How to use ‘Google Fit’ in Android!

Google Fit Link: *https://fit.google.com*

How to use

Google Fit is available on Android devices with Google Play services 7.0 or higher. So Google play store is required for running Google Fit App in android mobile.

Activate the Fitness API

- Go to the Google Developers Console and make sure you have a Google account which is of course self explanatory.
- In the left sidebar, click APIs and authorize.
- Find the Fitness API and set its status to ‘ON’
- Import Google play services library to your project

Obtain the SHA1 fingerprint of your certificate

- Find the location of your keystore.

```
$ keytool -exportcert -alias \androiddebugkey -keystore \~/.android/debug.keystore -list -v
```

- The output is SHA1 fingerprint

Create a new client ID

- Go to the Google Developers Console.
- In the left sidebar, click Credentials.
- Click Create a new Client ID. The Create Client ID dialog box appears.
- Under Application Type, select Installed application.
- Under Installed Application Type, select Android.

- In the Package Name field, enter the package name of your Android app
- In the Signing Certificate Fingerprint (SHA1) field, enter the SHA1 fingerprint of your certificate.
- Click Create Client ID.

Configure your project

- Import google play services library to your project

Connect to the fitness service

- Add <meta-data android:name="com.google.android.gms.version"

android:value="@integer/google_play_services_version" /> to your manifest file.

- Define variables in your activity to help you track the connection status:

```
private static final int REQUEST_OAUTH = 1;

private static final String AUTH_PENDING = "auth_state_pending";
private boolean authInProgress = false;
private GoogleApiClient mClient = null;
```

- Determine whether the authentication is in progress in the onCreate method:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  // Put application specific code here.
  if (savedInstanceState != null) {
     authInProgress = savedInstanceState.getBoolean(AUTH_PENDING);
  }
  buildFitnessClient();
}
```

- Create the Google API client and provide the required callback methods:

```
Private void buildFitnessClient() {
   // Create the Google API Client
   mClient = new GoogleApiClient.Builder(this)
         .addApi(Fitness.SENSORS_API)
         .addScope(new Scope(Scopes.FITNESS_LOCATION_READ))
         .addConnectionCallbacks(
               new GoogleApiClient.ConnectionCallbacks() {
                  @Override
                  public void onConnected(Bundle bundle) {
                     Log.i(TAG, "Connected!!!");
                     // Now you can make calls to the Fitness APIs.
                     // Put application specific code here.
                  }
@Override
 public void onConnectionSuspended(int i) {
                     // If your connection to the sensor gets lost at some point,
                     // you'll be able to determine the reason and react to it here.
                     if (i == ConnectionCallbacks.CAUSE_NETWORK_LOST) {
                        Log.i(TAG, "Connection lost.  Cause: Network Lost.");
                     } else if (i == ConnectionCallbacks.CAUSE_SERVICE_DISCONNECTED) {
                        Log.i(TAG, "Connection lost.  Reason: Service Disconnected");
                     }
                  }
               }
         )
         .addOnConnectionFailedListener(
               new GoogleApiClient.OnConnectionFailedListener() {
                  // Called whenever the API client fails to connect.
                  @Override
                  public void onConnectionFailed(ConnectionResult result) {
                     Log.i(TAG, "Connection failed. Cause: " + result.toString());
                     if (!result.hasResolution()) {
                        // Show the localized error dialog
                        GooglePlayServicesUtil.getErrorDialog(result.getErrorCode(),
                              MainActivity.this, 0).show();
                        return;
```

```
                    }
                    // The failure has a resolution. Resolve it.
                    // Called typically when the app is not yet authorized, and an
                    // authorization dialog is displayed to the user.
                    if (!authInProgress) {
                        try {
                            Log.i(TAG, "Attempting to resolve failed connection");
                            authInProgress = true;
                            result.startResolutionForResult(MainActivity.this,
                                    REQUEST_OAUTH);
                        } catch (IntentSender.SendIntentException e) {
                            Log.e(TAG,
                                    "Exception while starting resolution activity", e);
                        }
                    }
                }
            }
        )
        .build();
}
```

- Manage the connection lifecycle of your client inside your activity:

```
@Override
protected void onStart() {
    super.onStart();
    // Connect to the Fitness API
    Log.i(TAG, "Connecting...");
    mClient.connect();
}
@Override
protected void onStop() {
    super.onStop();
    if (mClient.isConnected()) {
        mClient.disconnect();
    }
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   if (requestCode == REQUEST_OAUTH) {
      authInProgress = false;
      if (resultCode == RESULT_OK) {
         // Make sure the app is not already connected or attempting to connect
         if (!mClient.isConnecting() && !mClient.isConnected()) {
            mClient.connect();
         }
      }
   }
}
@Override
protected void onSaveInstanceState(Bundle outState) {
   super.onSaveInstanceState(outState);
   outState.putBoolean(AUTH_PENDING, authInProgress);
}
```

Google Fit is available on Google Play for devices running Android 4.0, Ice Cream Sandwich and above.