



# Android SDK (2)

## Lecture 3

Security of Mobile Devices

2023



## Intents

Broadcast Receivers

Content Providers

Tools

Bibliography

## Intents

Broadcast Receivers

Content Providers

Tools

Bibliography



- ▶ An object used for delivering a message
- ▶ Includes 3 components: target, action & data
  - ▶ The class name of the target component
  - ▶ The action to be executed by the target component
  - ▶ The data used in that action



- ▶ Declare the types of intents that a component can receive
- ▶ Specified in the manifest - <intent-filter>
- ▶ <action>, <data>



SMD

- ▶ Starting an activity
  - ▶ Pass Intent to `startActivity()` or `startActivityForResult()`
- ▶ Starting or binding a service
  - ▶ Pass Intent to `startService()` or `bindService()`
- ▶ Delivering a broadcast message
  - ▶ Pass Intent to `sendBroadcast()` or `sendOrderedBroadcast()` or `sendStickyBroadcast()`



SMD

- ▶ Types: implicit & explicit intents
- ▶ Explicit intents
  - ▶ Specify exactly the class name of the target app
  - ▶ Typically used to start components in your own app
  - ▶ Will be delivered even if there is no intent filter declared



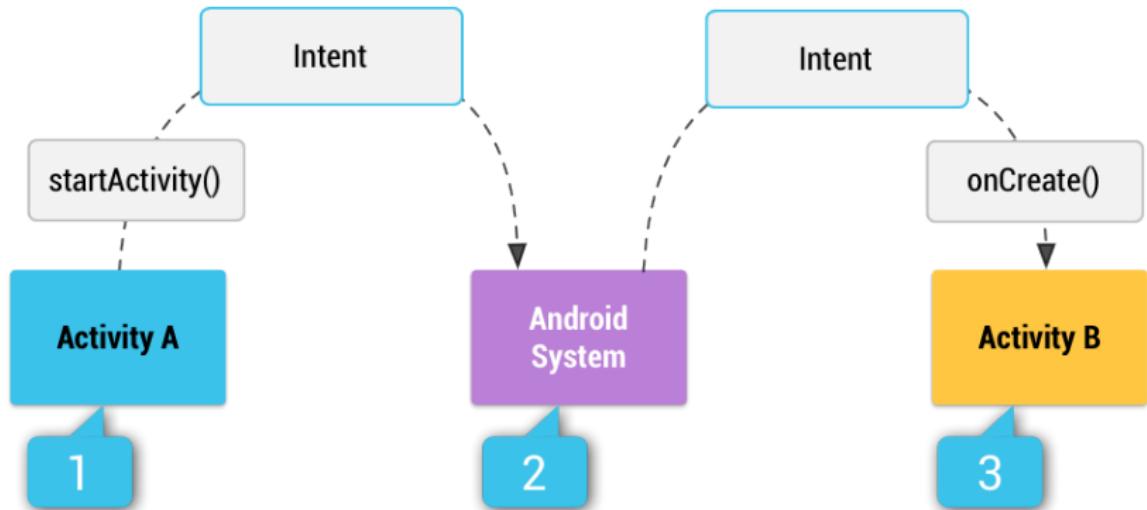
SMD

## Explicit Intents - Example

```
// Executed in an Activity, so 'this' is the Context  
// The fileUrl is a string URL  
Intent downloadIntent = new Intent(this, DownloadService.class);  
downloadIntent.setData(Uri.parse(fileUrl));  
startService(downloadIntent);
```

► Implicit intents

- ▶ Do not specify the exact component
- ▶ Declare a general action to be performed
- ▶ The Android system finds the appropriate component
- ▶ Compares the intent to the intent filters in the manifest of the apps
- ▶ Multiple components that match the intent
- ▶ Intent filters are mandatory





SMD

## Implicit Intents - Example

```
// Create the text message with a string.  
Intent sendIntent = new Intent();  
sendIntent.setAction(Intent.ACTION_SEND);  
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);  
sendIntent.setType("text/plain");  
  
// Try to invoke the intent.  
try {  
    startActivity(sendIntent);  
} catch (ActivityNotFoundException e) {  
    // Define what your app should do if  
    // no activity can handle the intent.  
}
```



