Android Application Fundamentals

Informatics 133

11/22/10

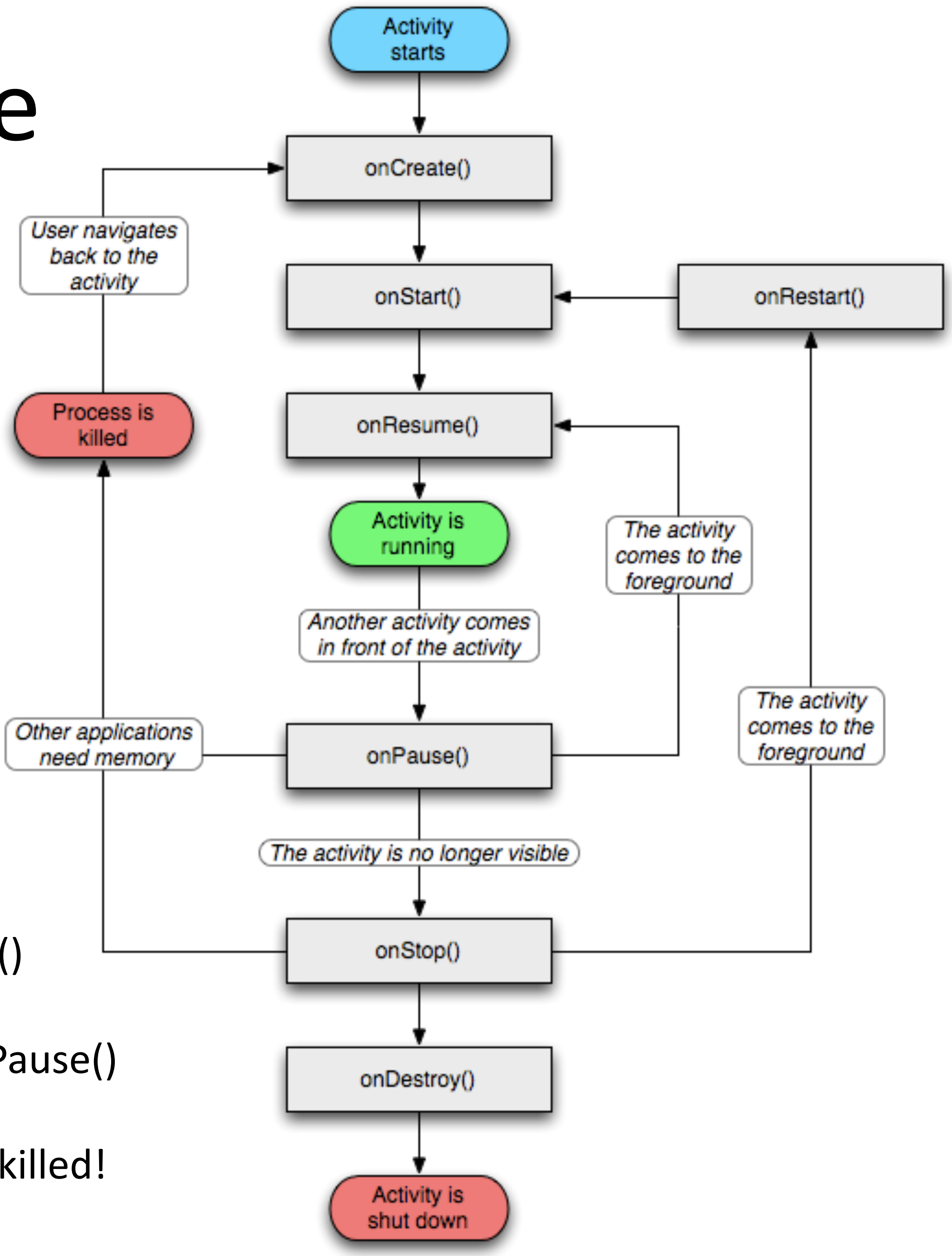
Mohamad Monibi

Application Components

- Activities
  - a visual user interface for one focused endeavor the user can undertake
- Services
  - doesn't have a visual user interface, but rather runs in the background for an indefinite period of time
- Broadcast Receivers
  - receives and reacts to broadcast announcements
- Content Providers
  - makes a specific set of the application's data available to other applications

"Android applications don't have a single entry point for everything in the application (no main() function, for example). Rather, they have essential *components* that the system can instantiate and run as needed."

