Displaying a Location Address

THIS LESSON TEACHES YOU TO

1. Define the Address Lookup Task
2. Define a Method to Display the Results
3. Run the Lookup Task

YOU SHOULD ALSO READ

- Setup Google Play Services SDK
- Retrieving the Current Location
- Receiving Location Updates

TRY IT OUT

Download the sample app

LocationUpdates.zip

The lessons Retrieving the Current Location (retrieve-current.html) and Receiving Location Updates (receive-location-updates.html) describe how to get the user's current location in the form of a `Location (/reference/android/location/Location.html)` object that contains latitude and longitude coordinates. Although latitude and longitude are useful for calculating distance or displaying a map position, in many cases the address of the location is more useful.

The Android platform API provides a feature that returns an estimated street addresses for latitude and longitude values. This lesson shows you how to use this address lookup feature.

> **Note:** Address lookup requires a backend service that is not included in the core Android framework. If this backend service is not available, `Geocoder.getFromLocation() (/reference/android/location/Geocoder.html#getFromLocation(double, double, int))` returns an empty list. The helper method `isPresent() (/reference/android/location/Geocoder.html#isPresent())`, available in API level 9 and later, checks to see if the backend service is available.

The snippets in the following sections assume that your app has already retrieved the current location and stored it as a `Location (/reference/android/location/Location.html)` object in the global variable `mLocation`.

Define the Address Lookup Task

To get an address for a given latitude and longitude, call `Geocoder.getFromLocation() (/reference/android/location/Geocoder.html#getFromLocation(double, double, int))`, which returns a list of addresses. The method is synchronous, and may take a long time to do its work, so you should call the method from the `doInBackground() (/reference/android/os/AsyncTask.html#doInBackground(Params...))` method of an `AsyncTask (/reference/android/os/AsyncTask.html)`.

While your app is getting the address, display an indeterminate activity indicator to show that your app is working in the background. Set the indicator's initial state to `android:visibility="gone"`, to make it invisible and remove it from the layout hierarchy. When you start the address lookup, you set its visibility to "visible".

The following snippet shows how to add an indeterminate `ProgressBar (/reference/android/widget/ProgressBar.html)` to your layout file:

```
<ProgressBar
android:id="@+id/address_progress"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:indeterminate="true"
android:visibility="gone" />
```

To create the background task, define a subclass of `AsyncTask (/reference/android/os/AsyncTask.html)` that calls getFromLocation() (/reference/android/location/Geocoder.html#getFromLocation(double, double, int)) and returns an address. Define a TextView (/reference/android/widget/TextView.html) object mAddress to contain the returned address, and a ProgressBar (/reference/android/widget/ProgressBar.html) object that allows you to control the indeterminate activity indicator. For example:

```
public class MainActivity extends FragmentActivity {
    ...
    private TextView mAddress;
    private ProgressBar mActivityIndicator;
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ...
    mAddress = (TextView) findViewById(R.id.address);
    mActivityIndicator =
            (ProgressBar) findViewById(R.id.address_progress);
    }
    ...
    /**
    * A subclass of AsyncTask that calls getFromLocation() in the
    * background. The class definition has these generic types:
    * Location - A Location object containing
    * the current location.
    * Void     - indicates that progress units are not used
    * String   - An address passed to onPostExecute()
    */
    private class GetAddressTask extends
            AsyncTask<Location, Void, String> {
        Context mContext;
        public GetAddressTask(Context context) {
            super();
            mContext = context;
        }
        ...
        /**
         * Get a Geocoder instance, get the latitude and longitude
         * look up the address, and return it
         *
         * @params params One or more Location objects
         * @return A string containing the address of the current
         * location, or an empty string if no address can be found,
         * or an error message
         */
        @Override
        protected String doInBackground(Location... params) {
            Geocoder geocoder =
                    new Geocoder(mContext, Locale.getDefault());
            // Get the current location from the input parameter list
            Location loc = params[0];
            // Create a list to contain the result address
            List<Address> addresses = null;
            try {
                /*
                 * Return 1 address.
                 */
                addresses = geocoder.getFromLocation(loc.getLatitude(),
                        loc.getLongitude(), 1);
            } catch (IOException e1) {
```

```
            Log.e("LocationSampleActivity",
                    "IO Exception in getFromLocation()");
            e1.printStackTrace();
            return ("IO Exception trying to get address");
            } catch (IllegalArgumentException e2) {
            // Error message to post in the log
            String errorString = "Illegal arguments " +
                    Double.toString(loc.getLatitude()) +
                    " , " +
                    Double.toString(loc.getLongitude()) +
                    " passed to address service";
            Log.e("LocationSampleActivity", errorString);
            e2.printStackTrace();
            return errorString;
            }
            // If the reverse geocode returned an address
            if (addresses != null && addresses.size() > 0) {
                // Get the first address
                Address address = addresses.get(0);
                /*
                 * Format the first line of address (if available),
                 * city, and country name.
                 */
                String addressText = String.format(
                        "%s, %s, %s",
                        // If there's a street address, add it
                        address.getMaxAddressLineIndex() > 0 ?
                                address.getAddressLine(0) : "",
                        // Locality is usually a city
                        address.getLocality(),
                        // The country of the address
                        address.getCountryName());
                // Return the text
                return addressText;
            } else {
                return "No address found";
            }
        }
        ...
    }
    ...
}
```

