Android SDK (2)

Lecture 3

Security of Mobile Devices

2023

Intents

Broadcast Receivers

Content Providers

Tools

Bibliography

Outline

Intents

Broadcast Receivers

Content Providers

Tools

Bibliography

Intent

- An object used for delivering a message or executing an action
- 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

Intent Filters

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

Intents - Use Cases

- 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()`
- Give temporary permissions to another app to access a Content URI

Types of Intents

- 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

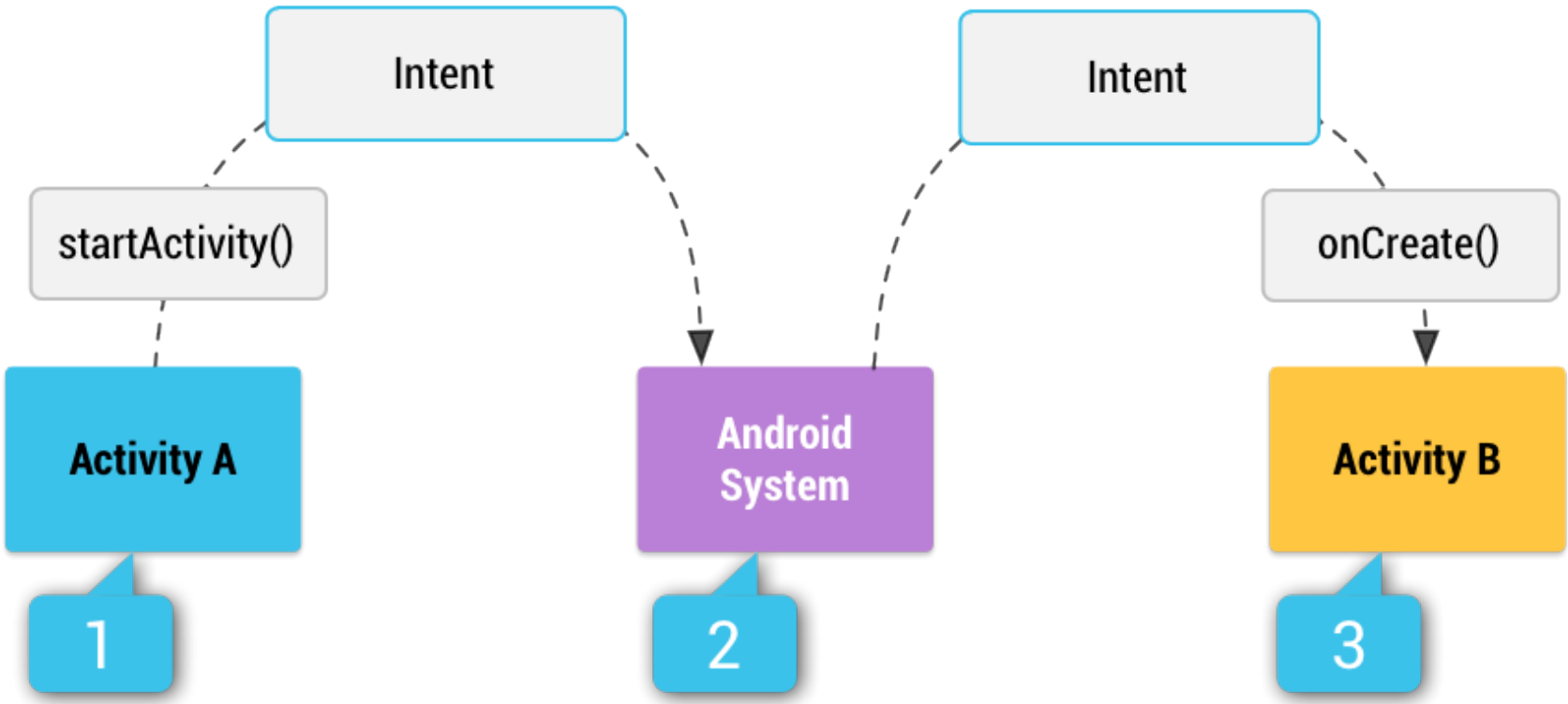
Explicit Intents - Example

```
// 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);
```

Types of Intents

- 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

Implicit Intents



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.
}
```

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>
```

