# Session #4A: Activity Lifecycle and Data

February 26th, 2015

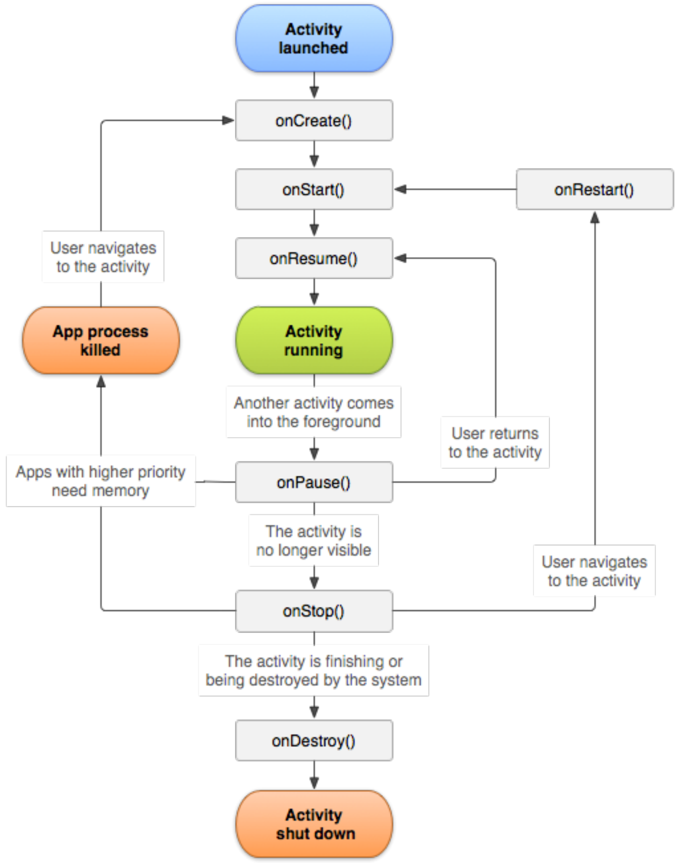
## Overview

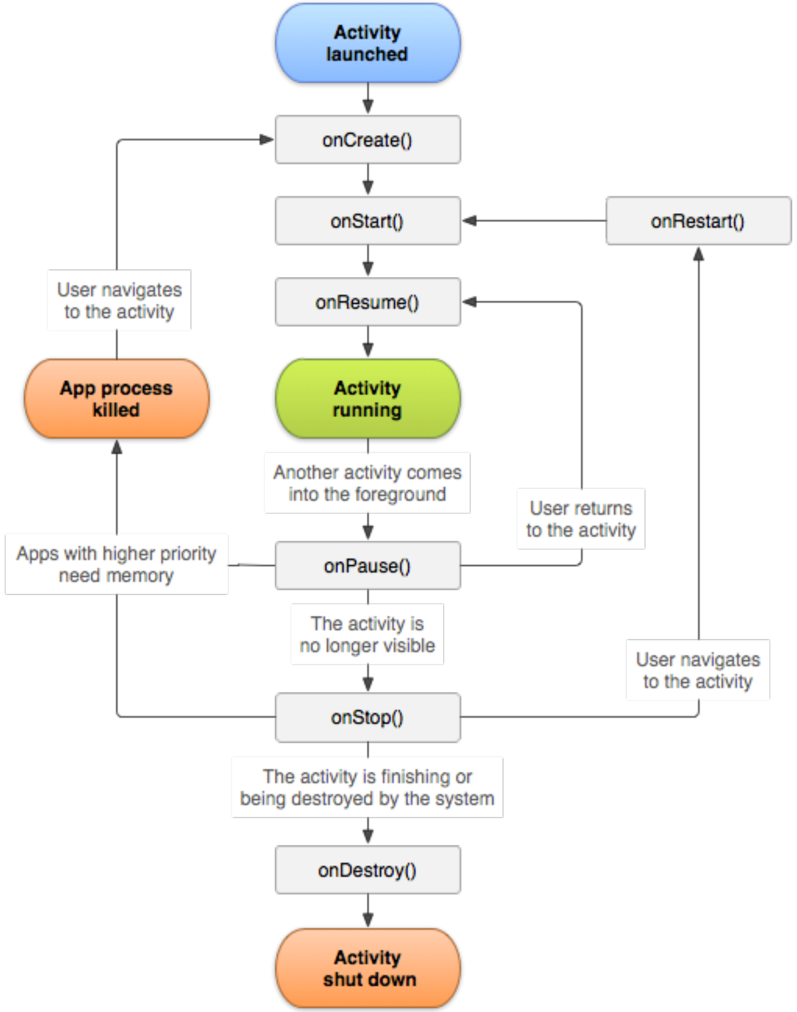
- The Activity Lifecycle
- Data Persistence
- Unit Testing

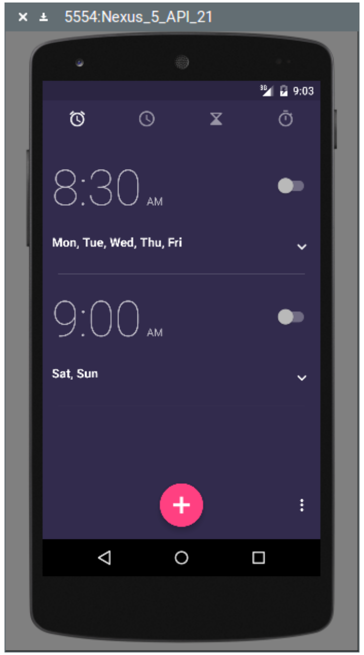
## The Activity Lifecycle

## The Activity Lifecycle

- The Activity back**stack** is a **stack** of recently use activities
- All activities that aren't visible can be destroyed at any time


