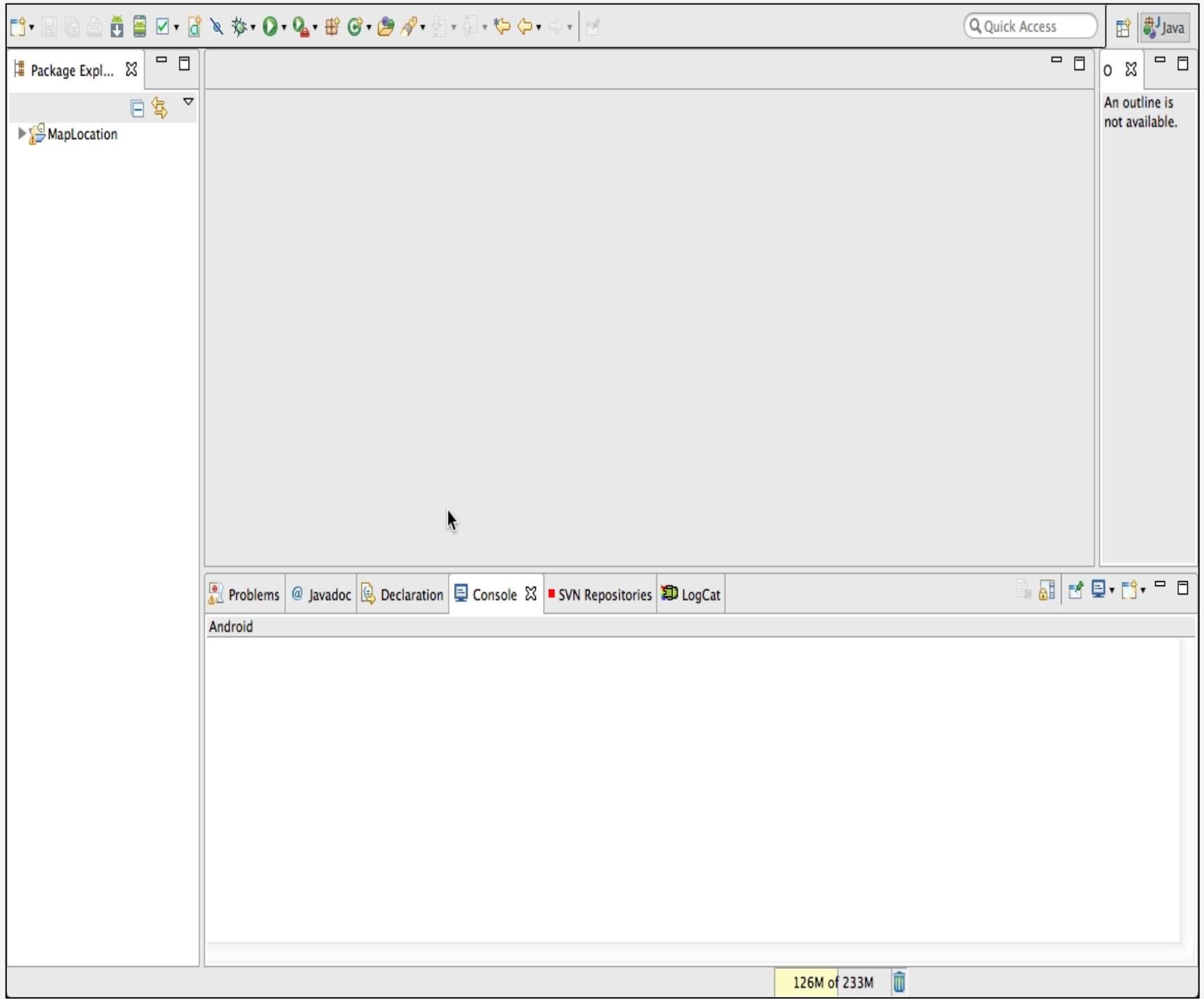
TYPICAL ONCREATE() WORKFLOW

RESTORE SAVED STATE

SET CONTENT VIEW

INITIALIZE UI ELEMENTS

LINK UI ELEMENTS TO CODE ACTIONS



# MAPLOCATION

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
  
    // Required call through to Activity.onCreate()  
    // Restore any saved instance state  
    super.onCreate(savedInstanceState);  
  
    // Set content view  
    setContentView(R.layout.main);  
  
    // Initialize UI elements  
    final EditText addrText = (EditText) findViewById(R.id.location);  
    final Button button = (Button) findViewById(R.id.mapButton);
```

# MAPLOCATION

```
// Initialize UI elements
final EditText addrText = (EditText) findViewById(R.id.location);
final Button button = (Button) findViewById(R.id.mapButton);

// Link UI elements to actions in code
button.setOnClickListener(new OnClickListener() {

    // Called when user clicks the Show Map button
    public void onClick(View v) {
        try {

            // Process text for network transmission
            String address = addrText.getText().toString();
            address = address.replace(' ', '+');

            // Create Intent object for starting Google Maps application
            Intent geoIntent = new Intent(
                android.content.Intent.ACTION_VIEW, Uri
                .parse("geo:0,0?q=" + address));

