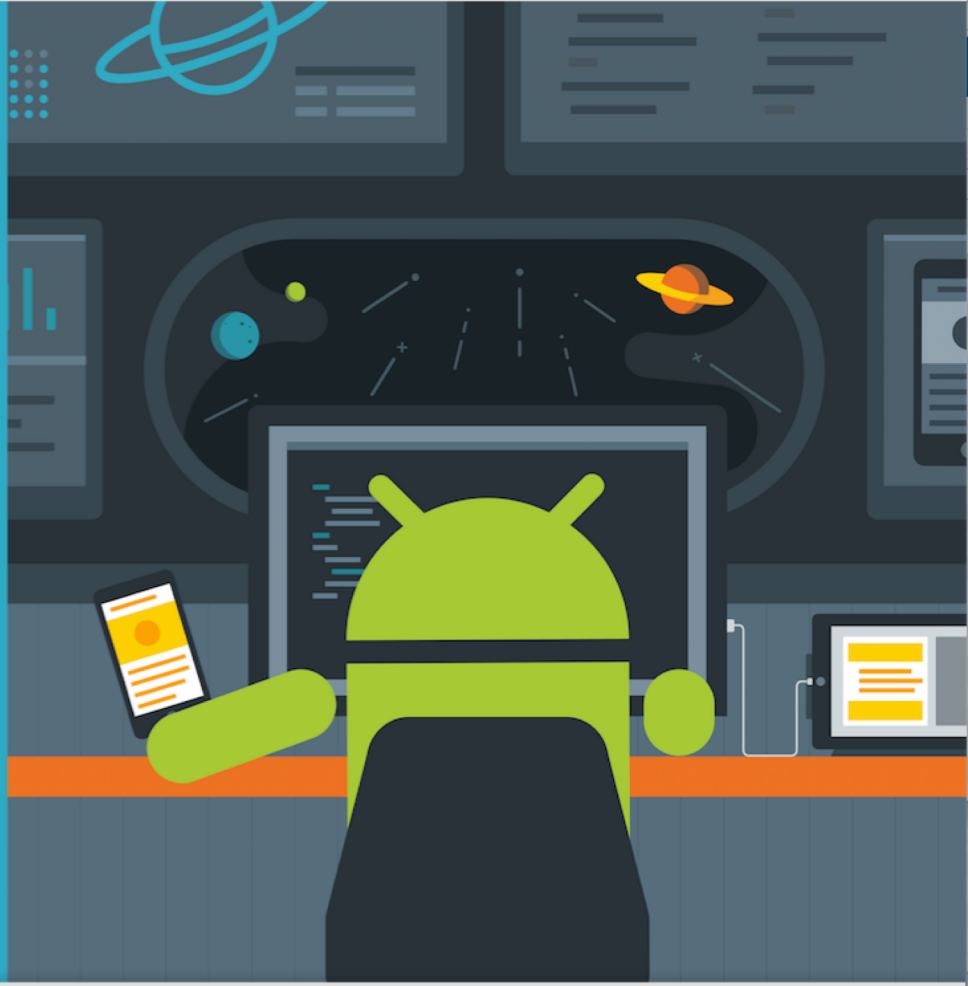
Advanced Android Development

Location

Lesson 7



7.1 Location services

Using physical location in an app

Contents

- Overview
- Setting up Google Play services
- Location permissions
- Get device location
- Geocoding and reverse geocoding
- Creating a `LocationRequest` object
- Working with user settings
- Requesting location updates

Overview

Introduction

- Mobile phones – key word is *MOBILE*
- Users move around and go places
- Your app can detect and use the device location to customize the experience for the user

Using location in your app

- Check if location permission has been granted
- Ask for permission if necessary
- Request most recent location
- Request location updates

Get the device location

- Use `FusedLocationProviderClient`
- Makes location requests combining GPS, Wi-Fi, and cell network
- Balances fast, accurate results with minimal battery drain
- Returns `Location` object with latitude and longitude