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

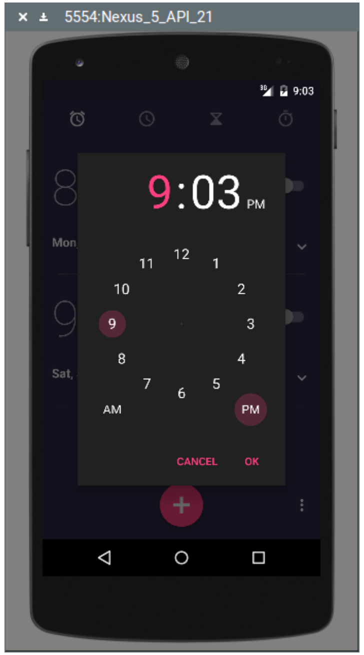

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down


5554:Nexus_5_API_21

9:03

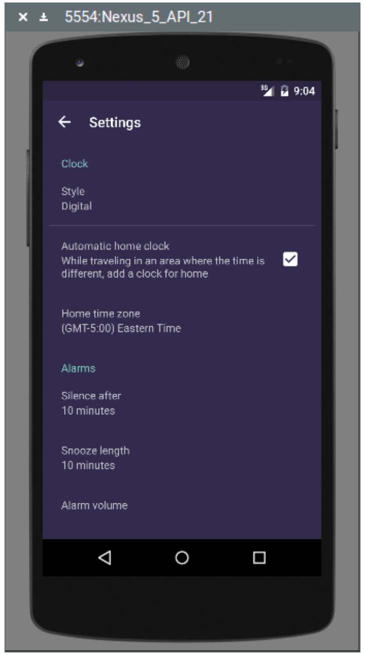
8:30 AM
Mon, Tue, Wed, Thu, Fri

9:00 AM
Sat, Sun


5554:Nexus_5_API_21

9:03

9:03 PM


5554:Nexus_5_API_21

9:04

Settings

Clock

Style
Digital

Automatic home clock ✓
While traveling in an area where the time is different, add a clock for home

Home time zone
(GMT-5:00) Eastern Time

Alarms

Silence after
10 minutes

Snooze length
10 minutes

Alarm volume

$$\begin{tikzcd}
\text{ACTIVE} \arrow[r, bend left, "onPause"] & \text{VISIBLE} \arrow[l, bend left, "onResume"] \arrow[r, bend left, "onStop"] & \text{BACKGROUND} \arrow[l, bend left, "onStart"]
\end{tikzcd}$$

## What happens when my activity is destroyed?

## Preparing for a destroyed activity

Prepare for termination in onStop (pre-Honeycomb onPause)

- save form data
- save listview permissions
- stop updating the user interface
- de-register sensor listeners
- dynamic broadcast receivers

## Saving State

- onSaveInstanceState(Bundle)
    - called after onPause

- onRestoreInstanceState(Bundle)
    - called after onCreate

## Data Persistence

## SharedPreferences

- Values are stored in a file (app private data directory), as key-value pairs

- Values are accessible via SharedPreferences object

## SQLite3

- Useful & retrieving for storing tabular data
- Datatypes
  - NULL, INTEGER, NUMERIC, REAL, TEXT, BLOB
- Query Operations
  - INSERT, SELECT, UPDATE, DELETE, DROP
- Unique & Foreign Keys

## Contracts

- Define an interface between the view and the data(base)
- Useful for defining the table name, schema, and columns of a SQLite3 table

## Contracts Cont’d

1. Create a table entry class that extends BaseColumns (inherits “_id” column)
2. Define constants representing each column in the given table

## SQLiteOpenHelper

Convenience class for managing database creation and versioning

1. Subclass SQLiteOpenHelper
2. Define database filename & version
3. Define CREATE TABLE query
4. Override onCreate() & onUpgrade()

## ContentValues

Represents a database record, very similar to a HashMap

## Reading Databases

1. Get reference to readable database - getReadableDatabase()
2. Execute query - SQLiteDatabase.query(...)
3. Process returned Cursor (ie. extract relevant data from each row)
4. Close your cursor & database*

## Writing Databases (Insert)

1. Get reference to writable database - getWritableDatabase()
2. Execute insert - SQLiteDatabase.insert(...)
3. Process returned rowId (if -1, insert has failed)
4. Close database

## Writing Databases (Update)

1. Get reference to writable database - getWritableDatabase()
2. Execute insert - SQLiteDatabase.update (...)
3. Close database

## Unit Testing

## Android JUnit

- Add JUnit test files to androidTest/ folder
- If you need a reference to a **Context**, your JUnit tests need to subclass **AndroidTestCase**
- Tests prepended by the word “test”
- Optionally implement setUp() and tearDown()

## Lesson #4B Overview

## Lesson #4B Overview

- **ContentProviders**, allow you to share data within your app and with other apps
- Abstracts database / persistence implementation from the application
- Using DataObservers, can automatically update attached views with underlying data changes