LOCATION & MAP

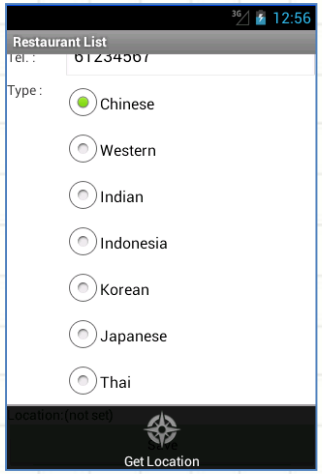
Today's Overview

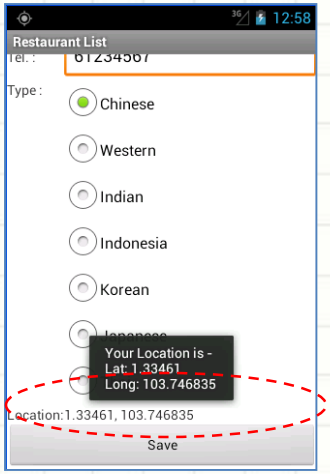
1. Location
2. Map

Location

Location

- In first part of Practical 6 exercise, the Restaurant List app is added with an extra feature to allow user to save the latitude and longitude of a restaurant's location using Android's LocationManager system service





Location

- Let's check what do we need to modify from the previous exercise to save restaurant's GPS coordinates (latitude & longitude)?
  - ❑Model - Any change in Data Model?
  - ❑View - Do you need to modify any of the user interface view?
  - ❑Controller - Do you need to tell the Controller to do any thing new?

Location

- Model – YES

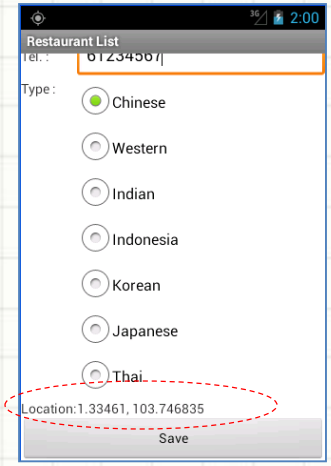
  Extra data elements are introduced to store latitude and longitude of a restaurant's location, and handling methods are added to RestaurantHelper

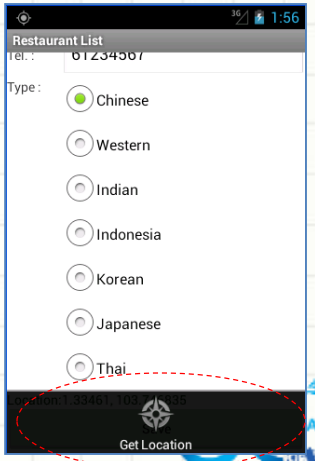
Location

❑View – YES

There will be some changes involved

- ✓ For *detail_form.xml* layout, a *TextView* widget is added to display restaurant's location information
- ✓ *details_option.xml* layout for MENU option View is created in res/menu folder. A MENU item "Get Location" is added to allow user to get restaurant's GPS coordinates





Location

❑Controller – YES

The changes will be done at *DetailForm* Controller

- ✓When MENU button is pressed, the *onCreateOptionsMenu(Menu)* is used to inflate the *details_option.xml* layout at bottom of the UI View
- ✓*onOptionsItemSelected(MenuItem)* detects "Get Location" MENU option item to retrieve GPS location

Location

- Controller – YES

  The changes will be done at *DetailForm* Controller

  ✓ *GPSTracker* object (which is a sub-class of *Service* with *LocationListener* interface implemented) will be created upon creation of *DetailForm* Activity

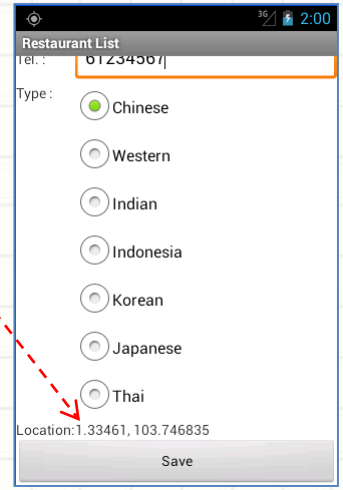
  ✓ onDestroy() callback methods is updated to stop the *GPSTracker* service and closing of database