Get the device location

- Use `Geocoder` to convert lat/long location to physical address

Latitude 51.5
Longitude -0.078

Tower Bridge,
Tower Bridge Rd,
London SE1 2UP
UK

Setting up Google Play services

Setting up Google Play services

Location services are provided by Google Play Services

Install Google Repository in Android Studio

1. Select **Tools > Android > SDK Manager**
2. Select the **SDK Tools** tab
3. Expand **Support Repository**
4. Select **Google Repository** and click **OK**

Adding Google Play to your project

Add to `dependencies` in `build.gradle (Module: app)`:

```
compile 'com.google.android.gms:play-services:xx.x.x'
```

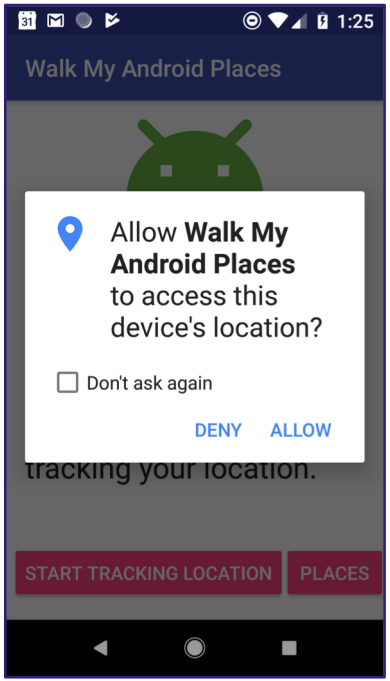
- *xx.x.x* is version number, such as 11.0.2.
- Replace with new version number, if Android Studio suggests it

Location permissions

Users choose to share their location

From Marshmallow onwards:

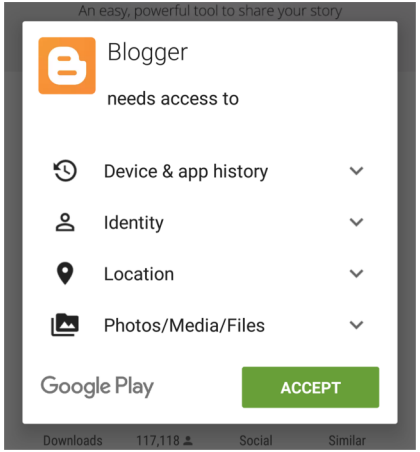
- Users grant or deny access to their location for each app
- Users can change access permission at any time
- Your app can prompt user to grant permission to use location



Users choose to share their location

For apps created before Marshmallow:

- users grant permission before installing
- after installation, user cannot change access permission
- your app can check if permission has been granted



Requesting location permission

Apps must request location permission

- `ACCESS_COARSE_LOCATION`
  for location accurate to within a city block
- `ACCESS_FINE_LOCATION`
  for get precise location

Request permission in manifest

```
<uses-permission
    android:name=
        "android.permission.ACCESS_FINE_LOCATION"
/>
```

Requesting permission at run time

- User can revoke permission at any time
- Check for permission each time your app uses location
- Details and examples: Requesting Permissions at Runtime

Steps to check/request permission

1. Use `checkSelfPermission()` to see if permission granted
2. Use `requestPermissions()` to request permission
3. Check user response to see if request was granted

Check/request permission

```
if (ActivityCompat.checkSelfPermission(this,
          Manifest.permission.ACCESS_FINE_LOCATION) !=
                    PackageManager.PERMISSION_GRANTED) {
   ActivityCompat.requestPermissions(this,
      new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
      REQUEST_LOCATION_PERMISSION);
} else {
   Log.d(TAG, "getLocation: permissions granted");
}
```

Get user's response

Override `onRequestPermissionsResult()` to check if user granted permission

```
@Override
public void onRequestPermissionsResult(int requestCode,
          String permissions[], int[] grantResults) {
              switch (requestCode) {
                  case REQUEST_LOCATION_PERMISSION: {
                      // Check if the permission is granted.
```