Outline

Intents

Broadcast Receivers

Content Providers

Tools

Bibliography

Broadcast Receiver

- 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

Broadcast Receiver

- 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

Broadcast Receiver

- 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()`

Register a Receiver

- Register a receiver in two ways:
- Manifest-declared receivers - declared in the Manifest file
- Context-registered receivers - registered dynamically

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`)

Lifetime of Manifest-declared Receiver

- Scenario: the app is not running already
- 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

Context-registered Receivers

- Dynamically using `Context.registerReceiver()`
- Available only when the context is valid
- Activity context - while activity is not destroyed
- App context - while app is still running
- Unregister using `unregisterReceiver()`

Types of Broadcasts

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

Types of Broadcasts

- 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>
```

Broadcast Receiver Implementation

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

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);
```

Outline

Intents

Broadcast Receivers

Content Providers

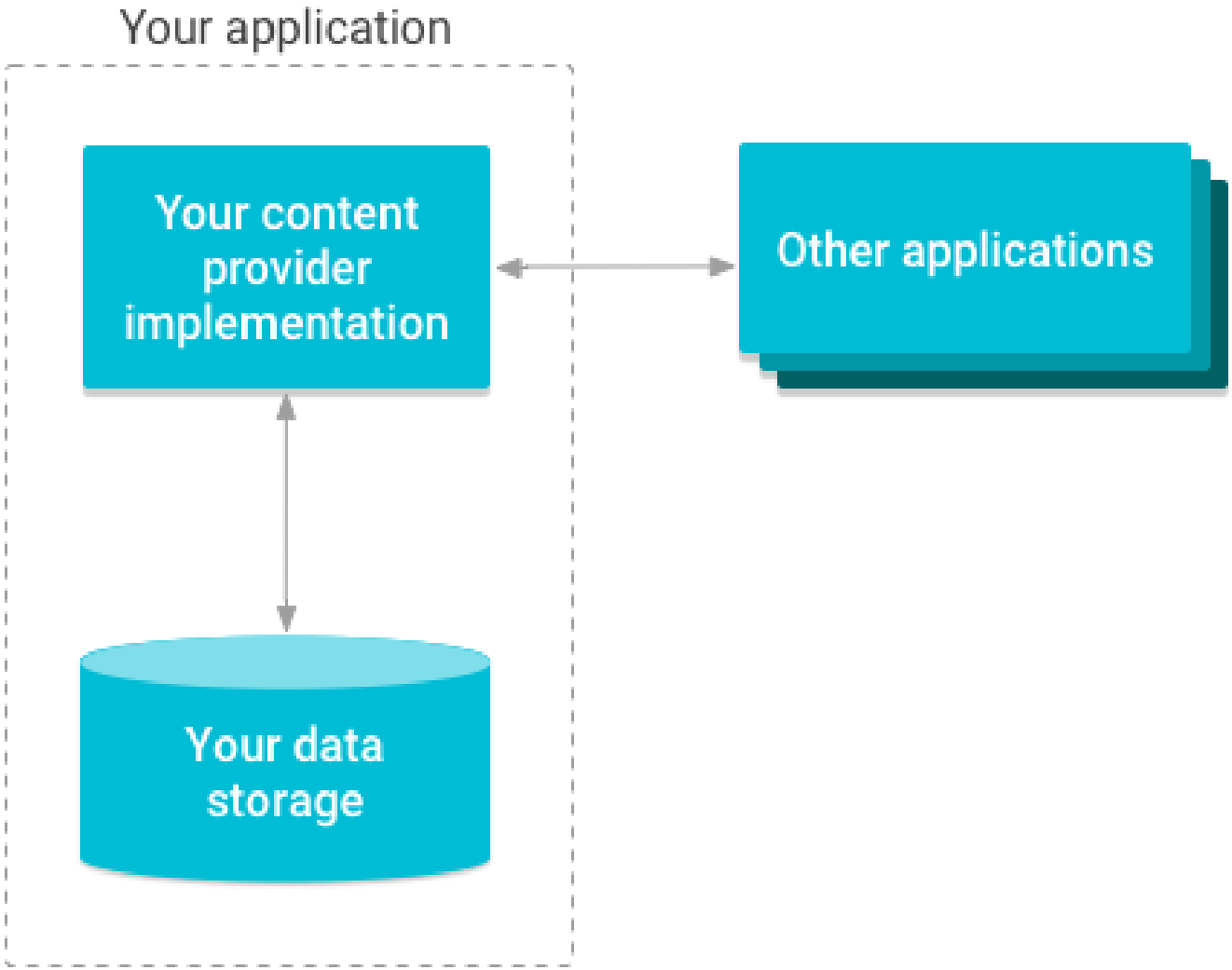
Tools

Bibliography

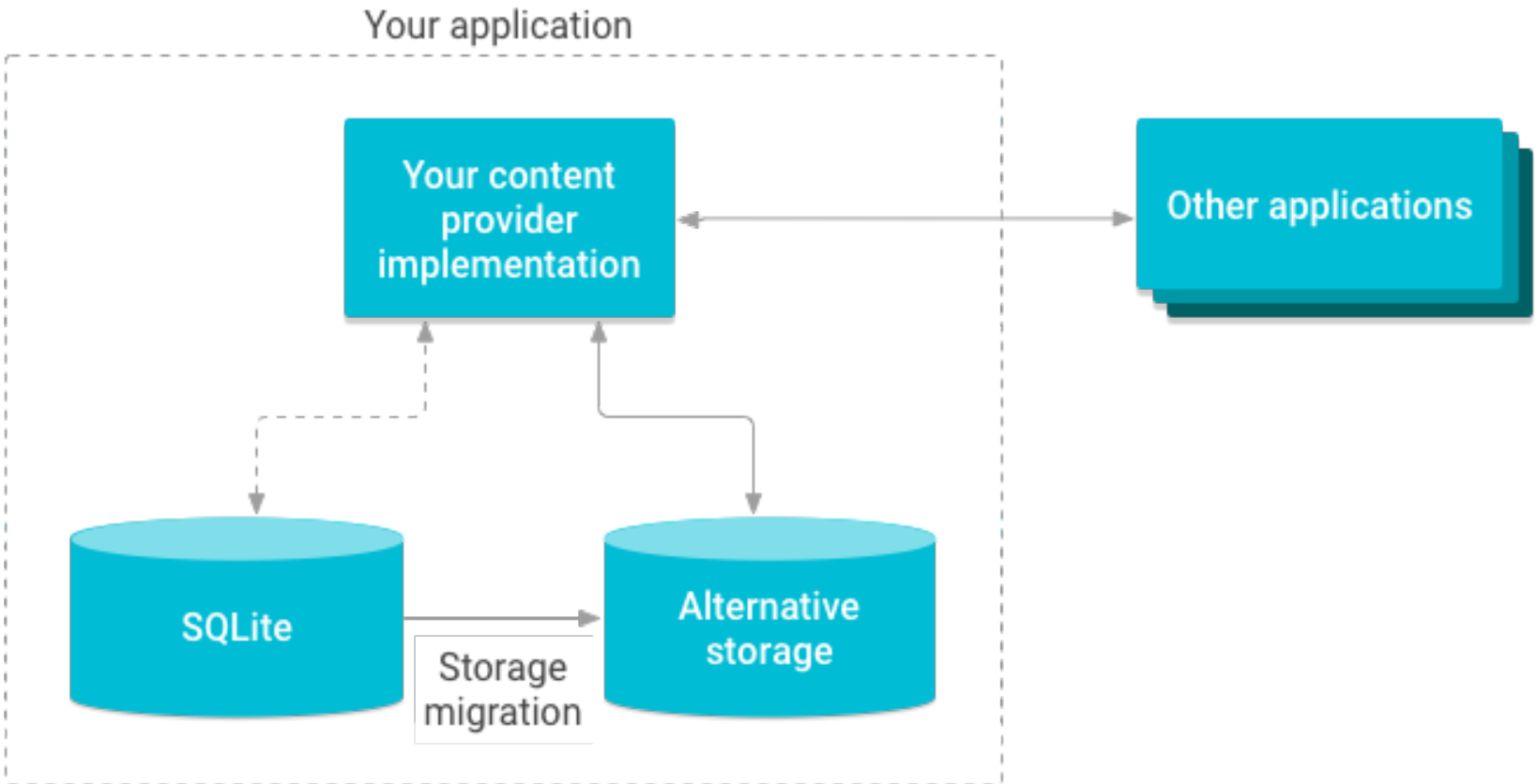
Content Provider

- 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.

