Android advanced

Jaroslav Dytrych

Faculty of Information Technology Brno University of Technology
Božetěchova 1/2. 612 66 Brno - Královo Pole
dytrych@fit.vutbr.cz

16 December 2020

Content

- Interacting with other Apps
- Content sharing
- Background jobs
- Google Maps and Location API
- Google Play Services
- Location API
- Maps API

- Explicit intents are only for remote services.
- Otherwise implicit intents must be used.
- Simple intents
  - Make a call.
  - View a webpage or a map.
  - ...
- Intents with extras
  - Compose an email.
  - Create calendar event.
  - ...

- Verify your intent can be handled

```
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities =
packageManager.queryIntentActivities(intent, 0);
boolean isIntentSafe = activities.size() > 0;
```

- Default chooser appears, when multiple apps can handle your intent.
- Handle result of an App
  - `startActivityForResult(intent)`
  - Override `OnActivityResult()` callback.
  - Resolve which result it is by `REQUEST_ID`.
  - Query content resolver.

- Allow other apps to start your activity.
  - Utilize intent filter.
    - Action
    - Data (mime type)
    - Category
  - Handle request in `onCreate` method.
  - Return back results with `setResult` function.
  - Call `finish()` on your activity.

Example CalledActivity

Content sharing

- Data can be sent to other Apps via extras
  - `EXTRA_TEXT`
  - `EXTRA_STREAM` – URI (e.g. to image data)
- Receiving data
  - Intent filters `ACTION_SEND(_MULTIPLE)`
  - Mime type must be defined.
- File sharing
  - Declare file provider in manifest.
  - Specify sharing paths.
  - Accessible via

    ```
    content://name.of.package.fileprovider/sharingpath/
    default_image.jpg
    ```

Content sharing

- File provider must specify intent filter.
  - `ACTION_PICK`
  - Category `OPENABLE`
- Requesting App gets data via URI.
- Usually more Apps can supply files.
  - Explicit App can be called directly.
  - Avoid chooser by Intent's `setComponentName(package, full class name)`

Examples FileProvider, FileRequest

Android drawing

- Simple
  - Class must extend `View`
  - `onDraw()` – repaints whole view
  - `invalidate()`
- SurfaceView
  - `onDraw()` – called manually, draws on holder's canvas.
  - Drawing performed via thread.
- Advanced graphics
  - OpenGL ES
  - `GLSurfaceView`

Examples StartDraw, HelloOpenGLES

- Several possibilities in Android
  - Services
    - Local
    - External
  - `ThreadPoolExecutor`
    - Queue of Runnables
    - Method `poll()` for obtaining `Runnable` resource
    - `execute(Runnable)` to start the task in background
  - `AsyncTask`

- `AsyncTask<Params,Progress,Result>`
- `Params` – List of "settings" objects, telling AsyncTask what to do.
- `Progress` – can be returned via `publishProgress(Progress)`
- `Result` – return type of result.
- 4 steps of processing

- `AsyncTask` – continuation
  - `OnPreExecute`
    - invoked on the UI thread
    - initialization
  - `Result doInBackground(Progress ...)`
    - after `onPreExecute` finishes processing
  - `void onProgressUpdate(Progress)`
    - invoked by `publishResults()`
    - on UI thread – can update views
- Can be cancelled
  - Method `cancel(boolean mayInterruptIfRunning)`
  - `OnCancelled` called instead of `onPostExecute`

- Must use Google Play Services.
- Can be referenced as LibraryProject

```
<meta-data android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version" />
```