Check if request was granted

- Response is returned in `permissions` array
- Compare `grantResults` parameter to `PackageManager.PERMISSION_GRANTED`

```
// Check if the permission is granted.
if (grantResults.length > 0) && grantResults[0] ==
                    PackageManager.PERMISSION_GRANTED) {
        // Permission was granted.
        ...
    } else {// Permission was denied...
```

Get device location

FusedLocationProviderClient

- Use `FusedLocationProviderClient` to request last known location
- Usually, last known location is same as current location

Get FusedLocationProviderClient

- To get `FusedLocationProviderClient`:

```
FusedLocationProviderClient flpClient =
 LocationServices.getFusedLocationProviderClient(
    context);
```

Requesting last known location

- Call `FusedLocationProviderClient` `getLastLocation()`
  - Returns `Task` object representing async task to fetch `Location` object
  - `Task` supplies methods for adding success and failure listeners
- Retrieve latitude and longitude from `Location` object

getLastLocation() success listener

```
mFusedLocationClient.getLastLocation().addOnSuccessListener(
        new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                if (location != null) {
                    mLastLocation = location;
                    // Get the lat and long.
                } else { // Show "no location" }
        }
});
```

getLastLocation() failure listener

```
mFusedLocationClient.getLastLocation().addOnFailureListener(
        new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
               Log.e(TAG, "onFailure: ", e.printStackTrace());
            }
        }
);
```

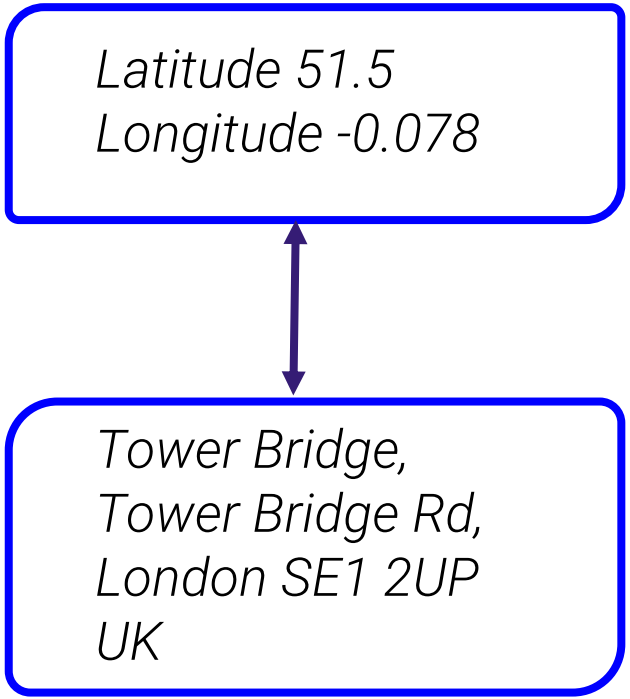
Get latitude and longitude

```
public void onSuccess(Location location) {
   if (location != null) {
      // Get the lat and long.
      lat = location.getLatitude(),
      long = location.getLongitude(),
      time = location.getTime()));
    } else { // no location}
```

Geocoding and reverse geocoding

Geocoding and reverse geocoding

- *Geocode:*
  Convert human-readable street address into latitude/longitude
- *Reverse geocode:*
  Convert lat/long into human-readable street address



Use the Geocoder class

- Use `Geocoder` for geocoding and reverse geocoding

```
Geocoder geocoder = new Geocoder(context,
        Locale.getDefault());
```

- Methods make network request—don't call on main thread

Geocoder backend service

- `Geocoder` requires backend service that are not included in core Android framework
- Use `isPresent()` to check if implementation exists
- `Geocoder` query methods return empty list if no backend service exists