SMD

## Intent Filter - Example

```
<activity android:name=".ExampleActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
    </intent-filter>
</activity>
```



Intents

Broadcast Receivers

Content Providers

Tools

Bibliography

- ▶ Handles broadcast messages
- ▶ No UI
- ▶ Broadcast messages:
  - ▶ Notifications/announcements
  - ▶ The system generates many broadcast messages
  - ▶ Examples: Battery is low, screen has turned off, phone has booted, etc.
  - ▶ Apps can send broadcasts to other apps or to themselves



SMD

- ▶ Always runs on the main UI thread
- ▶ Execute & return quickly
- ▶ Don't start threads or background services from the receiver
  - ▶ The system may kill the entire process after the receiver is completed
- ▶ For long running work schedule a JobService



SMD

- ▶ Each broadcast is delivered as an *Intent*
- ▶ Action of the intent defines the event
- ▶ Extra fields include additional information
- ▶ Intent passed to `sendBroadcast()` or  
`sendOrderedBroadcast()`



SMD

- ▶ Register a receiver in two ways
- ▶ Manifest-declared receivers
  - ▶ Statically in the manifest using the <receiver> tag
  - ▶ Start the app if it's not running already
  - ▶ For app specific broadcasts (API 26)
  - ▶ For some implicit broadcasts (ACTION\_BOOT\_COMPLETED)



SMD

- ▶ Context-registered receivers

- ▶ Dynamically using `Context.registerReceiver()`
- ▶ Available only when the context is valid
- ▶ e.g. Activity context , App context
- ▶ Unregister using `unregisterReceiver()`



SMD

- ▶ A foreground process will execute `onReceive()`
- ▶ After the method is executed, the system can kill the process
- ▶ Don't start threads from the receiver => they will be killed
- ▶ Schedule a `JobService` to do the work



SMD

- ▶ Normal broadcasts

- ▶ Completely Asynchronous
- ▶ All receivers run in an undefined order
- ▶ Don't propagate the results to other receivers
- ▶ `sendBroadcast()`



SMD

- ▶ Ordered broadcasts

- ▶ Delivered to one receiver at a time
- ▶ Each receiver executes and may propagate the result to the next or abort the broadcast
- ▶ The order is determined using the `android:priority` in the `<intent-filter>` of the receiver
- ▶ `sendOrderedBroadcast()`



## Broadcast Receiver Declaration in Manifest

```
<!-- If this receiver listens for broadcasts sent from the  
    system or from other apps, even other apps that you own,  
    set android:exported to "true". -->  
<receiver android:name=".MyBroadcastReceiver"  
          android:exported="false">  
    <intent-filter>  
        <action android:name="APP_SPECIFIC_BROADCAST" />  
    </intent-filter>  
</receiver>
```



```
public class MyBroadcastReceiver extends BroadcastReceiver {
    private static final String TAG = "MyBroadcastReceiver";
    @Override
    public void onReceive(Context context, Intent intent) {
        // Here you perform some operations
    }
}
```



SMD

## Sending Broadcast Messages

```
Intent intent = new Intent();
intent.setAction("com.example.APP_SPECIFIC_BROADCAST");
intent.putExtra("data", "Nothing_to_see_here,_move_along.");
sendBroadcast(intent);
```



