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
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

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
Strings

Types: String, String Array, Plurals
Strings

Types: String, String Array, Plurals

Typically stored in res/values/*.xml

Specified in XML, e.g.,

```xml
<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

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="show_map_string">Show Map</string>
  <string name="location_string">Enter Location</string>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="show_map_string">Mostra la mappa</string>
  <string name="location_string">Digita l'indirizzo</string>
</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

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="show_map_string">Show Map</string>
  <string name="location_string">Enter Location</string>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="show_map_string">Mostra la mappa</string>
  <string name="location_string">Digita l'indirizzo</string>
</resources>
```
At compilation time, resources are used to generate the R.java class.
Java code uses the R class to access resources.
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 RelativeLayout=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
@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);
// 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
<?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