Reverse geocoding coordinates

```
getFromLocation(
    double latitude, double longitude, int maxResults)
```

Returns list of `Address` objects:

```
List<Address> addresses = geocoder.getFromLocation(
    location.getLatitude(), location.getLongitude(), 1);
```

Geocoding address into coordinates

`getFromLocationName(`

`    String locationName, int maxResults)`

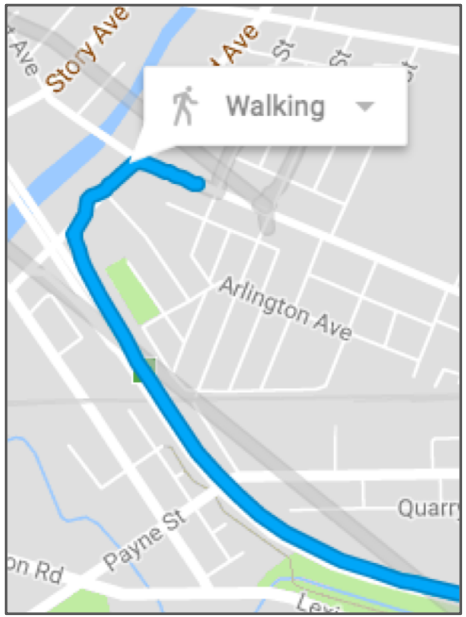
Returns list of `Address` objects with latitude/longitude coordinates:

```
List<Address> addresses = geocoder.getFromLocationName(
    "731 Market St, San Francisco, CA 94103", 1)
Address firstAddress = addresses.get(0);
double latitude = firstAddress.getLatitude();
double longitude = firstAddress.getLongitude();
```

Creating a LocationRequest object

Getting location updates

- Your app can get the last known location.
- It can also ask for regular updates to track location
- Use `LocationRequest` to set parameters for location update requests



LocationRequest parameters

Set `LocationRequest` parameters to control location requests

- `setInterval()`:
  Sets how frequently your app needs updates
- `setFastestInterval()`:
  Sets limit to the update rate to prevent flicker/data overflow
- `setPriority()`:
  Sets request priority and sources

Request priority values

PRIORITY_BALANCED_POWER_ACCURACY	Precise to within city block (100 meters); uses only Wi-Fi and cell network, to consume less power
PRIORITY_HIGH_ACCURACY	Uses GPS if available
PRIORITY_LOW_POWER	City-level precision (10 km)
PRIORITY_NO_POWER	Updates when triggered by other apps (zero additional power)

Create LocationRequest example

```
private LocationRequest getLocationRequest() {
    LocationRequest locationRequest = new LocationRequest();
    locationRequest.setInterval(10000);
    locationRequest.setFastestInterval(5000);
    locationRequest.setPriority(
        LocationRequest.PRIORITY_HIGH_ACCURACY);
   return locationRequest;
}
```

Requesting location updates

Requesting location updates

- Use `LocationRequest` with `FusedLocationProviderClient`
- Accuracy of location determined by:
  - Available location providers (network and GPS)
  - Location permission requested
  - Options set in location request

Steps to start location updates

1. Create `LocationRequest` object
2. Override `LocationCallback.onLocationResult()`
3. Use `requestLocationUpdates()` on `FusedLocationProviderClient` to start regular updates

Steps to start location updates

Use `requestLocationUpdates()` to start regular updates

- Pass in `LocationRequest` and `LocationCallback`
- Location updates are delivered to `onLocationResult()`

