🌐 Google Maps Wrapper 🗺️

A .NET wrapper to intergrate with Google Maps APIs.
Supports: Static API, Elevation API, Geocoding API Javascript API

Project Owner: @ScryptSnake https://github.com/ScryptSnake/GoogleMapsWrapper

GoogleMapsWrapper is a project that interfaces with Google Maps Platform APIs.
This project does not attempt to provide a one-to-one .NET mapping of Google's platforms, but rather, is a tool that provides utility by using these API services. This project is best suited as a supplement to other projects, but can be used as a standalone tool. This is a class library that targets NET 7.0.

> *GeocodeAPI:* acquires geolocation data, to include street address, local municipality, state, country and other information. Elevation information can also be returned from this service. All requests in this service are passed a single GPS coordinate parameter to retrieve data.

> *StaticMapsAPI:* produces a static image of a map. Maps are highly customizable and may contains placemarks and other objects on the map.

> *JavascriptAPI:* spawns a web browser to work interactively with a map. Data can be sent/received from .NET.
> See Project Status

⭐ Getting Started

Clone the repository and create a project reference to GoogleMapsWrapper and begin implementing into your project. Additionally, a simple Winforms demo app is included to showcase some basic features of the wrapper.

> *Note: This application requires a valid Google Maps API key and can be obtained from here.*

Keys should be contained within an IConfiguration passed to the application entry point. The consumer of the configuration is the engine and uses the key directly to create *KeyedRequests*.

Security Note / Keyed Requests:

A user's API key should be contained within the IConfiguration passed at the application entry point. The engine takes the config as a constructor parameter and therefore has access to the key. When requests are created and passed to the engine for sending, the provided requests Uri does not contain the key. Likewise, when a response is returned from the engine, the SentRequest property (stored request) of the response does not contain the Uri with the key. The engine constructs a private

KeyedRequest object from the received request, which contains the key, and therefore can be appended to the Uri and sent via HttpClient. This choice in design was chosen to ensure stored requests within responses (*SentRequest* property) do not expose the key to other parts of the application.

💡 Project Status

This project continues to be contributed to over time. Users should avoid the Javascript API and consider it's current status *experimental*.

❗ Usage Disclaimer

This software attempts to optimize the use of requests to Google's API. However, users are solely responsible for all requests made from their accounts. The developer assumes no liability for any charges incurred due to excessive requests or for the frequency of requests sent to Google's services. Further, the developer does not make any guarantees of accuracy of data yielded from this software.

🔧 Architecture

This application can be conceptually divided into 7 primary components.

- **API**
  High level objects that contain the core functionality of end usage. The primary entry point of the application is GoogleMapsApi which exposes each *sub-API* (GeocodeApi and StaticMapsApi).

- **Engine**
  Responsible for sending requests and receiving responses to the API endpoint. Request objects are constructed in an API and passed to the engine for http processing. The engine is capable of retrieving http responses in two types: JsonDocument and byte array.

- **Requests**
  Constructed by an API and contains relevant details about the request, as well as the Uri. A request is attached to a response object and returned from the engine. A user API key is never be contained within this object.

- **Responses**
  Holds data returned from the engine/endpoint and relevant information about the request. A response contains a Parse() method allowing the consumer to parse the response with a Parser into a container.

- **Parsers**
  A parser is an object which is responsible for taking a response of some data type (JSON or byte array) and containerizing the data into an IContainer (container).

- **Containers**
  These are DTOs or data containers that hold parsed information from a response. Usually provided via a *Parser*.

- **Elements**
  DTO objects which model map features. This includes Map, Marker, Polyline. Elements are used throughout the application, primarily for passing as parameters to an API method.

The javascript API is conceptualized as a different component entirely.

✅ Credits

Thank you for viewing this project. I hope you find it interesting.

Namespace GoogleMapsWrapper.Api

Classes

GeocodeApi

Provides methods for retrieving data from Google's Geocoding API.

GoogleMapsApi

The entry point of the wrapper. Exposes API objects to interact with the endpoint:

StaticMapCustomIcon

An object that represents a custom icon for a marker. The source must be a Uri of an image.

StaticMapsApi

Provides methods for retrieving a static image of a map from Google's Static API. This object leverages an IApiEngine to send IRequests and return data in form of IResponse<TResponse>s.