View – Restaurant Form & Menu

View

Restaurant Form

- The *detail_form.xml* layout has been modified
  - A *TextView* widget added to display GPS location information



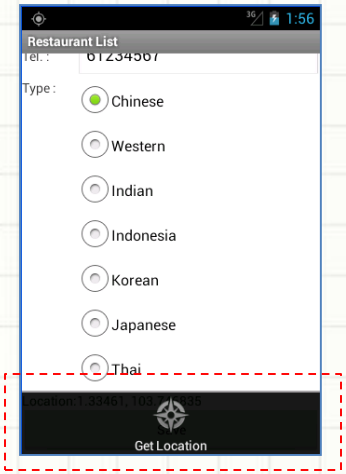

```
<TableRow>

    <TextView android:text="Location:" />

    <TextView
        android:id="@+id/location"
        android:text="(not set)" />
</TableRow>
```

View

MENU Option

- A *details_option.xml* layout is created for the MENU option

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/location"
        android:icon="@drawable/ic_menu_compass"
        android:title="Get Location"/>

</menu>
```



Model – RestaurantHelper

Model

RestaurantHelper

- Restaurants table model is modified to store latitude and longitude of a restaurant

```
@Override
public void onCreate(SQLiteDatabase db) {
    // Will be called once when the database is not created
    db.execSQL("CREATE TABLE restaurants_table (_id INTEGER PRIMARY KEY AUTOINCREMENT, "
            + "name TEXT, address TEXT, telephone TEXT, restaurantType TEXT, "
            + "lat REAL, lon REAL);");
}
```

Model

RestaurantHelper

- For upgrading of existing app, the SCHEMA_VERSION is required to increase. When the existing SCHEMA_VERSION (old version) is smaller then the new SCHEMA_VERSION, the onUpgrade() method will be called and the restaurants_table will be added with the latitude and longitude fields. With that all records in the existing database will not be deleted

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    if (oldVersion < SCHEMA_VERSION) {
        db.execSQL("ALTER TABLE restaurants_table ADD COLUMN lat REAL");
        db.execSQL("ALTER TABLE restaurants_table ADD COLUMN lon REAL");
    }
}
```

Model

RestaurantHelper

- The two query methods, getAll() and getById(), are updated to return latitude and longitude data from restaurants table Model

```
/* Read all records from restaurants_table in order specified by Preference */
public Cursor getAll(String orderBy) {
    return (getReadableDatabase()
            .rawQuery(
                    "SELECT _id, name, address, telephone, restaurantType, "
                          + "lat, lon FROM restaurants_table ORDER BY "
                          + orderBy, null));
}

/* Read a record from restaurants_table by ID provided */
public Cursor getById(String id) {
    String[] args = { id };

    return (getReadableDatabase().rawQuery(
            "SELECT _id, name, address, telephone, restaurantType, "
                    + "lat, lon FROM restaurants_table WHERE _ID=?", args));
}
```

Model

RestaurantHelper

- Method update() is changed to include latitude and longitude values to existing record in restaurants_table Model

```
public void update(String id, String name, String address,
        String telephone, String restaurantType, double lat, double lon) {
    ContentValues cv = new ContentValues();
    String[] args = { id };
    cv.put("name", name);
    cv.put("address", address);
    cv.put("telephone", telephone);
    cv.put("restaurantType", restaurantType);
    cv.put("lat", lat);
    cv.put("lon", lon);

    getWritableDatabase().update("restaurants_table", cv, "_ID=?", args);
}
```

Model

RestaurantHelper

- New getter methods to return latitude and longitude data from Cursor Model are added

```
SELECT _id, name, address, telephone, restaurantType, lat, lon FROM restaurants_table
```

0 1 2 3 4 5 6

```
public double getLatitude(Cursor c) {
    return (c.getDouble(5));
}

public double getLongitude(Cursor c) {
    return (c.getDouble(6));
}
```

Controller – DetailForm

Controller

- GPS radio is normally not on. On creation of *DetailForm* Controller, the *GPSTracker* service is created to retrieve the *LocationManager* for controlling location update through getSystemService() method

DetailForm

```
private GPSTracker gpsTracker;

gpsTracker = new GPSTracker(DetailForm.this);
```

GPSTracker

```
// Declaring a Location Manager
protected LocationManager locationManager;

locationManager = (LocationManager) mContext
        .getSystemService(LOCATION_SERVICE);
```