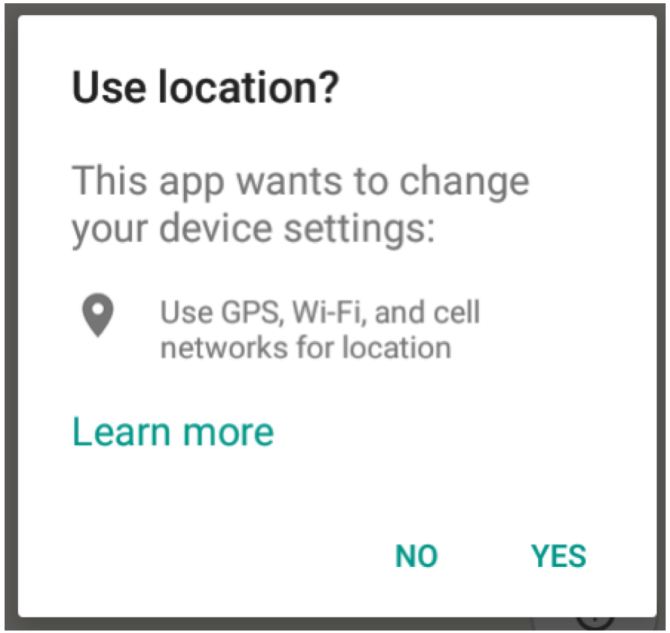
Using LocationCallback

```
mLocationCallback = new LocationCallback() {
  @Override
  public void onLocationResult(
                              LocationResult locationResult) {
    for (Location location : locationResult.getLocations()) {
        // Update UI with location data
        // ...
    }
};
```

Working with user settings

User settings for location services

- Users can control balance between accuracy and power consumption
- Your app can:
  - Detect device settings
  - Prompt user to change them



Steps to check device settings

1. Create a `LocationSettingsRequest` object
2. Create a `SettingsClient` object
3. Use `checkLocationSettings()` to see if device settings match `LocationRequest`
4. Use `OnFailureListener` to catch when settings don't match `LocationRequest`

1. Create a LocationSettingsRequest

Create a `LocationSettingsRequest` object and add one or more location requests:

```
LocationSettingsRequest settingsRequest =
    new LocationSettingsRequest.Builder()

.addLocationRequest(mLocationRequest).build();
```

2. Create a SettingsClient

Create a `SettingsClient` object using `LocationServices.getSettingsClient()` and pass in the context:

```
SettingsClient client =
        LocationServices.getSettingsClient(this);
```

3. See if device settings match request

- Use `checkLocationSettings()` to see if device settings match `LocationRequest`:

```
Task<LocationSettingsResponse> task =
    client.checkLocationSettings(settingsRequest);
```

- Returns a `Task` object

4. Catch when settings don't match

- Use `OnFailureListener` to catch when settings don't match `LocationRequest`
- Exception passed into `onFailure()` contains `Status`
- If settings don't match, then status is `LocationSettingsStatusCodes.RESOLUTION_REQUIRED`
- Show dialog to change settings

Sample code for user dialog

```
task.addOnFailureListener(this, new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        int statusCode = ((ApiException) e).getStatusCode();
        if (statusCode ==
              LocationSettingsStatusCodes.RESOLUTION_REQUIRED) {
            // Show the user a dialog.
            // ...
        }
    }
});
```

Sample code for user dialog (continued)

```
// Show the user a dialog.
try {
     // Call startResolutionForResult(), check
     // result in onActivityResult()
     ResolvableApiException resolvable =
                                  (ResolvableApiException) e;
     resolvable.startResolutionForResult
               (MainActivity.this,REQUEST_CHECK_SETTINGS);
     } catch (IntentSender.SendIntentException sendEx) {
               // Ignore the error
     }
```

Handle the user's decision

- Override `onActivityResult()` in `Activity`
- Ensure `requestCode` matches constant in `startResolutionForResult()`

Sample onActivityResult method

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
   if (requestCode == REQUEST_CHECK_SETTINGS) {
      if (resultCode == RESULT_CANCELED) {
         stopTrackingLocation();
       } else if (resultCode == RESULT_OK) {
         startTrackingLocation();
       }
   }
   super.onActivityResult(requestCode, resultCode, data);
}
```

What's next?

- Concept chapter: 7.1 Location services
- Practical: 7.1 Using the device location

END