Activity Lifecycle



Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Key Loops:

•Entire lifetime: onCreate() – onDestroy()

•Visible lifetime: onStart() – onStop()

•Foreground lifetime: onResume() – onPause()

Once onPause is called, activity may be killed!

Intents

- abstract description of an operation to be performed
- asynchronous message that provides runtime binding between application components (even across different applications)
- Main attributes:
  - action (ACTION_VIEW, ACTION_IMAGE_CAPTURE)
  - data (http://i.imgur.com/E5166.jpg, content://contacts/people/1)
- Intent resolution
  - Explicit intents: specifies the exact component to be run
  - Implicit intents: uses intent-filters to match components to intents based on the attributes

Starting Activities

- Context.startActivity(Intent i)
  - Intent i = new Intent(Class class)
  - Intent i = new Intent(Action action)

- Context.startActivityForResult(Intent i)
  - calling activity's onActivityResult() is called after new activity exits

- Home Screen (Launcher)
  - specified in AndroidManifest.xml

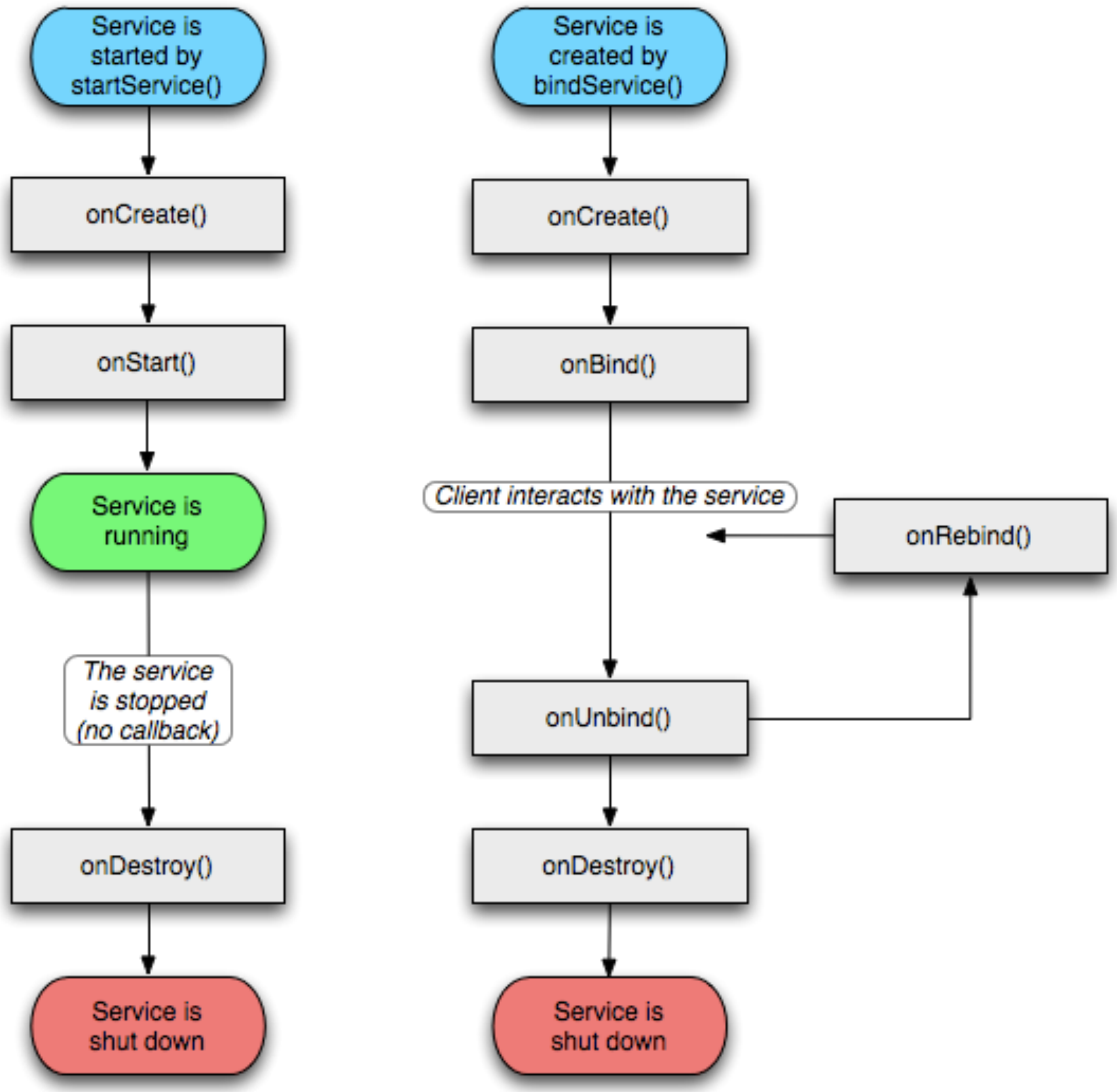
```xml
<intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
```