Intents

Broadcast Receivers

Content Providers

Tools

Bibliography

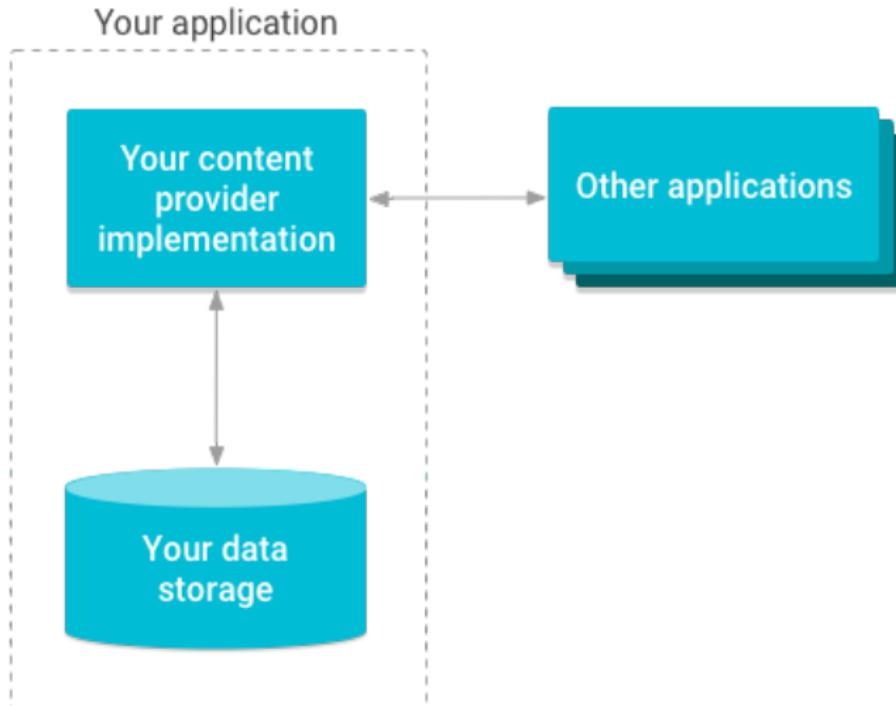


SMD

- ▶ Provides access to a repository of data
- ▶ Storing and sharing data
- ▶ Encapsulate data
- ▶ Allow other apps to securely access and modify your app data
- ▶ System Content Providers
  - ▶ Contacts, Dictionary, Settings, etc.



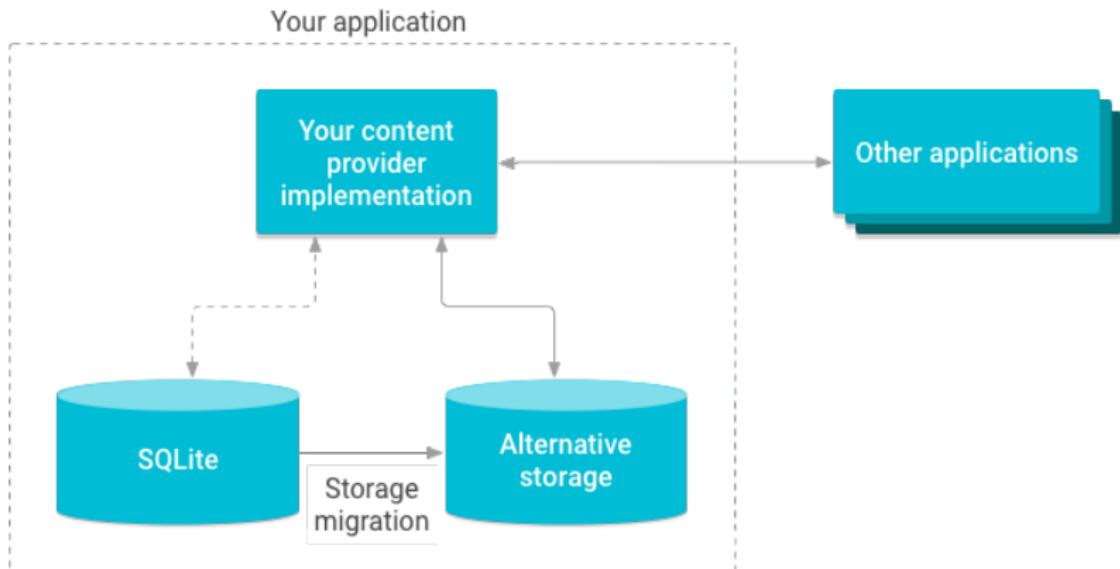
SMD





SMD

# Easy Storage Migration



- ▶ Two ways of storing data
  - ▶ File data - audio, video, photos
  - ▶ Structured data - database, array, etc.
    - ▶ Form compatible with tables of rows and columns
    - ▶ Usually a SQLite database