Controller

- Upon creation of GPSTracker service, the *LocationManager* will first check on GPS and network status before starting to fetch location data via the requestLocationUpdates() method

GPSTracker

```
//Getting GPS status
//This provider determines location using satellites.
//Depending on conditions, this provider may take a while
//to return a location fix. Requires the permission ACCESS_FINE_LOCATION.
isGPSEnabled = locationManager
        .isProviderEnabled(LocationManager.GPS_PROVIDER);

//Getting network status
//This provider determines location based on availability of cell tower
//and WiFi access points. Results are retrieved by means of a network lookup
isNetworkEnabled = locationManager
        .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
```

Controller

- If user has enabled the network provider in the phone Settings menu, *LocationManager* fetches location data based on availability of cell tower and WiFi access points

```
if (isNetworkEnabled) {
    locationManager.requestLocationUpdates(
            LocationManager.NETWORK_PROVIDER,
            MIN_TIME_BW_UPDATES,
            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
    Log.d("Network", "Network");
    if (locationManager != null) {
        location = locationManager
                .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
        if (location != null) {
            latitude = location.getLatitude();
            longitude = location.getLongitude();
        }
    }
}
```

Controller

- If user has enabled the GPS provider in the phone Settings menu, *LocationManager* fetches location data using satellites. Depending on conditions, this provider may take a while to return a location fix

```
if (isGPSEnabled) {
    if (location == null) {
        locationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER,
                MIN_TIME_BW_UPDATES,
                MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
        Log.d("GPS Enabled", "GPS Enabled");
        if (locationManager != null) {
            location = locationManager
                    .getLastKnownLocation(LocationManager.GPS_PROVIDER);
            if (location != null) {
                latitude = location.getLatitude();
                longitude = location.getLongitude();
            }
        }
    }
}
```

Controller

- When user taps on the “Get Location” MENU item, the *GPSTracker* getLatitute() and getLongitude() methods will be called and *LocationManager* will return the coordinates data

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.location) {
        // check if GPS enabled
        if(gpsTracker.canGetLocation()){
            latitude = gpsTracker.getLatitude();
            longitude = gpsTracker.getLongitude();

            location.setText(String.valueOf(latitude) + ", "
                    + String.valueOf(longitude));

            // \n is for new line
            Toast.makeText(getApplicationContext(), "Your Location is - \nLat: "
                    + latitude + "\nLong: " + longitude, Toast.LENGTH_LONG).show();
        }else{
            // can't get location
            // GPS or Network is not enabled
            // Ask user to enable GPS/network in settings
            gpsTracker.showSettingsAlert();
        }
        return (true);
    }
    return (super.onOptionsItemSelected(item));
}
```

Controller

- It is possible that user will leave the *DetailForm* activity (i.e. when BACK button or "Save" is pressed) while waiting on GPS fix. The *DetailForm* Controller will need to stop requesting GPS updates by stopping the *GPSTracker* service