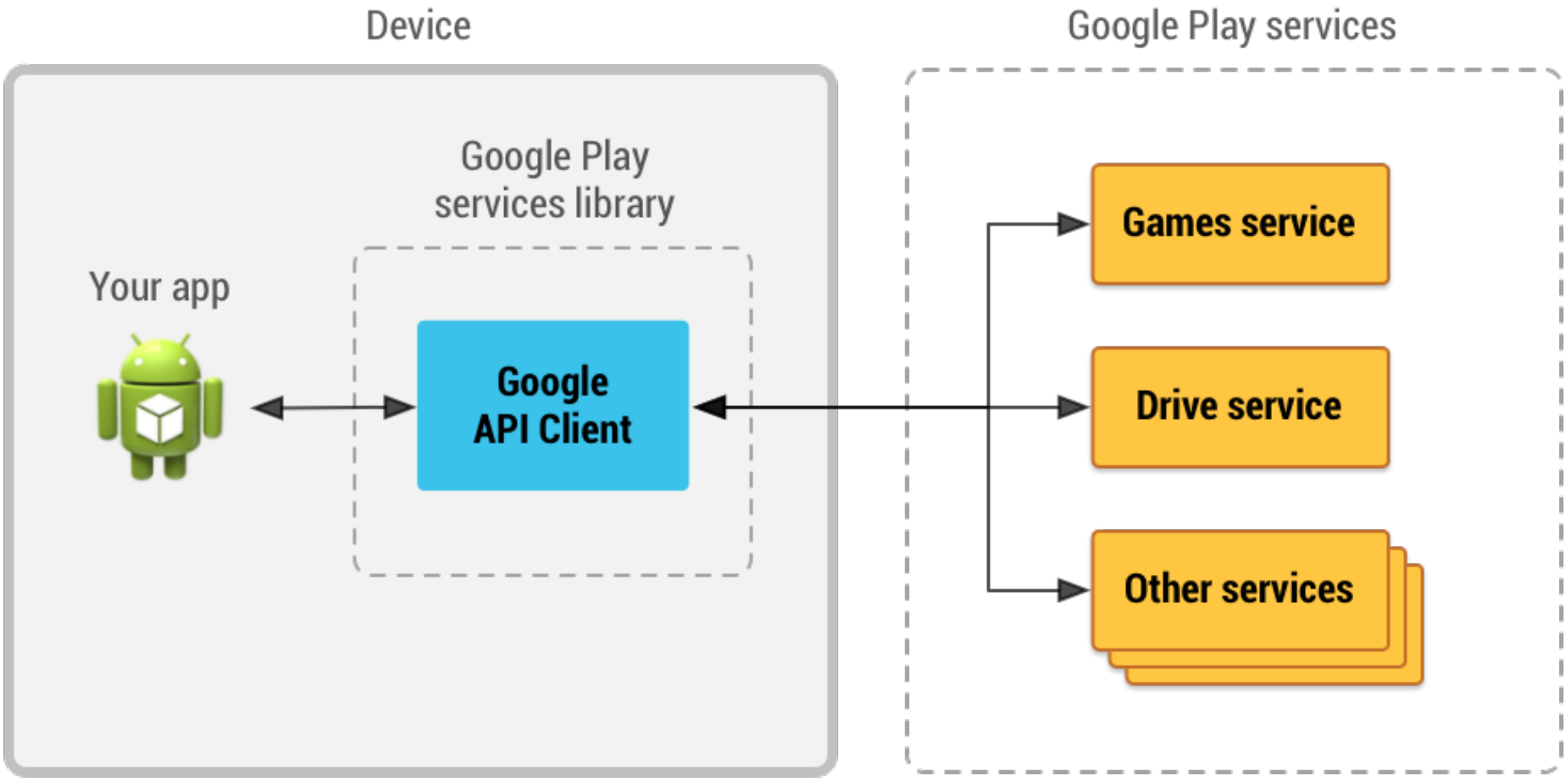
- Usually present on nowadays android devices, but programmer should check.
- Needs extra libraries for emulator.
  - `com.android.vending.apk`
  - `com.google.android.gms.apk`
- Some emulators have them preinstalled.

- Google Play Services runs on device with Android 2.3 and higher.
  - Emulator 4.2.2 and higher

    ```
    GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
    ```
  - Interfaces must be implemented

    `GooglePlayServicesClient.ConnectionCallbacks`,

    `GooglePlayServicesClient.OnConnectionFailedListener`
  - Respective callbacks
    - `OnConnected`
    - `OnConnectionFailed`
    - `OnDisconnected`
    - `onConnectionSuspended`



Asynchronous communication

Location API

- Location API must declare permission
  - `ACCESS_COARSE_LOCATION`
  - `ACCESS_FINE_LOCATION`
- Class `GoogleApiClient` for interaction
  - proxy object
- `LocationServices.FusedLocationProviderApi`
  - entry point for interacting with the fused location provider.
- Current location can be obtained via client

```
LocationServices.FusedLocationApi.getLastLocation(GAClient);
```

- Application can handle location updates.

- Location updates
  - Programmer must form `LocationRequest` object.
- Accuracy
- Update interval

```
locationClient.requestLocationUpdates(locationRequest, Context);
```

- Location update callback

```
@Override
 public void onLocationChanged(Location location) {
   // Report to the UI that the location was updated
   String msg = "Updated Location: " +
       Double.toString(location.getLatitude()) + "," +
       Double.toString(location.getLongitude());
   Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
 }
```

- Can be used to convert location to address.
- Address computation can take some time.
  - It can't be done in UI thread.
- Addresses can be obtained from Geocoder object.
  - Address object can be used to get
    - State
    - Administrative unit
    - Locality (usually city)
    - Sub-locality
    - Street
    - Address line
    - Phone (if known)
    - Postal code

Example AndroidLocation

- Geofences
  - points of interest
  - user location combined with nearby features
  - Geofence is rather an area.
- Geofence consists of
  - longitude, latitude, radius,
  - expiration time, Geofence ID, Transition Type.
- Geofence storage
  - holds defined geofences.
- Intent can be defined to handle transitions
  - `OnAddGeofencesResultListener`
  - `OnHandleIntent` – programmer can make updates in the App based on transition type.
- Geofence monitoring can be turned off.

