2017 Android of WMNTAA

App components

App components

- Activity
- Service
- Broadcast receiver
- Content provider

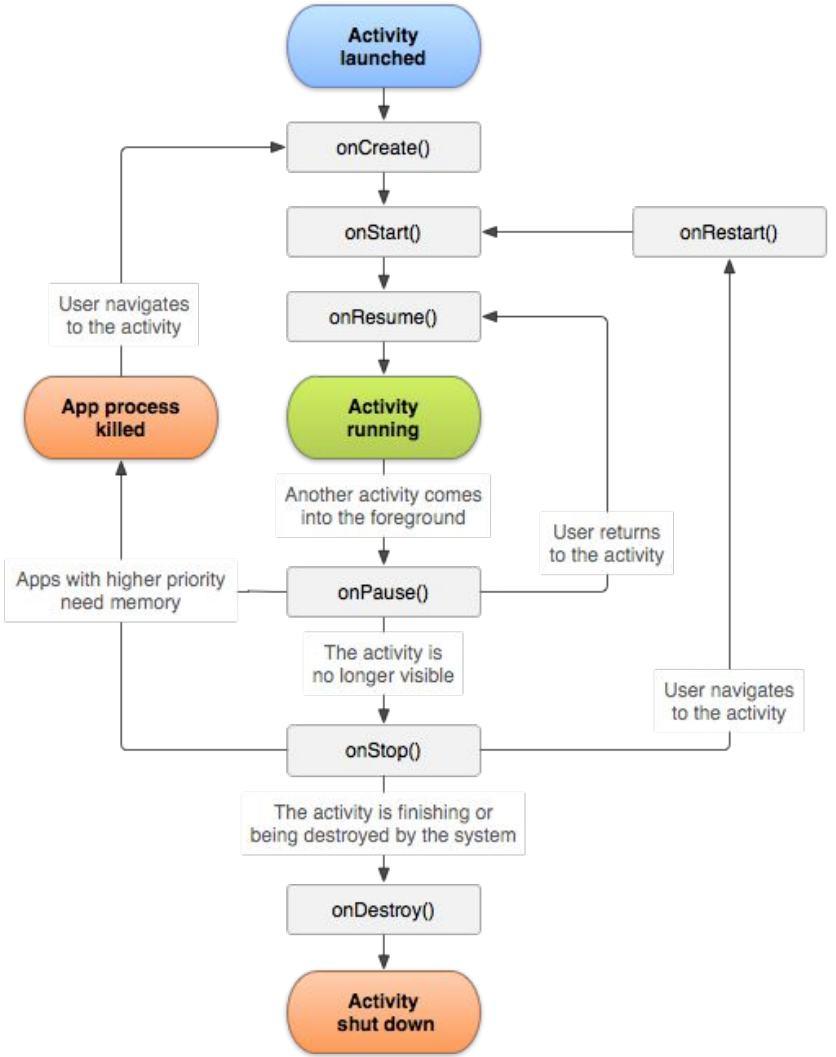
App components

- App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others.

Activity

- Activity Lifecycle
- Create an Activity

Activity Lifecycle



Create an Activity

- Declare activities
- Declare intent filters
- Declare permissions
- Start an Activity