```
@Override
public void onDestroy() {
    super.onDestroy();
    gpsTracker.stopSelf();
    helper.close();
}
```

Controller

AndroidManifest.xml

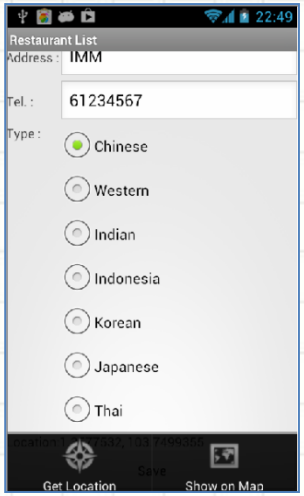
- In order to allow *LocationManager* to have access to GPS information, a permission must be set in manifest

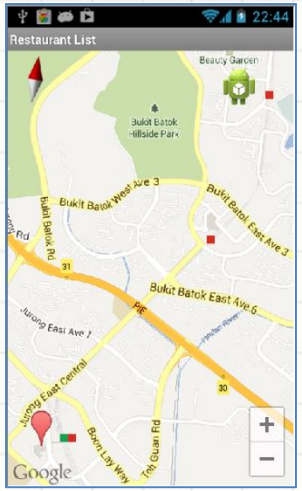
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="sp.com"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="16" />

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

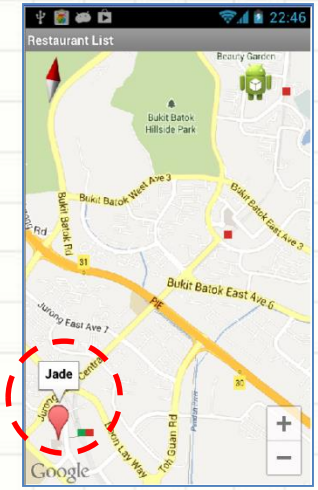
MAP

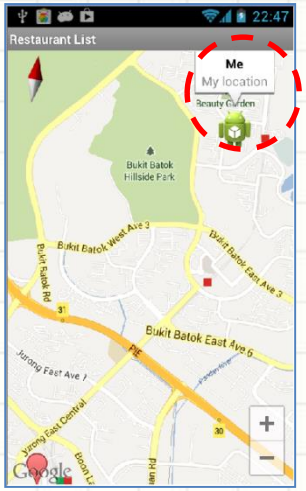
Map

- In second part of Practical 6 exercise, *RestaurantMap* Activity and *MapFragment* are integrated to show the restaurant and user locations on Google map









When tap on the Marker

Map

- Let's check what do we need to modify from the previous exercise to show restaurant and user current location using Google map?
  - Model - Any change in Data Model?
  - View - Do you need to modify any of the user interface view?
  - Controller - Do you need to tell the Controller to do any thing new?

Map

- Model – NO

  Since there is nothing change in data Model and handling methods, there will be no change for RestaurantHelper

Map

- View – YES

  A “Show on Map” MENU item is added to *details_option.xml* layout and a *activity_restaurant_map.xml* layout is created in res/layout folder for loading the *RestaurantMap* FragmentActivity

Map

❑Controller – YES

There are some changes at the Controller

✓DetailForm Controller

➢Update onOptionsItemSelected(MenuItem) to detect "Show Map" MENU option item pressed and do an Explicit Intent call to *RestaurantMap* FragmentAcitivity

Map

- Controller – YES

  There are some changes at the Controller

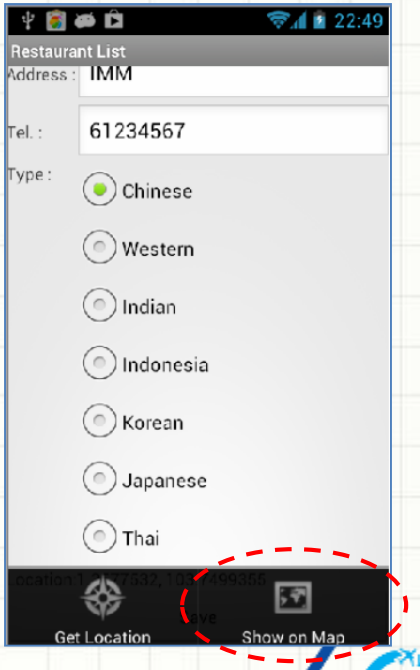
  - ✓ RestaurantMap Controller, sub-class of FragmentActivity, is created to handle the display of restaurant and user's locations on Google map

View – Map

View

- An extra “Show on Map” item is added to *details_option.xml* layout for Map display selection

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".RestaurantList">
    <item
        android:id="@+id/get_location"
        android:orderInCategory="100"
        android:title="Get Location"
        app:showAsAction="never" />

    <item
        android:id="@+id/map"
        android:orderInCategory="100"
        android:title="Show Map"
        app:showAsAction="never" />

</menu>
```



View

- For viewing map, a *activity_restaurant_map.xml* layout is created in res/layout folder
  - fragment - Fragments can then be thought of as "mini-controllers" or components that can be dropped into Activities either at runtime or through an Activity's layout XML

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/restaurant_map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.sp.restaurantlist.RestaurantMap" />
```

View

- For viewing map, a *activity_restaurant_map.xml* layout is created in res/layout folder
  - SupportMapFragment - A Map component in an app. This fragment is the simplest way to place a map in an application. It's a wrapper around a view of a map to automatically handle the necessary life cycle needs

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/restaurant_map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.sp.restaurantlist.RestaurantMap" />
```