* All activities (all components actually) must be defined in manifest!

Service Lifecycle

startService():
runs until someone stops it or it stops itself

bindService():
is operated programmatically using an interface it defines and exports (can also start the service)


Service is started by startService()
onCreate()
onStart()
Service is running
The service is stopped (no callback)
onDestroy()
Service is shut down
Service is created by bindService()
onCreate()
onBind()
Client interacts with the service
onRebind()
onUnbind()
onDestroy()
Service is shut down

Using Services

- Services that expose an interface (which we connect to using bindService()) can be local using direct method call, or remote using AIDL.

- Services do not run in a separate process or even **thread!**
  - start a new thread in the onCreate() callback

- Painless threading:
  - AsyncTask
    - provides three callbacks
      - onPreExecute()
      - doInBackground()
      - onProgressUpdate()
      - onPostExecute()
  - IntentService
    - each intent passed to startService() is handled in turn using a worker thread

- Wakelocks!!!!!!!!!!!!!!!!

Broadcast Receiver

- Intents can be broadcast
  - by the system (ACTION_BATTERY_LOW)
  - by you (ACTION_MY_TASK_DONE)

- Guess who receives broadcasts?

- onReceive() is the only callback method
  - runs with foreground priority
    - must return within short period of time
  - typically show a notification or start a service

- Receiver registration
  - manifest (using intent-filters)
    - can launch your app even if not currently running
    - *some* system broadcasts can't be registered for here!
  - code (Context.registerReceiver())
    - must manage registration manually (call unregisterReceiver())
    - only works when component is active

Tip: setup Alarms that broadcast intents to do scheduled tasks (see AlarmManager)

Content Providers

- Android apps live in separate worlds
  - seperate process
  - seperate virual machine
  - separate linux user id

- Files and data are only visible to owning app

- Use ContentProviders to share data with other applications
  - Underlying data storage mechanism is irrelevant and invisible to callers
  - Provides an interface very similar to databases we are used to: query, insert, update, delete
  - These calls must be implemented in the ContentProvider

Data Storage

- Shared Preferences
  - primitive key-value pairs (private)
- Internal Storage
  - private
- External Storage
  - public
- SqLite Database
  - private
- Cloud

*See “Developing Android REST client applications” talk from Google I/O 2010.