# Assignment A4 - RESTful Web Services, Google Maps and Services

In this assignment, you'll create an application to track an alien invasion. A RESTful web service is provided that tells you the current location of several alien ships. Your application will use this service to ascertain the ship locations and display them on your device using Google Maps.

## Prerequisites

This assignment assumes you have completed the following modules

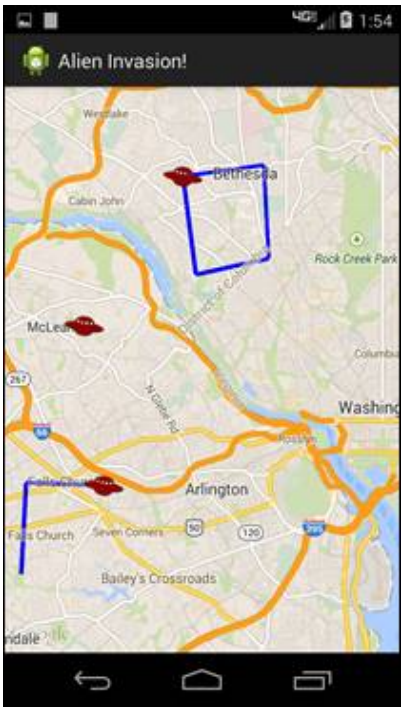
- 1.2 Android Versions
- 1.3 Development Environment
- 1.5 Gradle Basics
- 1.7 Activities
- 2.1 Intents
- 2.2 RecyclerView and CardView
- 2.3 Toolbars
- 3.1 Fragments
- 3.2 Databases and Content Providers
- 4.1 Gradle Details
- 4.3 Permissions
- 6.1 Services
- 6.2 RESTful Web Services
- 7.1 Maps

NOTE: Be sure to name this project HW4 and its packages starting with lastname.firstname.hw4.

Your application will use an Android service to fetch changes via the RESTful web service. The Android service will poll for changes once per second.

*Note this this a **very** bad way to implement this in real life. Ideally you would use something like Google Cloud Messaging to push updates to devices rather than poll. However, I thought it was more important for you to try communicating between an Activity and an Android service than set up Google Cloud Messaging.*

***The aliens are trying to tell us something! Figure out a way to see what they're saying!***



Sample Screen

# Application Parts

For this assignment, you'll need

- A Service to poll for alien invasion updates
- An Activity to display the map, ship icons, and the paths the ships have taken
- A Parcelable representation of a UFO position
- AIDL files
    - A description of the UFO position (UFOPosition.aidl) - it only needs to contain:

```
package lastname.firstname.hw5;

parcelable UFOPosition;
```

    - A reporter interface that the service can use to communicate status updates to the activity (it will pass UFOPosition objects). The report(...) method should pass a List of UFOPosition objects.
    - The alien service's interface (allows add/remove of the reporter)

***Note: Use a single project for your assignment - the service and activity should be in the same project. Note that by using a single project we don't really need to use AIDL or make the UFOPosition be Parcelable (because they're in the same memory space). However, I want you to use AIDL and make UFOPosition Parcelable to demonstrate what you need do to make the Service be accessible from other projects.***

# UFOPosition

This will be a Parcelable class (so it can be passed between the service and the activity) that contain the following information

- int shipNumber
- double lat
- double lon

# The Service

Your service must do the following:

- Allow Activities to bind to the service
- Allow reporter instances to be added and removed
- Start a thread to perform the REST requests
    - ***THERE SHOULD BE ONE THREAD IN YOUR SERVICE. DO NOT START A SEPARATE THREAD FOR EACH REST REQUEST.***
    - **_Do not use AsyncTask in your service - it communicates with a UI thread, which you should not be using in your service. The REST example given in the lecture wraps the HTTP calls in an AsyncTask but that is not needed for this assignment - extract the HTTP code that's needed for the REST calls from that AsyncTask and use it directly in your service thread. _**
    - Once per second, make a request to http://javadude.com/aliens/n.json where "n" is a number starting at 1, and should be incremented for each request. (Note that I'm simulating REST requests by using static JSON files on my web server.) To see an example of the JSON, point your browser to http://javadude.com/aliens/1.json
    - If a request returns a 404 (HttpStatus.NOT_FOUND) status code, stop polling. *Do not assume the number of reports that are available.*
    - Parse the returned JSON. It will be a JSONArray containing JSONObjects that represent each visible ship's position. The JSONObjects contain
    - "ship" - an int representing which ship is being reported
    - "lat" - a double representing the latitude of the ship
    - "lon" - a double representing the longitude of the ship
    - Create an instance of UFOPosition to hold the data for each ship
    - Create a List of UFOPositions and pass it to all registered reporters

# The Activity

Notes:

- Lock your screen orientation on the activity by adding the following to the tag in the manifest:

  ```
  android:screenOrientation="portrait"
  ```

- You *do not* need to worry about saving/restoring data for configuration changes or if the user presses back or home while running the application. These would be necessary in a real application, but this assignment already has a good deal of features to focus upon. In other words, it's ok if everything restarts if back/home is pressed and the application is re-entered.
- Your report() method will run in a non-UI thread. To update the map, you'll need to run code in the UI thread. To do this, make sure all code inside report() runs inside runOnUIThread.

For example:

```
public void report(...) {
    runOnUiThread(new Runnable() {
        public void run() {
            // UI updating code
        }});
    }
```

Your activity must do the following:

- First create an Activity that only displays a map centered on lat=38.9073, lon=-77.0365. Test this to be sure you have properly set up the google maps API and properly registered an Android key. Do not write any other code until you have this working!
- Bind to your service
- When the service has been connected, add a reporter to be notified of ship positions
- Track the bounding box that will be large enough to contain all UFO positions that you have seen. You can use a com.google.android.gms.maps.model.LatLngBounds object for this.
- When you receive a list of UFOPositions
  - If a position represents a new ship, create a new UFO marker on the map (and whatever data you need to save its positions)
  - If a position represents an existing ship, update its marker's position
  - If the ship has been reported for more than one location, draw a line on the map representing the path from its last reported position to its current position. *Note: **do not** draw navigation directions! Only draw a straight line between points using Google Map's PolyLine. You only need to draw PolyLines for each new line; the map will remember the previous lines (until you exit and re-enter)*
  - If a ship was previously reported but is *not* in the current report list, remove its marker from the map. *Do not remove the lines that represented its path!*
  - Whenever you see a new position for a UFO, add that position to your bounds so it will expand to include that position. You can do this using

    ```
    bounds = bounds.including(latLng);
    ```

  - Update the map to include the new bounds using (use a value in a dimens.xml file to set the padding to something reasonable)

    ```
    googleMap.animateCamera(CameraUpdateFactory.newLatLngBounds(bounds, padding));
    ```

  - Note: To center your map icons over the proper lat/lng, use

    ```
    markerOptions.anchor(0.5f, 0.5f);
    ```

- Be sure to unbind from the service when you stop your application
- Get the UFO icons from ufo-icons.zip . Rights for use of these icons are at http://icons.mysitemyway.com/terms-of-use

## Submitting Your Assignment

To submit your assignment (do this outside Android Studio):

1. Close the project in Android Studio (and/or quit Android Studio)

2. Find the directory for your project
3. Delete the build and app/build directories from the project folder
4. Zip the project directory. Be sure to get all files and subdirectories, including those that start with '.'. Be sure to name the zip file lastname.firstname.hw4
5. Go to the Blackboard course website
6. Click on the ***Assignment Submission*** link on the left navigation bar
7. Click on the ***Assignment A4*** link
8. Attach your submission and enter any notes
9. Be sure to ***submit*** your assignment, not just save it!

---