- ▶ Identify data in the provider
- ▶ Includes:
  - ▶ Authority = a symbolic name for the provider
  - ▶ Path = a name for the table or file
  - ▶ Optional row ID
  - ▶ URI for table: `content://<authority>/<path>`
  - ▶ URI for row: `content://<authority>/<path>/<id>`
- ▶ Usually authority is based on the package name



SMD

- ▶ Package name: com.example.app
- ▶ Authority: com.example.app.provider
- ▶ Table: table3
- ▶ URI: content://com.example.app.provider/table3
- ▶ Row: 6
- ▶ URI: content://com.example.app.provider/table3/6



- ▶ `onCreate()` - initialize provider
- ▶ Data access methods:
- ▶ `query()` - retrieve data from provider
- ▶ `insert()` - insert a new row into provider
- ▶ `update()` - update existing row in provider
- ▶ `delete()` - delete existing rows in provider
- ▶ all data access methods receive an URI as argument



SMD

- ▶ Interface for accessing data from another process
  - ▶ Provider and client
  - ▶ The app that owns the data includes the Content Provider
  - ▶ The client app includes the Content Resolver
- ▶ Access data using a *ContentResolver* client object
  - ▶ Methods: `query()`, `insert()`, `update()`, `delete()`
  - ▶ Calls the methods with the same name in the *ContentProvider* object



SMD

## Using Content Resolver

```
mCursor = getContentResolver().query(  
    UserDictionary.Words.CONTENT_URI,  
    projection, // URI  
    selectionClause, // columns to return  
    selectionArgs, // criteria for selection  
    sortOrder); // args for the selection  
    // sort order  
[...]  
newUri = getContentResolver().insert(  
    UserDictionary.Words.CONTENT_URI, // URI  
    newValues // values to insert  
);  
[...]  
rowsUpdated = getContentResolver().update(  
    UserDictionary.Words.CONTENT_URI, // URI  
    updateValues, // columns to update  
    selectionClause, // column to select on  
    selectionArgs // value to compare to  
);
```



SMD

## Permissions to Access a Provider

- ▶ The app that wants to access the Content Provider must have permissions
- ▶ For accessing user dictionary provider:
- ▶ `<uses-permission android:name="android.permission.READ_USER_DICTIONARY">`



SMD

- ▶ The provider may have one or more permission elements in Manifest
- ▶ Permissions unique to that provider
  - ▶ com.example.app.provider.permission.READ\_PROVIDER
- ▶ From general to fine grained permissions
  - ▶ Entire provider, a table, some rows



SMD

- ▶ Single read-write provider-level permission
  - ▶ android:permission
- ▶ Separate read and write provider-level permission
  - ▶ android:readPermission and android:writePermission
- ▶ Path-level permission
  - ▶ <path-permission> to specify URI
  - ▶ Permissions for specific URIs
- ▶ Temporary permission
  - ▶ Delegate temporary access to an application
  - ▶ android:grantUriPermissions or  
<grant-uri-permission>



SMD

## Content Provider Declaration in Manifest

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

Intents

Broadcast Receivers

Content Providers

Tools

Bibliography



# SMD

# Android Studio

- ▶ Official IDE
- ▶ Gradle-based build system

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure for "My Application2". It includes the "app" module with "manifests", "java" (containing "com.example.myapplication" with "MainActivity"), "res", and "Gradle Scripts".
- Code Editor:** The main window displays the "MainActivity.java" file under the "My Application - MainActivity.java [My\_Application.app]" tab. The code is as follows:

```
1 package com.example.myapplication;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12
13 }
14 }
```

- Build Output:** The bottom pane shows the build log:

```
Sync
MyApplication2: finished at 3/7/21 2:04 PM      7 s 695 ms > Task :prepareKotlinBuildScriptModel UP-TO-DATE
BUILD SUCCESSFUL in 6s
```
- Bottom Navigation:** The navigation bar includes tabs for TODO, Terminal, Build, Logcat, Profiler, Database Inspector, Event Log, Layout Inspector, and a status message: "daemon started successfully (2 minutes ago)".