Sharing Data



Easy Storage Migration



Content Provider

- 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

Content URIs

- 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

Content URI Examples

- 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`

Content Provider Methods

- Abstract class `ContentProvider`
- `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

Content Resolver

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

Using Content Resolver

```
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI,   // URI
    projection,                         // columns to return
    selectionClause,                    // criteria for selection
    selectionArgs,                      // args for the selection
    sortOrder);                         // 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
);
```

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">`

Content Provider Permissions

- 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

Content Provider Permissions

- 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>`

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>
```

Outline

Intents

Broadcast Receivers

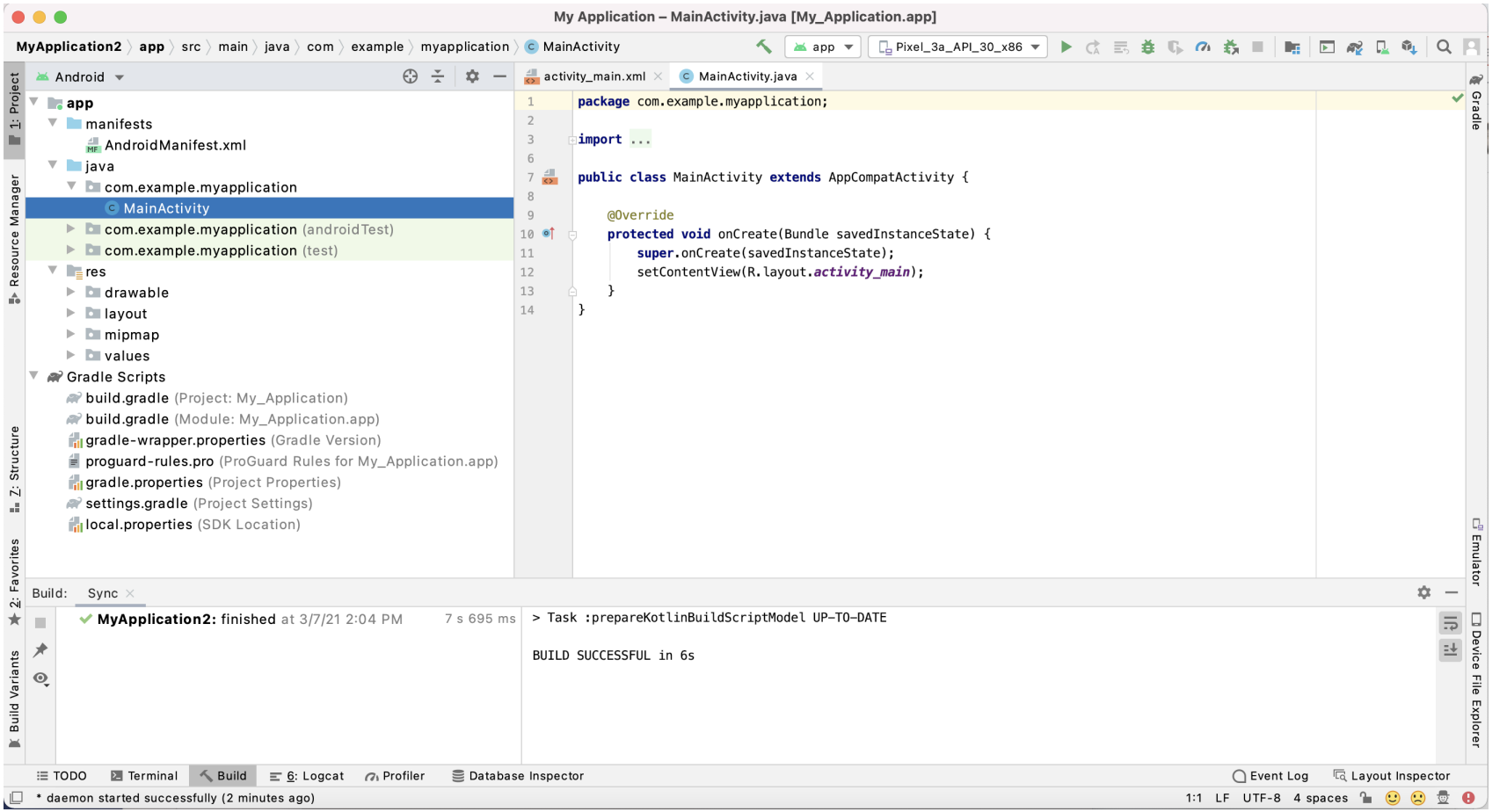
Content Providers

Tools