Controller – DetailForm & RestaurantMap

Controller

DetailForm

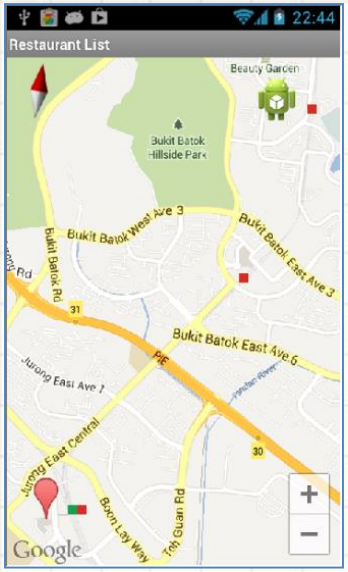
- Update onOptionsItemSelected() method to capture “Show on Map” MENU item pressed and do an Explicit Intent call to *RestaurantMap* activity with extra parameters (restaurant and user’s GPS coordinates)

```
} else if (item.getItemId() == R.id.map) {
    //Get my current location
    myLatitude = gpsTracker.getLatitude();
    myLongitude = gpsTracker.getLongitude();

    Intent i = new Intent(this, RestaurantMap.class);
    i.putExtra("LATITUDE", latitude);
    i.putExtra("LONGITUDE", longitude);
    i.putExtra("MYLATITUDE", myLatitude);
    i.putExtra("MYLONGITUDE", myLongitude);
    i.putExtra("NAME", name.getText().toString());
    startActivity(i);
    return (true);
}
```

Controller

RestaurantMap

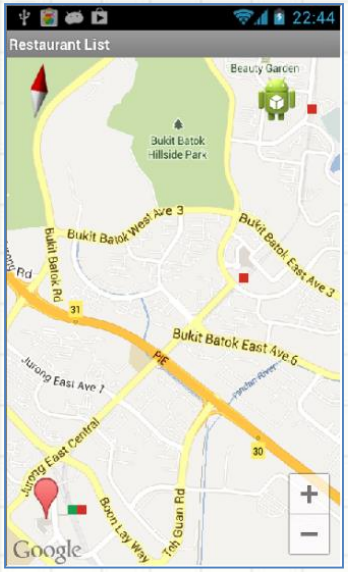
- The *RestaurantMap* Controller in-charges of setting up the UI View and plotting the restaurant and user's location on Google map with markers



Controller

RestaurantMap

- CameraUpdateFactory class containing methods for creating CameraUpdate objects that change a map's camera is imported to allow map manipulation



Controller

RestaurantMap

```java
public class RestaurantMap extends FragmentActivity implements OnMapReadyCallback {
    private GoogleMap mMap;

    private double lat;
    private double lon;
    private String restaurantName;
    private double myLat;
    private double myLon;
    private LatLng RESTAURANT;
    private LatLng ME;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_restaurant_map);

        lat = getIntent().getDoubleExtra("LATITUDE", 0);
        lon = getIntent().getDoubleExtra("LONGITUDE", 0);
        restaurantName = getIntent().getStringExtra("NAME");
        myLat = getIntent().getDoubleExtra("MYLATITUDE", 0);
        myLon = getIntent().getDoubleExtra("MYLONGITUDE", 0);

        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.restaurant_map);
        mapFragment.getMapAsync(this);
    }
```

```java
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        RESTAURANT = new LatLng(lat, lon);
        ME = new LatLng(myLat, myLon);

        Marker restaurant = mMap.addMarker(new MarkerOptions().position(RESTAURANT).title(restaurantName));
        Marker me = mMap.addMarker(new MarkerOptions().position(ME).title("ME")
                .snippet("My location")
                .icon(BitmapDescriptorFactory.fromResource(R.drawable.marker_me)));

        // Move the camera instantly to restaurant with a zoom of 15.
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(RESTAURANT, 15));
    }
}
```

Controller

AndroidManifest.xml

- To use Google Map v2 in the Restaurant List app,
  - A valid Google Map API key needs to be included
  - The key is unique to individual computer

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyBD5a5zQ6Q3N14jIqvLDqoBtSvV1coICMg" />
```

Controller

AndroidManifest.xml

- For any new *Activity* (*RestaurantMap* Activity) created must also be registered to the manifest file. Otherwise, an error will occur during run time

```
<activity
    android:name="sp.com.RestaurantList"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name="sp.com.DetailForm" />
<activity android:name="sp.com.EditPreferences" />
<activity android:name="sp.com.AlarmActivity" />
<activity android:name="sp.com.RestaurantMap" />
```

END