## Android SDK Manager

► Download SDK packages, samples, emulator images, tools

Appearance & Behavior > System Settings > Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/laura/Library/Android/sdk

Edit Optimize disk space

SDK Platforms SDK Tools SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
Android S Preview			
Android SDK Platform S	S	1	Not installed
Google APIs ARM v8a System Image	S	1	Not installed
Google APIs Intel x86 Atom_64 System Image	S	1	Not installed
Google Play ARM v8a System Image	S	1	Not installed
Google Play Intel x86 Atom_64 System Image	S	1	Not installed
Android 11.0 (R)			
Android SDK Platform 30	30	3	Installed
Sources for Android 30	30	1	Installed
Google APIs ARM v8a System Image	30	10	Not installed
Google APIs Intel x86 Atom System Image	30	9	Installed
Google APIs Intel x86 Atom_64 System Image	30	10	Not installed
Google Play ARM v8a System Image	30	9	Not installed
Google Play Intel x86 Atom System Image	30	9	Not installed
Google Play Intel x86 Atom_64 System Image	30	10	Not installed
Android 10.0 (Q)			
Android SDK Platform 29	29	5	Installed
Sources for Android 29	29	1	Installed
Automotive with Play Store Intel x86 Atom System Image	29	1	Not installed
Android TV Intel x86 Atom System Image	29	3	Not installed
Intel x86 Atom System Image	29	7	Not installed
Intel x86 Atom_64 System Image	29	7	Not installed
Google APIs Intel x86 Atom System Image	29	11	Not installed
Google APIs Intel x86 Atom_64 System Image	29	11	Not installed
Google Play Intel x86 Atom System Image	29	8	Not installed
Google Play Intel x86 Atom_64 System Image	29	8	Not installed
Android 9.0 (Pie)			
Android SDK Platform 28	28	6	Installed
Sources for Android 28	28	1	Installed
Automotive Intel x86 Atom System Image	28	5	Not installed

Hide Obsolete Packages  Show Package Details

Cancel Apply OK



## AVD Manager

- ▶ Manages Android Virtual Devices (for emulator)
- ▶ Emulator
  - ▶ Virtual mobile devices running on a PC

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
Pixel	Pixel_3a_API_30_x86		1080 × 2220: 440dpi	30	Android 11.0 (Googl...)	x86	8.9 GB	
Pixel	Pixel 4 API 30		1080 × 2280: 440dpi	30	Android 11.0 (Googl...)	x86	513 MB	



- ▶ QEMU
- ▶ Screen, Keyboard, Network, Audio, GPS, Radio
- ▶ Can be accelerated through virtualization
  - ▶ x86 System Image
  - ▶ Intel Hardware Accelerated Execution Manager (HAXM) on Windows and MacOS
  - ▶ KVM on Linux
- ▶ GPU accelerated

- ▶ Communication between the development tools and (virtual) device
- ▶ Three components
  - ▶ Client: runs on the development machine
  - ▶ Server: background process on the development machine
  - ▶ Daemon: background process on the (virtual) device
- ▶ Copy files (adb push, adb pull)
- ▶ Install apps (adb install)
- ▶ Debug (adb logcat)
- ▶ Shell on the (virtual) device (adb shell)

Intents

Broadcast Receivers

Content Providers

Tools

Bibliography



- <https://developer.android.com/guide/components/intents-filters>
- ▶ <https://developer.android.com/guide/components/broadcasts>
- ▶ <https://developer.android.com/guide/topics/providers/content-providers>
- ▶ <https://developer.android.com/guide/topics/providers/content-provider-basics>
- ▶ <https://developer.android.com/guide/topics/providers/content-provider-creating>
- ▶ <https://developer.android.com/studio/run/advanced-emulator-usage>
- ▶ <https://developer.android.com/studio/command-line/adb>



- ▶ Intent
- ▶ Implicit Intents
- ▶ Explicit Intents
- ▶ Broadcast Message
- ▶ Broadcast Receiver
- ▶ Content Provider
- ▶ Content Resolver
- ▶ Content URI
- ▶ Android Studio
- ▶ SDK Manager
- ▶ AVD Manager
- ▶ Emulator
- ▶ ADB