- Recognizing user current activity
  - On foot
  - Tilting
  - In vehicle
  - Riding a bike
  - Running
  - Walking
- Permission `ACTIVITY_RECOGNITION`
  - Requires `ACCESS_FINE_LOCATION`
- Registered `ActivityRecognitionClient` makes programmer-defined `IntentService` receive updates.
- Detected activity has method `describeContents`
  - can return confidence.

- Programmer must get API key.
  - Accessible in Google API console

    `https://console.developers.google.com/apis/`
  - Registered App must enable Maps API.
    - Key can be generated for WebApp, Android, iOS device and server.

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="your api key"/>
```

- Defined in manifest
  - Key can be generated in combination with SHA1 hash from local keystore (can be created within IDE when exporting apk).

- Debug key
  - Key must be named "androiddebugkey".
  - Password both to keystore and key must be "android".
- Release key
  - Private key must be generated

```
keytool -genkey -v -keystore my-release-key.keystore
-alias alias_name -keyalg RSA -keysize 2048
-validity 10000
```

  - Then can be application compiled.
  - At the end – APK must be signed with the key

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1
-keystore my-release-key.keystore my_application.apk
    alias_name
```

- Using predefined map fragments

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
  <fragment
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:tag="maps"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    />
</FrameLayout>
```

- `MapFragment` supported in API >=12
  - `SupportMapFragment` for older versions.
- `MapView` – for embedding as View.

- Map types
  - Normal – typical road map, important natural features such as rivers are shown.
  - Hybrid – satellite photograph data with road maps.
  - Satellite – raw satellite photograph data.
  - Terrain – topographic data. The map includes colors, contour lines and labels, and perspective shading.
  - None – the map will be rendered as an empty grid with no tiles loaded.
- Indoor maps – floor plans can be added to Google maps directly, it will be visible for all users.

Example AndroidMaps

- Map fragments can be added dynamically

```
mMapFragment = MapFragment.newInstance();
FragmentTransaction fragmentTransaction =
    getFragmentManager().beginTransaction();
fragmentTransaction.add(R.id.my_container, mMapFragment);
fragmentTransaction.commit();
```

- Check map availability

```
mMap = ((MapFragment)
    getFragmentManager().findFragmentById(R.id.map))
        .getMap();
    // Check if we were successful in obtaining the map.
    if (mMap != null) {
        // The Map is verified. It is now safe to
        // manipulate the map.

    }
```

- Map state
  - Camera position
    - Location
    - Zoom
    - Bearing
    - Tilt
  - Map type
  - Controls and gestures
- Can be defined in xml layout file or programatically.

- State configuration on XML layout file

```
xmlns:map="http://schemas.android.com/apk/res-auto"
```

- Configuration via map namespace

```
map:cameraBearing="112.5"
map:cameraTargetLat="-33.796923"
map:cameraTargetLng="150.922433"
map:cameraTilt="30"
map:cameraZoom="13"
map:mapType="normal"
```

- Programatic configuration

```
GoogleMapOptions options = new GoogleMapOptions();
options.mapType(GoogleMap.MAP_TYPE_SATELLITE)
    .compassEnabled(false)
    .rotateGesturesEnabled(false)
    .tiltGesturesEnabled(false);
```

- Drawing on the map
  - Markers

```
mMap = ((MapFragment)
  getFragmentManager().findFragmentById(R.id.map)).getMap();
mMap.addMarker(new MarkerOptions()
    .position(new LatLng(10, 10))
    .title("Hello world"));
```

- Properties like `title`, `position`, `alpha`, `draggable`, `icon`, `snippet`, `visible`, `location`, `flag`, `color`, `image`, `rotation`.
- Markers can be animated
  - Info windows
    - Only one displayed at the time.
    - Method of a marker.
  - Overlays
    - Ground – drawing image on the map.
    - Tile – grid with coordinates and zoom level.

- Drawing shapes
  - Polyline
  - Polygon
    - Shapes are autocompleted.
  - Circles
  - Z-index may be specified.
- StreetView

```
<fragment
 android:id="@+id/streetviewpanorama"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 class="com.google.android.gms.maps.StreetViewPanoramaFragment"/>
```

- Enable/disable user navigation.

- Interacting with a map
  - UI controls
    - Zoom controls
    - Compass
    - My Location button
    - Level picker
  - Map gestures
    - Zoom
    - Scroll
    - Tilt
    - Rotate
  - Events
    - Click/Long Click
    - Camera change

References

- https://developer.android.com/training/index.html
- https://cloud.google.com/maps-platform/

Thank you for your attention!