Application Fundamentals

CS 2046

Mobile Application Development

Fall 2010

Announcements

- CMS is up
  - If you did not get an email regarding this, see me after class or send me an email.
- Still working on room for office hours.
- Lab Session: Monday 10/25, Upson B7 (Regular class time)

Sites of Interest

- Android Developers Blog: http://android-developers.blogspot.com/?hl=en
- Android Central: http://www.androidcentral.com/
- Both of these are on the Resources page on the course website.

Recap

- Components:
  - Activity
  - Service
  - ContentProvider
  - BroadcastReceiver
- Intents:
  - Action, Data = Implicit
  - Action, Data, Component = Explicit

Additional Components of Intents

- Category: describe the kind of component that should handle the intent.
  - CATEGORY_LAUNCHER: Activity should appear in launcher.
  - CATEGORY_PREFERENCE: Activity defines preferences for an application.
- Extras: Key-value pairs for additional information.
  - ACTION_HEADSET_PLUG: “state” for plug or unplug
- Flags: Mainly, instruct system how to launch Activity
  - FLAG_ACTIVITY_NO_ANIMATION: no animation when launching new activities

Accessing Application Components

- How do Activities, Services, etc. get launched?

- In Java:
  - Write a Class to perform some task.
    - This is still the same.
  - Write a main method which calls a constructor of Class and runs any method.
    - This is *not* the way Android works.
    - Depending on the type of object, Android will call the constructor and manage its lifecycle.

Contexts

- The Context class provides access to system functions and services.
  - Most functions which involve the Android framework require use of a Context object.
- Activities and Services extend Context, so they can call methods from the class directly.
- BroadcastReceivers receive an input Context for their event handling function.
- ContentProviders call getContext() to obtain a Context object.

Activities

- Launch an Activity by calling startActivity(Intent)
- Subactivities: startActivityForResult
  - Takes an Intent as input, as well as an integer code.
  - When subactivity exits, returns a result code.
  - Original activity’s onActivityResult method is called.
  - Note – this is asynchronous (non-blocking).
- Activities form a stack
  - New activities appear at the top of the stack (i.e. on screen)
  - Pressing back button goes to previous activity (usually)

Tasks

- Android groups the activities of one application into a single task = stack of related activities.

- User presses HOME and launches a new application:
  - Moves current task to background
  - Starts new task, puts default Activity for new application on top of stack.
- If application is resumed, the old task (with previous stack) is restored.