Bibliography

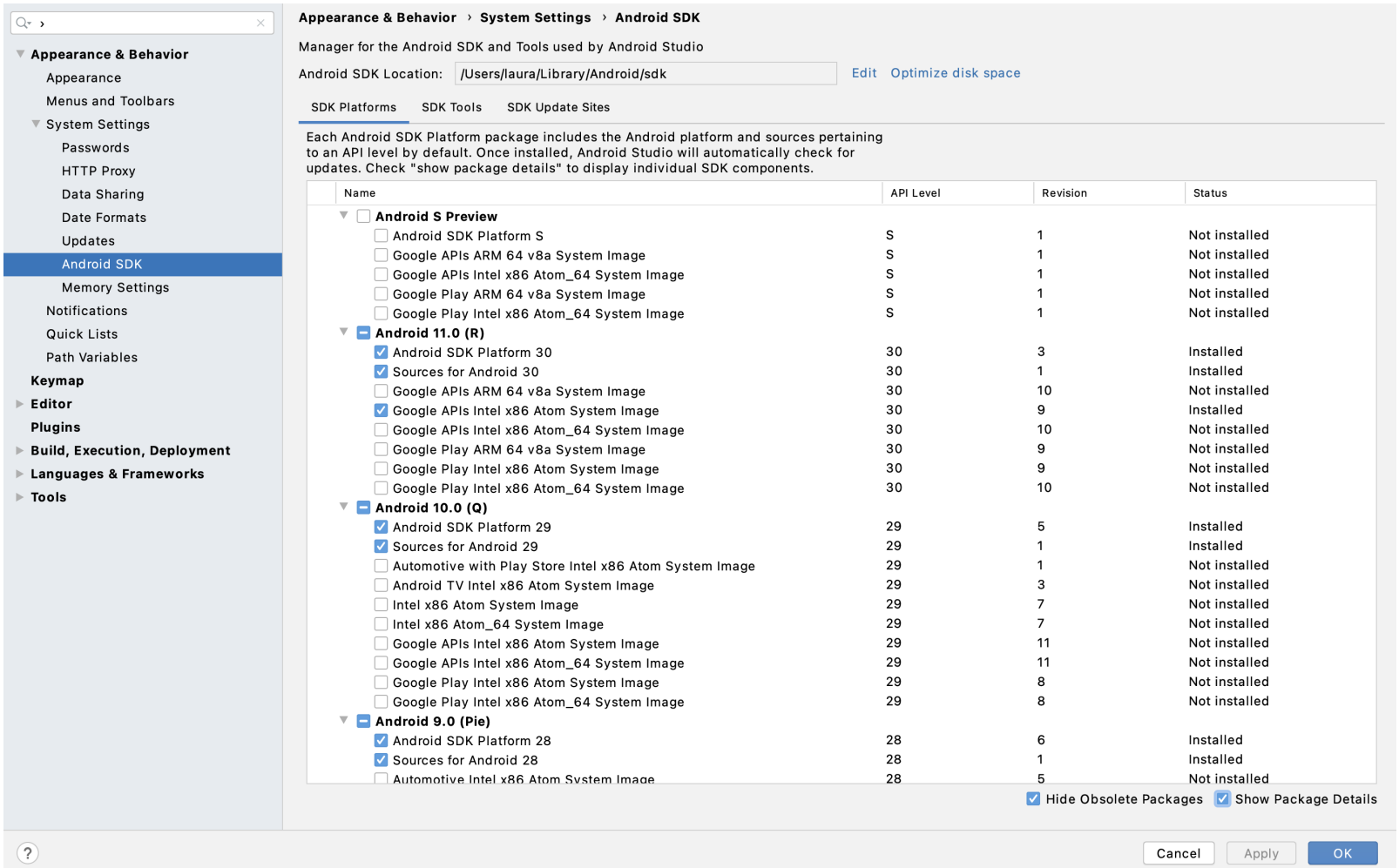
Android Studio

- Official IDE
- Gradle-based build system

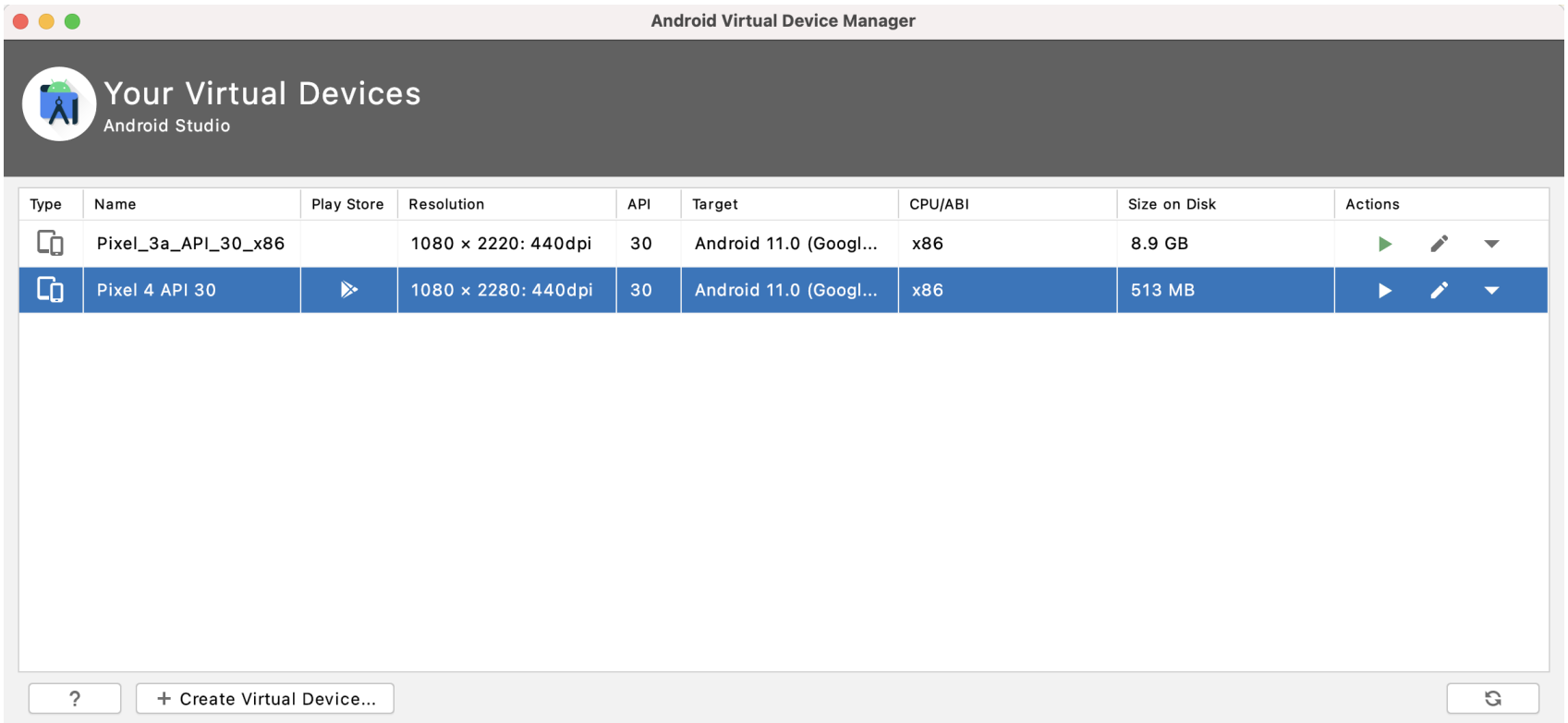


Android SDK Manager

- Download SDK packages, samples, emulator images, tools



- AVD Manager
  - Manages virtual devices for the emulator



Emulator

- Virtual mobile devices running on a PC
- Screen, Keyboard, Network, Audio, GPS, Radio
- Based on QEMU

Android Debug Bridge

- 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`)

Outline

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

Keywords

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