            // Use the Intent to start Google Maps application using Activity.startActivity()
            startActivity(geoIntent);

        } catch (Exception e) {
            // Log any error messages to LogCat using Log.e()
            Log.e(TAG, e.toString());
        }
    }
});
```

### 3. PACKAGE APPLICATION

SYSTEM PACKAGES APPLICATION COMPONENTS  
& RESOURCES INTO A .APK FILE

DEVELOPERS SPECIFY REQUIRED APPLICATION  
INFORMATION IN A FILE CALLED  
ANDROIDMANIFEST.XML

# ANDROIDMANIFEST.XML

INFORMATION INCLUDES:

APPLICATION NAME

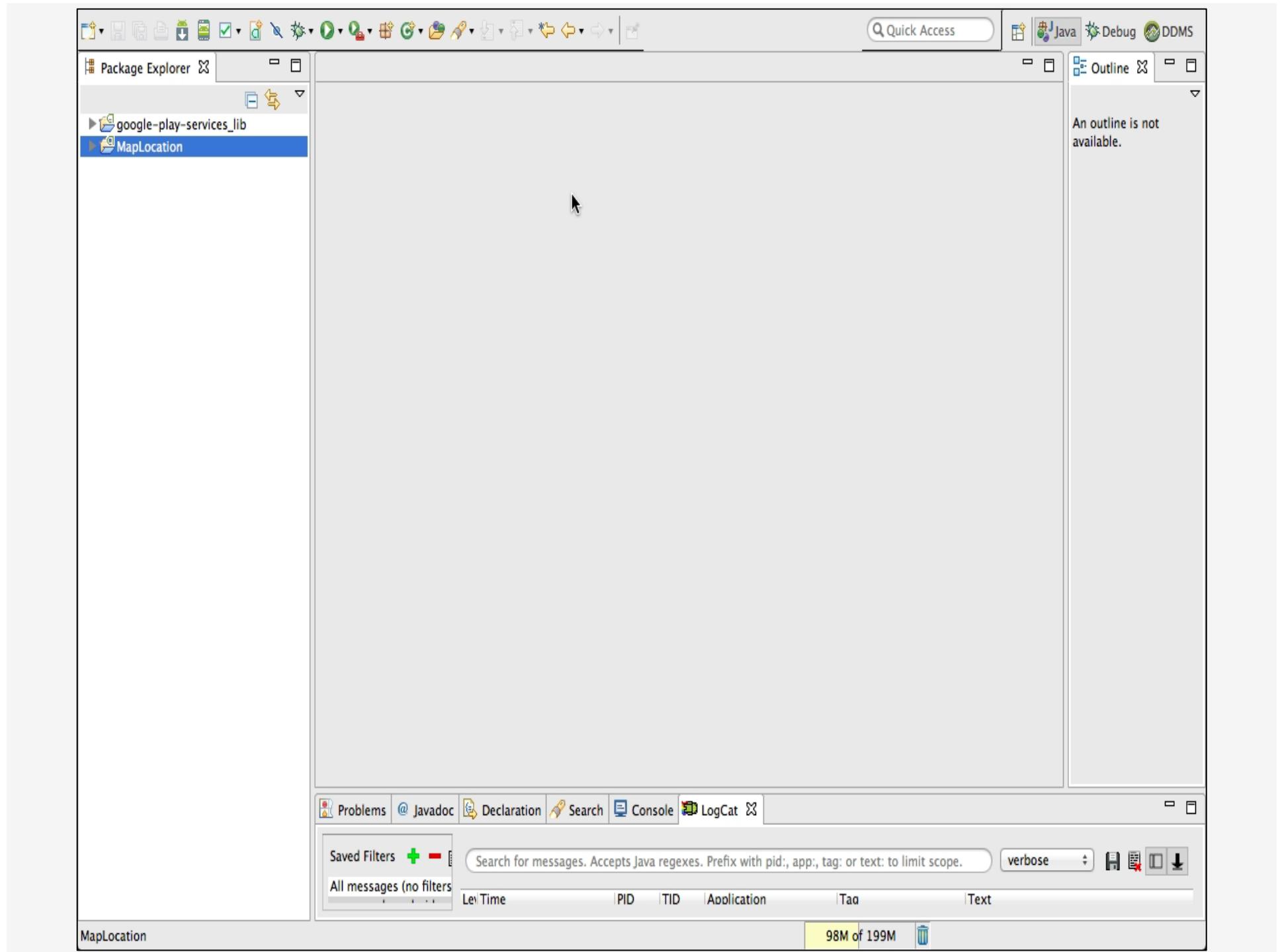
COMPONENTS

OTHER

REQUIRED PERMISSIONS

APPLICATION FEATURES

MINIMUM API LEVEL



# MAPLOCATION

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="course.examples.MapLocation"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="13"
        android:targetSdkVersion="19" />

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="MapLocation" >
        <activity android:name="course.examples.MapLocation.MapLocation" >
            <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>

NEXT TIME

THE ACTIVITY CLASS