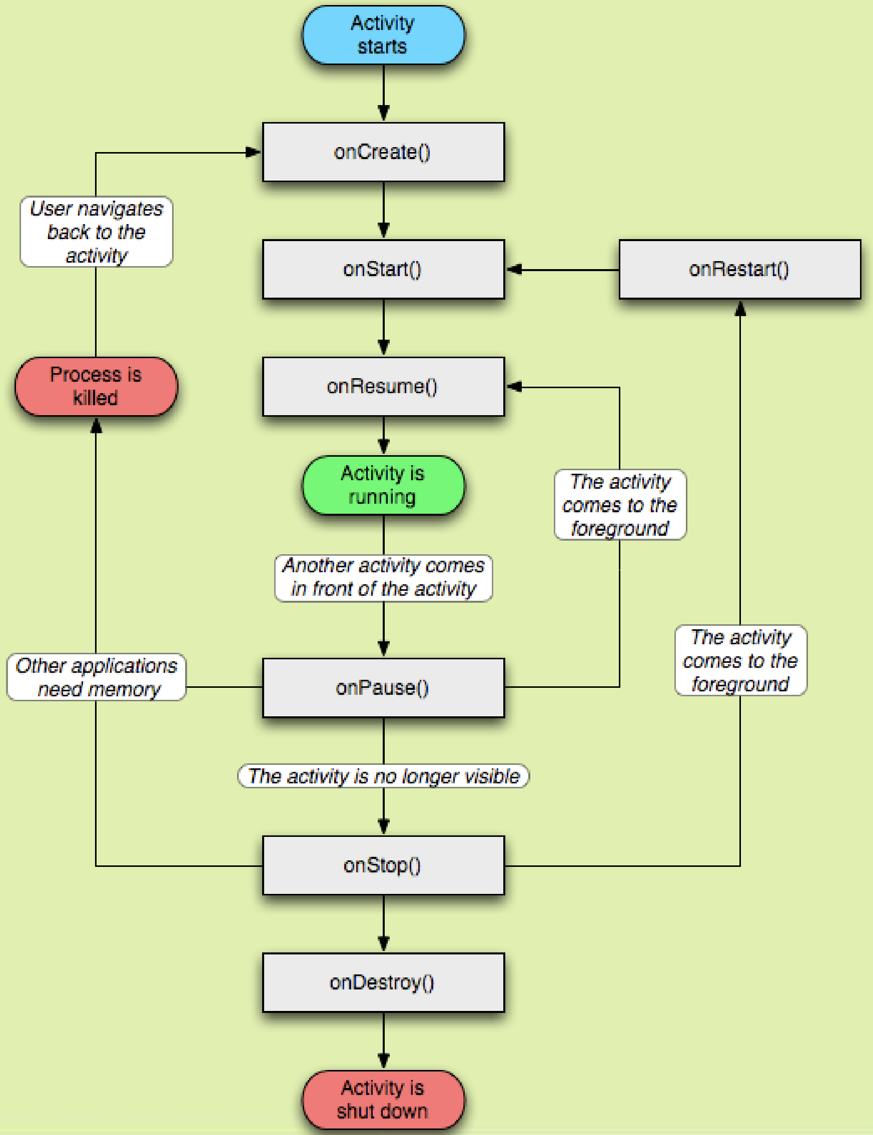
Activity Lifecycle

- All startActivity cares about is making sure the Activity is launched.
  - If it's already launched, just brings it to the front.
  - How does an Activity get managed?

- Event-driven model:
  - Activity has functions to handle each event:
    - onCreate, onResume, onPause, etc.
  - Can take default behavior, or override.
  - All Activities must override onCreate() to do anything.
  - Any overridden method must call through to superclass method.

Activity Lifecycle

From http://developer.android.com/reference/android/app/Activity.html:


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
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Activity Lifecycle

- Three states:
  - Active = In the foreground, running.
  - Paused = Still visible, but obscured by another Activity which takes up part of the screen (or is transparent).
    - Same as active, but may be killed if running very low on memory.
  - Stopped = Not visible on the screen.

Activity Lifecycle

- onCreate()
  - Called when Activity is first created
  - Prepare GUI, other initialization steps.
- onResume()
  - Called when Activity is now on top of the stack.
  - Update GUI values
  - Note: This is called when Activity is first launched!
- onPause()
  - Activity is about to disappear.
  - Commit unsaved data, stop any CPU-intensive tasks.

Service Lifecycle

- Two types:
  - Do some background work on request:
    - Just call startService() – much like an Activity
    - Service has onStartCommand() [or onStart()] to handle this.
    - Service will keep running even after command is executed: best practice to stop with stopSelf(startId) at end.
  - Ongoing communication
    - Example: Music player
    - Use bindService() to make a persistent connection.
    - Client gets an object to make calls to service.
    - We'll discuss this more when we talk about background tasks.

Other Lifecycles

- ContentProviders:
  - ContentResolver cr = Context.getContentResolver();
  - Access through querying content:// URIs
    - Example: cr.query(content://android.provider.Contacts.Phones.CONTACT_URI,…)
  - Have query, insert, delete, etc. methods.
  - More on this in the data storage lecture.

- BroadcastReceivers:
  - Awoken by a system broadcast.
  - Extremely simple – just an onReceive handler.
    - Gets Context and Intent describing broadcast.

Application Resources

- Separate anything that isn't code out of the code
  - Example: Strings, images
  - Make it easy to support different device configurations
    - i.e. different languages, screen sizes

- Common files in res/ directory:
  - drawable/icon.png: Icon for the program in launcher
  - layout/main.xml: User interface for main Activity
  - values/strings.xml: Any Strings appearing in UI

R.java

- Eclipse combs through res/ directory, generating Java class to access resources in code.
- Examples:
  - R.string.<string_name>, R.layout.<layout_name>

- Accessing resources through the R class lets Android determine the right resource to use.