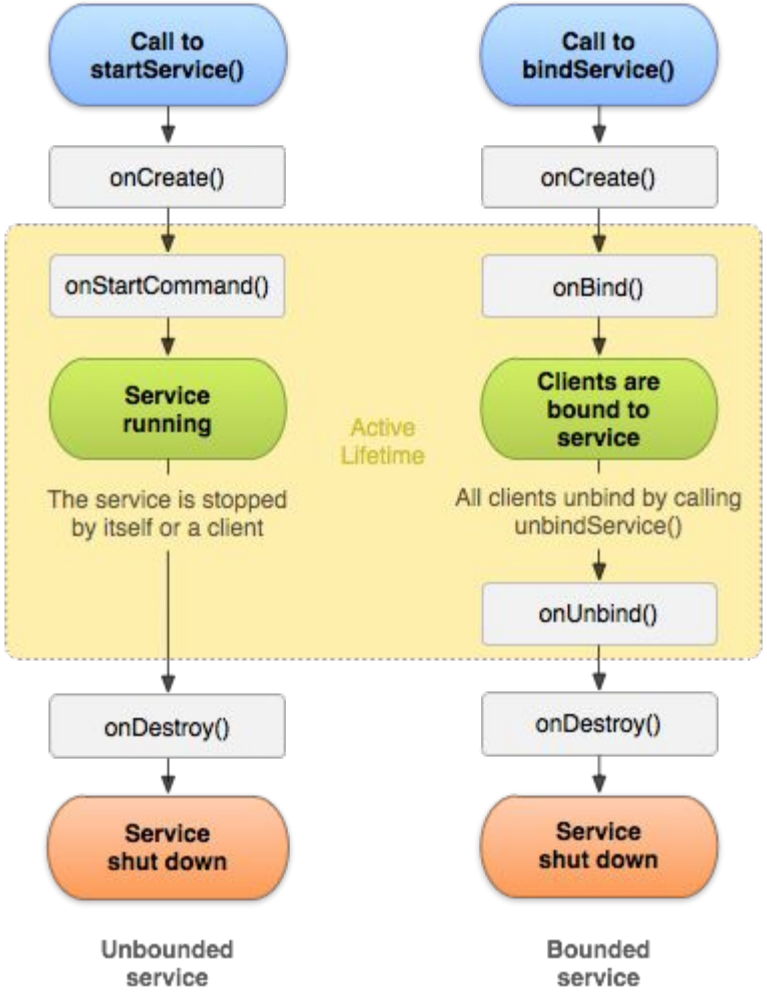
Service

- Overview
- Service Lifecycle
- Create a Service

Overview

- A Service is an application component that can perform long-running operations in the background, and it does not provide a user interface.
- Another application component can start a service, and it continues to run in the background even if the user switches to another application.
- Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

Service Lifecycle



Create a Service

- Declaring a service in the manifest
- Creating a started service
- Starting a service

Declaring a service in the manifest

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

Creating a started service

- onStartCommand()
  - The system invokes this method by calling startService() when another component (such as an activity) requests that the service be started.
- onBind()
  - The system invokes this method by calling bindService() when another component wants to bind with the service (such as to perform RPC). However, if you don't want to allow binding, you should return null.
- onCreate()
  - The system invokes this method to perform one-time setup procedures when the service is initially created (before it calls either onStartCommand() or onBind()).
- onDestroy()
  - The system invokes this method when the service is no longer used and is being destroyed.

Creating a started service

```
public class HelloService extends Service {
  private Looper mServiceLooper;
  private ServiceHandler mServiceHandler;

  // Handler that receives messages from the thread
  private final class ServiceHandler extends Handler {
      public ServiceHandler(Looper looper) {
          super(looper);
      }
      @Override
      public void handleMessage(Message msg) {
          // Normally we would do some work here, like download a file.
          // For our sample, we just sleep for 5 seconds.
          try {
              Thread.sleep(5000);
          } catch (InterruptedException e) {
              // Restore interrupt status.
              Thread.currentThread().interrupt();
          }
          // Stop the service using the startId, so that we don't stop
          // the service in the middle of handling another job
          stopSelf(msg.arg1);
      }
  }
```

Creating a started service

```
@Override
public void onCreate() {
  // Start up the thread running the service.  Note that we create a
  // separate thread because the service normally runs in the process's
  // main thread, which we don't want to block.  We also make it
  // background priority so CPU-intensive work will not disrupt our UI.
  HandlerThread thread = new HandlerThread("ServiceStartArguments",
          Process.THREAD_PRIORITY_BACKGROUND);
  thread.start();

  // Get the HandlerThread's Looper and use it for our Handler
  mServiceLooper = thread.getLooper();
  mServiceHandler = new ServiceHandler(mServiceLooper);
}
```

Creating a started service

```
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show();

        // For each start request, send a message to start a job and deliver the
        // start ID so we know which request we're stopping when we finish the job
        Message msg = mServiceHandler.obtainMessage();
        msg.arg1 = startId;
        mServiceHandler.sendMessage(msg);

        // If we get killed, after returning from here, restart
        return START_STICKY;
    }

    @Override
    public IBinder onBind(Intent intent) {
        // We don't provide binding, so return null
        return null;
    }

    @Override
    public void onDestroy() {
      Toast.makeText(this, "service done", Toast.LENGTH_SHORT).show();
    }
}
```

