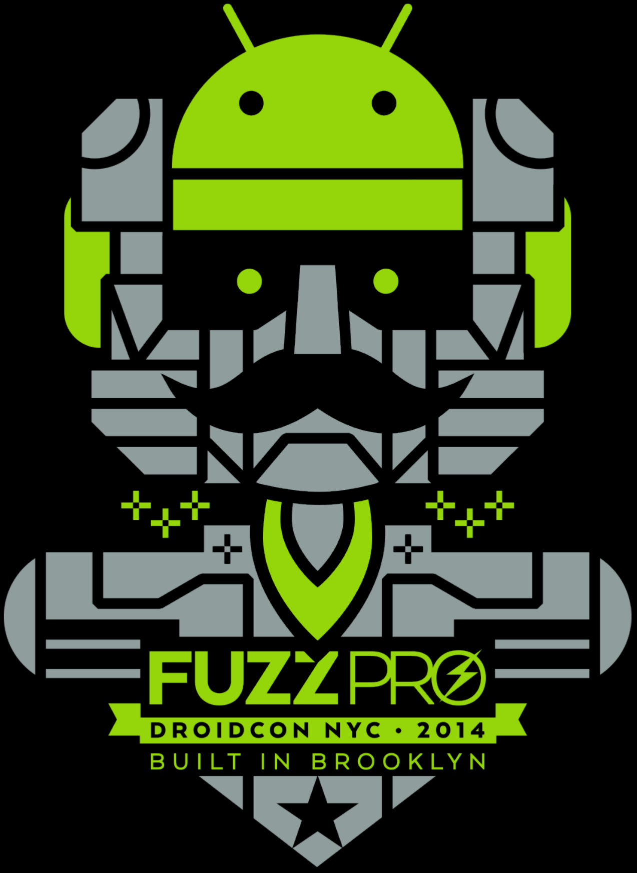
FUZZ

Google Fit
A Primer

INTRO

CESAR AGUILAR

Android Director, FUZZ



We’re Hiring!

fuzz.pro/droidcon

UNDER CONSTRUCTION

WHAT DO YOU GET?

BODY DATA	ACTIVITY DATA	LOCATION DATA
Weight	Calories	Speed
Height	Cadences	Location
Heart Rate	Steps	Distance
	Sample	

GETTING STARTED

https://developers.google.com/fit/preview

- Limited to Nexus 5 and Nexus 7
- Compile against android-L
- Also needs Google Play Services 5.2.08

Like most Google Services other services

- You need an api key from the developer console
  - Not needed for the preview edition

CONNECTING TO FIT

```
mClient = new GoogleApiClient.Builder(this)
            // select the Fitness API
            .addApi(Fitness.API)
                  // specify the scopes of access
            .addScope(FitnessScopes.SCOPE_BODY_READ_WRITE)
                  // provide callbacks
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .build();
      // Connect the Google API client
      mClient.connect();
```

FitActivity.java

OVERVIEW

Universal Data Source

Central Sensor Hub

OVERVIEW

Universal Data Source

Central Sensor Hub

Extensibility

EXTENSIBILITY

Custom Data Types

- Can be private or shareable with other apps
- https://developers.google.com/fit/android/data-types#shareable_data_types

Custom Sensors

- These are “software” sensors

CUSTOM DATA TYPES

```
String BP = “com.fuzz.android.bloodpressure.bp”;
//Check if the data type exist already
PendingResult<DataTypeResult> pendingResult =
            Fitness.HistoryApi.readDataType(mClient, BP);
//Otherwise create it
DataTypeCreateRequest request = new DataTypeCreateRequest.Builder()
            .setName(BP)
            .addField("systolic", DataType.Field.FORMAT_INT32)
            .addField("diastolic", DataType.Field.FORMAT_INT32)
            .addField(DataTypes.Fields.BPM)
            .build();
PendingResult<DataTypeCreateResult> pendingResult =
            Fitness.HistoryApi.addDataType(mClient, request);
```

BPActivity.java

UNIVERSAL DATA SOURCE

Can Read and Write Data

Saved to the Cloud (Fitness Store)

