Location

Joe Rogers
Android 310

Location

- Most common sensor used in apps.
- Mobile devices have unique characteristic in that they move, and apps are able to provide context based on the user's context.
  - For example, nearby businesses, maps, directions, etc.

Types of sensors

- GPS (Global Positioning Satellite)
  - Most accurate location sensor on device
  - Requires line of sight to satellite so great if outdoors or near a lot of windows (ie car).
  - Almost useless inside a building.
  - Draws most battery especially if requesting updating frequently.

Types of sensors cont.

- Cell Network
  - Uses triangulation to estimate where the user is.
  - Roughly accurate to a city block
  - Drawback is also could be wildly inaccurate in that the phone may see 3 towers, but user is not in “center” of triangle. Especially true with hills.
  - Can be useful inside a building.

Types of sensors cont.

- Wifi
  - Uses the mac address of the router and historical location of where the mac address was seen to provide a location, ie GPS/Cell towers.
  - Great if router is stationary. However, may be very inaccurate if the router moves.
  - Can be useful in a building

Types of sensors cont.

- Bluetooth
  - Stationary beacons reporting lat/long can be useful for approximate location inside a store.
  - Typically application specific code, vs built-in logic.
  - Variances can be seen depending on bluetooth radios, device battery level, etc.
  - Maybe most accurate inside, but again, not an out of the box solution at this time.

Challenges with location

- Choosing which provider meets your app requirements.
- Dealing with “movement”, do you care if the user moves?
- Handling accuracy. Some location updates are not as accurate as others. Have to decide if new location is better.

LocationManager

- Core OS service designed to allow apps to request location updates.
- More “manual” way of requesting location.
  - App must choose GPS/Network provider (or both)
  - App must manually request user to enable location settings via custom dialog, etc.
  - Easier to make mistakes and consume more battery than desired.

FusedLocationProviderApi

- Equivalent variation in Google Play Services
- Wraps the OS Location Manager for ease of use.
  - The location request specifies conditions for returning location to help manage battery, timeliness, accuracy, etc.
  - Utilizes Google's map data to improve accuracy and eliminate the need to know GPS/Network/Wifi, etc.

Geocoder

- Core OS feature (that may or may not be available). Ie If no play store, may not exist.
- Provides ability to translate lat/long coordinates to a physical place
- Also does reverse translation. Ie zip, address, etc to lat/long coordinates.

Foreground Requests

- Usually active when user is on a particular screen in app, or using app.
- Typically implemented via a listener.
- If have large operations when new location received, should provide a looper on request, to place in background thread vs UI thread.

Background Requests

- Want location updates, when user is not running app.
- Typically use a lower frequency
- Provide a pending intent to broadcast receiver to process location when arrives.
  - Could be a service, but you need to ensure the service sets up a wake lock. Wakeful receiver easier.

Code Overview

High level steps

- Request location api
- Determine if location services enabled
  - Handle if not.
- Request last known location
  - Useful if requesting a fresh location so user sees something reasonable as they wait.
  - May be all you need if it meets the “criteria” needed by the app.
  - Typically free, by using requests from other apps.

Code Overview

- Start location requests
- When new location received, verify it is “better”. Especially true when using base Android.
- Process location update
- Turn off location updates (especially if running in foreground).

Emulator

Emulator

- Able to provide “mock” locations provided APP requests GPS locations.
- Drawback is if app only needs network, need to enable FINE location updates to use.
- If trying to use OS apis, make sure you do not use a Google Services image. The location updates are always “old” if you do.

Android Device Monitor

- Able to inject mock locations via “emulator” controls.
- Drawback is you are not able to “debug” the app at the same time which can defeat the purpose especially if you are having an issues handling/receiving the location update.

Telnet and geo command

- Use telnet to emulator and inject geo fix commands.
- Able to use debugger at same time.
  - telnet localhost <emulator port>
  - Issue geo command
    - geo fix <long> <lat> [<elevation>]
    - geo fix -122.3320769 47.5833386

Resources

Resources

- Location Strategies (good for OS api)
- Geo emulator command
- Google Play Services Location