The next section shows you how to display the address in the user interface.

Define a Method to Display the Results

doInBackground() (/reference/android/os/AsyncTask.html#doInBackground(Params...)) returns the result of the address lookup as a String (/reference/java/lang/String.html). This value is passed to onPostExecute() (/reference/android/os/AsyncTask.html#onPostExecute(Result)), where you do further processing on the results. Since onPostExecute() (/reference/android/os/AsyncTask.html#onPostExecute(Result)) runs on the UI thread, it can update the user interface; for example, it can turn off the activity indicator and display the results to the user:

```
    private class GetAddressTask extends
            AsyncTask<Location, Void, String> {
        ...
        /**
```

```
         * A method that's called once doInBackground() completes. Turn
         * off the indeterminate activity indicator and set
         * the text of the UI element that shows the address. If the
         * lookup failed, display the error message.
         */
        @Override
        protected void onPostExecute(String address) {
            // Set activity indicator visibility to "gone"
            mActivityIndicator.setVisibility(View.GONE);
            // Display the results of the lookup.
            mAddress.setText(address);
        }
        ...
    }
```

The final step is to run the address lookup.

Run the Lookup Task

To get the address, call execute() (/reference/android/os/AsyncTask.html#execute(Params...)). For example, the following snippet starts the address lookup when the user clicks the "Get Address" button:

```
public class MainActivity extends FragmentActivity {
    ...
    /**
     * The "Get Address" button in the UI is defined with
     * android:onClick="getAddress". The method is invoked whenever the
     * user clicks the button.
     *
     * @param v The view object associated with this method,
     * in this case a Button.
     */
    public void getAddress(View v) {
        // Ensure that a Geocoder services is available
        if (Build.VERSION.SDK_INT >=
                Build.VERSION_CODES.GINGERBREAD
                            &&
                Geocoder.isPresent()) {
            // Show the activity indicator
            mActivityIndicator.setVisibility(View.VISIBLE);
            /*
             * Reverse geocoding is long-running and synchronous.
             * Run it on a background thread.
             * Pass the current location to the background task.
             * When the task finishes,
             * onPostExecute() displays the address.
             */
            (new GetAddressTask(this)).execute(mLocation);
        }
        ...
    }
    ...
}
```

The next lesson, Creating and Monitoring Geofences (geofencing.html), demonstrates how to define locations of interest called **geofences** and how to use geofence monitoring to detect the user's proximity to a location of interest.