Always up to date

Its not app centric its user centric

DATA ARCHITECTURE

DataSource

DataPoints and DataSets

Sessions and Buckets

HistoryApi and RecordingApi

READING DATA

```
DataReadRequest readreq = new DataReadRequest.Builder()
                .addDefaultDataSource(DataTypes.HEART_RATE_BPM)
                //.bucketByTime(1, TimeUnit.DAYS)
                .setTimeRange(startTime, endTime)
                .build();
PendingResult<DataReadResult> pendingResult =
                Fitness.HistoryApi.readData(mClient, readreq);
```

Response either has Buckets or DataSets

BPActivity.java

WRITING DATA

Step 1: Create a Data Source

```
DataSource dsApp = new DataSource.Builder()
        .setAppPackageName(this)
        .setDataType(DataTypes.HEART_RATE_BPM)
        .setName("fuzz-bp-recorder")
        .setType(DataSource.TYPE_RAW)
        .build();
```

RecorderActivity.java

WRITING DATA

Step 2: Create your data

```
DataSet dataSet = DataSet.create(dsApp);
DataPoint point = dataSet.createDataPoint()
                .setTimestamp(new Date().getTime(), TimeUnit.MILLISECONDS);
//point.setIntValues(sys,dia,bpm);
//point.setFloatValues(sys,dia,bpm);
point.getValue(dateType.getFields().get(0)).setInt(sys);
point.getValue(dateType.getFields().get(1)).setInt(dia);
point.getValue(DataTypes.Fields.BPM).setFloat(bpm);
dataSet.add(point);
```

RecorderActivity.java

WRITING DATA

Step 2: Save your data

```
DataInsertRequest insreq = new DataInsertRequest.Builder()
                    .setDataSet(dataSet)
                    .build();
PendingResult<Status> pendingResult =
                    Fitness.HistoryApi.insert(mClient, insreq);
```

RecorderActivity.java

WORKING WITH SESSIONS

- Session
- HistoryAPI
    - SessionInsertRequest
    - SessionReadRequest
- RecordingAPI
    - startSession
    - endSession

CENTRAL SENSOR HUB

Connects to any sensors the devices itself offers

- Either Software or Hardware

BLE Support can connect to BLE sensor that support specific data types

- Android Wear support out of the box

SensorsApi and BleApi

Can use the sensor api to update the user in realtime

FINDING A SENSOR

```
DataSourcesRequest request = new DataSourcesRequest.Builder()
        // At least one datatype must be specified.
        .setDataTypes(DataTypes.HEART_RATE_BPM)
        .setDataSourceTypes(DataSource.TYPE_RAW)
        .build();
PendingResult<DataSourcesResult> result =
        Fitness.SensorsApi.findDataSources(mClient, request);
```

RecorderActivity.java:177

START LISTENING

```
SensorRequest request = new SensorRequest.Builder()
                .setDataSource(dataSource) // Optional but recommended
                .setDataType(dataType) // Can't be omitted.
                .setSamplingRate(10, TimeUnit.SECONDS)
                .build();
PendingResult<Status> result =
                Fitness.SensorsApi.register(mClient, request, mListener);
```

RecorderActivity.java:224

CUSTOM SENSORS

Create Service with Intent Filter

```
<intent-filter>
<action android:name="com.google.android.gms.fitness.service.ApplicationSensorService" />
<data android:mimeType="vnd.google.android.fitness.data_type/com.google.heart_rate.bpm" />
</intent-filter>
```

Your Service extends ApplicationSensorService

```
void onCreate();
List<DataSource> findDataSources(List<DataType> dataTypes);
boolean register(ApplicationSensorRequest request);
boolean unregister(DataSource dataSource);
```

BPMSensorService.java and SensorService.java

PUBLISHING DATA

From your Service

```
boolean register(ApplicationSensorRequest request);
DataPoint point;
….
request.getDispatcher().publish(point);
```

BPMSensorService.java and SensorService.java

DEMO

HELPFUL LINKS

https://developers.google.com/fit/preview

https://developers.google.com/fit/android/samples

https://plus.google.com/communities/103314459667402704958

Q&A