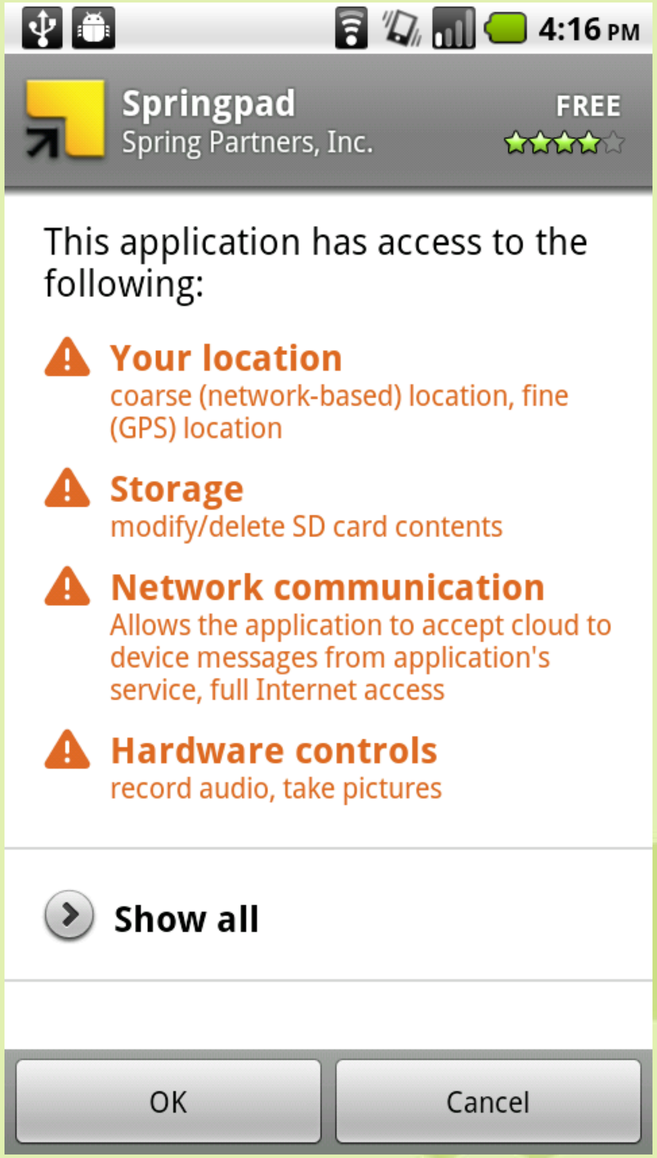
Configuration Changes

- Default Android behavior: If configuration changes, restart Activity.
  - Examples: Pull out hardware keyboard, rotate screen
  - Easy way to ensure correct resources are used.

- Sometimes, this approach takes too long.
  - What if Activity takes a few moments to load data?
  - Can override onConfigurationChanged() to prevent restart and still handle certain common changes.

Android Manifest

- The last piece of an application.
  - XML file
  - The "metadata" about the application and its components.
  - Application:
    - Name, icon, version, required Android version
    - What permissions does this application need?
    - What features does this application need or use?



This application has access to the following:

Your location
coarse (network-based) location, fine (GPS) location

Storage
modify/delete SD card contents

Network communication
Allows the application to accept cloud to device messages from application's service, full Internet access

Hardware controls
record audio, take pictures

Android Manifest

- For each component:
  - Specify intent filters so Android knows the Intents that each component can handle.
- Activity
  - Specify the action “android.intent.action.MAIN” and the category “android.intent.category.LAUNCHER” for default activity – shows up in launcher.
  - Specify other actions that this activity can handle, i.e. the types of files it can view, or URLs it can access.
- BroadcastReceiver
  - Specify the events that this receiver wants to handle.

Intent Matching

- What does it mean for an intent filter to match an intent?
  - Action specified in Intent must match one of the Actions listed in the filter.
    - A filter must list at least one action.
    - An intent with no action matches all filters (with >1 action).
  - Every category in the Intent must match a category in the filter.
    - Filter can have extra categories, not Intent.
  - The data (URI + MIME type) in the Intent must match
    - Matching rules are complex – see http://developer.android.com/guide/topics/intents/intents-filters.html

Multiple Intent Filters

- A component can specify multiple intent filters.
- If any of them match an Intent during resolution, the component may be started.

- Up to the component to figure out what action to take.
  - Activity calls getIntent() to obtain the Intent with which it was started.

- What if no filters are specified?

Android Application

- So – what is an Android application?
- Answer: .apk file
  - Desktop equivalent – the .jar file
- Contains code, resources, anything needed by app.