Starting a service

You can start a service from an activity or other application component by passing an Intent (specifying the service to start) to startService(). The Android system calls the service's onStartCommand() method and passes it the Intent.

```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
```

Broadcast receiver

- Overview
- Receiving broadcasts
- Sending broadcasts

Overview

- Android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the publish-subscribe design pattern. These broadcasts are sent when an event of interest occurs. For example, the Android system sends broadcasts when various system events occur, such as when the system boots up or the device starts charging.
- Apps can also send custom broadcasts, for example, to notify other apps of something that they might be interested in (for example, some new data has been downloaded).

Receiving broadcasts

- Manifest-declared receivers
- Context-registered receivers

Manifest-declared receivers

Specify the <receiver> element in your app's manifest.

```
<receiver android:name=".MyBroadcastReceiver"  android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />
    </intent-filter>
</receiver>
```

Manifest-declared receivers

Subclass BroadcastReceiver and implement onReceive(Context, Intent).

```
public class MyBroadcastReceiver extends BroadcastReceiver {
    private static final String TAG = "MyBroadcastReceiver";
    @Override
    public void onReceive(Context context, Intent intent) {
        StringBuilder sb = new StringBuilder();
        sb.append("Action: " + intent.getAction() + "\n");
        sb.append("URI: " + intent.toUri(Intent.URI_INTENT_SCHEME).toString() + "\n");
        String log = sb.toString();
        Log.d(TAG, log);
        Toast.makeText(context, log, Toast.LENGTH_LONG).show();
    }
}
```

Context-registered receivers

1. Create an instance of BroadcastReceiver.
2. Create an IntentFilter and register the receiver by calling registerReceiver(BroadcastReceiver, IntentFilter).

```
BroadcastReceiver br = new MyBroadcastReceiver();
```

```
IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
intentFilter.addAction(Intent.ACTION_AIRPLANE_MODE_CHANGED);
this.registerReceiver(br, filter);
```

Sending broadcasts

The following code snippet demonstrates how to send a broadcast by creating an Intent and calling sendBroadcast(Intent).

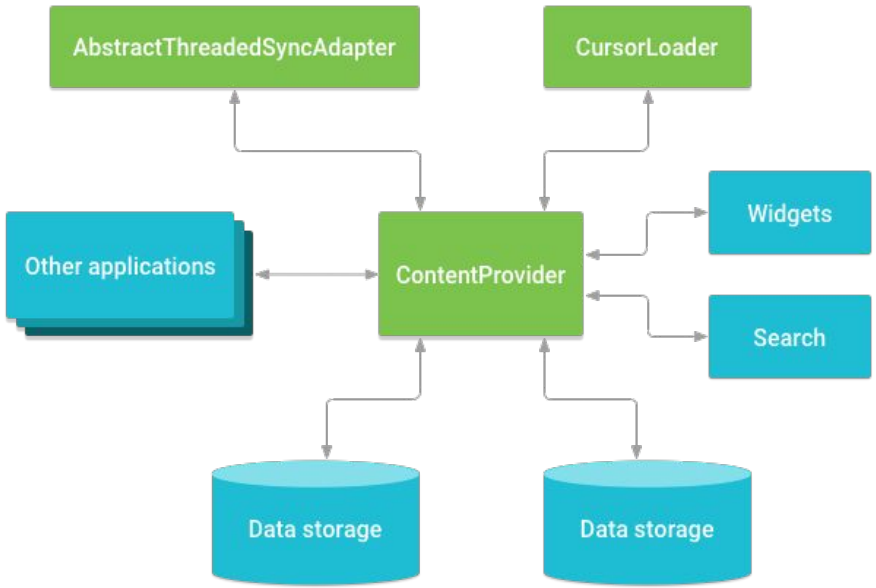
```
Intent intent = new Intent();
intent.setAction("com.example.broadcast.MY_NOTIFICATION");
intent.putExtra("data","Notice me senpai!");
sendBroadcast(intent);
```

