GoogleAPIRoadsData.Jar

Key-Configuration

Obtainging API Keys

The GoogleAPIRoadsData.jar package provides classes and methods for interacting with three Google Maps APIs and storing data from API request. The package supports the Distance Matrix, Directions, and Geocoding APIs. In order to use the APIs, users of this package will need to provide API keys from google. Instructions for obtaining a Google API key can be found here:

<https://developers.google.com/maps/documentation/javascript/get-api-key>

Installing API Keys

After obtaining an API Key save the key in an empty text file. Keys must be updated before calling the APIs. There are 4 ways to update the API Keys.

1. Manual Configuration

In a manual key configuration navigate to the KEYS.java file and replace the null instantiations for String *keyDistance*, String *keyDirections,* and String *keyGeoCode*.

1. Set String Key Methods

By using the static methods setStringKeyDist(), setStringKeyDir(), and setStringKeyGeo() before making an API call users can update only the keys they intend to use. If each user has their own personal keys or if the project has different keys for each API this would be a simple method.

These keys are called in the following ways:

KEYS.setStringKeyDist(“YOUR\_KEY\_HERE”);

KEYS.setStringKeyDir(“YOUR\_KEY\_HERE”);

KEYS.setStringKeyGeo(“YOUR\_KEY\_HERE”);

1. Set File Key Methods

By using the static methods setFileKeyDist(), setFileKeyDir(), and setFileKeyGeo() before making an API call users can update only the key(s) they intent to use. Pulling from a file allows the users to record their key outside of their program.

These keys are called in the following ways:

KEYS.setKeyDist(“C:\\Example\\Pathway\\KeyFile.txt”);

KEYS.setKeyDir(“C:\\Example\\Pathway\\KeyFile.txt”);

KEYS.setKeyGeo(“C:\\Example\\Pathway\\KeyFile.txt”);

1. Single Shared Key

If the project uses one key with permissions for all three keys, the singleSharedFileKey() and singleSharedStringKey() methods will update all three keys from either a file path or string respectively.

These keys are called in the following ways:

KEYS.singleSharedStringKey(“YOUR\_KEY\_HERE”);

KEYS.singleSharedFileKey(“C:\\Example\\Pathway\\KeyFile.txt”);

DataStore Objects

DataStore is an object that holds a Starting(Origin) Location and an Ending(Destination) Location, as well as the distance between the points and directions from the Starting Location to the Ending Location. The Starting Location and Ending Location can be Addresses, Longitude and Latitude Pairs, or Zip Codes. Distance and Directions can be calculated with the calcDistance() and calcDirections() methods.

To create a DataStore Object and find the Distance and Directions between the points use the following lines of code:

//Using Addresses

DataStore MyDataStore = new DataStore(“Butler PA”, “Slippery Rock PA”);

MyDataStore.calcDistance();

MyDataStore.calcDirections();

MyDataStore.printDistance();

System.out.println(MyDataStore.Holder.toString());

//Using Longitude and Latitude

DataStore MyDataStore = new DataStore(“40.8641043,-79.8938038”, “34.1220545,-118.2935891”);

MyDataStore.calcDistance();

MyDataStore.calcDirections();

MyDataStore.printDistance();

System.out.println(MyDataStore.Holder.toString());

//Using Zip Codes

DataStore MyDataStore = new DataStore(“16002”, “16057”);

MyDataStore.calcDistance();

MyDataStore.calcDirections();

MyDataStore.printDistance();

System.out.println(MyDataStore.Holder.toString());

Geo-Coordinates

To convert between Geo-Coordinates and Human-Readable Address the API\_GeoCode.java class has methods for converting between each form.

//To obtain an Address from Coordinates

API\_GeoCode myGeoCode = new API\_GeoCode();

myGeoCode.fetchAddressFromCoordinates(myGeoCode.buildFromCoordinates(

“40.8641043”, “-79.8938038”);

System.out.println(myGeoCode.getAddress());

//To obtain Coordinates from an Address

API\_GeoCode myGeoCode = new API\_GeoCode(“322 N McKean St Butler PA”);

myGeoCode.fetchCoordinatesFromAddress(myGeoCode.buildFromAddress(

myGeoCode.getAddress())):

System.out.println(myGeoCode.getLat() + myGeoCode.getLng());