Content provider

- Overview
- Accessing a provider

Overview

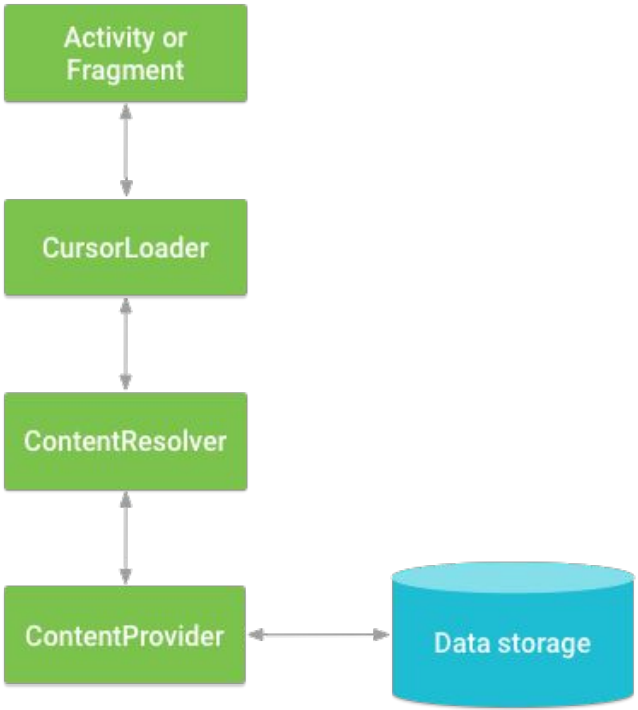
- A content provider presents data to external applications as one or more tables that are similar to the tables found in a relational database.
- A row represents an instance of some type of data the provider collects, and each column in the row represents an individual piece of data collected for an instance.



Accessing a provider

One of the built-in providers in the Android platform is the user dictionary, which stores the spellings of non-standard words that the user wants to keep.

word	app id	frequency	locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5



Accessing a provider

To get a list of the words and their locales from the User Dictionary Provider, you call ContentResolver.query()

```
// Queries the user dictionary and returns results
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI,   // The content URI of the words table
    mProjection,                        // The columns to return for each row
    mSelectionClause                    // Selection criteria
    mSelectionArgs,                     // Selection criteria
    mSortOrder);                        // The sort order for the returned rows
```

Accessing a provider

```
/*
 * This defines a one-element String array to contain the selection argument.
 */
String[] mSelectionArgs = {""};

// Gets a word from the UI
mSearchString = mSearchWord.getText().toString();

// Remember to insert code here to check for invalid or malicious input.

// If the word is the empty string, gets everything
if (TextUtils.isEmpty(mSearchString)) {
    // Setting the selection clause to null will return all words
    mSelectionClause = null;
    mSelectionArgs[0] = "";

} else {
    // Constructs a selection clause that matches the word that the user entered.
    mSelectionClause = UserDictionary.Words.WORD + " = ?";

    // Moves the user's input string to the selection arguments.
    mSelectionArgs[0] = mSearchString;

}
```

Accessing a provider

```
// Does a query against the table and returns a Cursor object
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI,  // The content URI of the words table
    mProjection,                       // The columns to return for each row
    mSelectionClause                   // Either null, or the word the user entered
    mSelectionArgs,                    // Either empty, or the string the user entered
    mSortOrder);                       // The sort order for the returned rows

// Some providers return null if an error occurs, others throw an exception
if (null == mCursor) {
    /*
     * Insert code here to handle the error. Be sure not to use the cursor! You may want to
     * call android.util.Log.e() to log this error.
     *
     */
// If the Cursor is empty, the provider found no matches
} else if (mCursor.getCount() < 1) {

    /*
     * Insert code here to notify the user that the search was unsuccessful. This isn't necessarily
     * an error. You may want to offer the user the option to insert a new row, or re-type the
     * search term.
     */

} else {
    // Insert code here to do something with the results

}
```

Challenge

- Use Content provider in a Service to get contacts
- Send broadcast from Service to Activity to notify job done
- Get contact from service with broadcast messages
- Show contacts on Activity

How to get contacts :
https://developer.android.com/guide/topics/providers/contacts-provider.html