GraphQL for Unity Documentation

Andreas Vogler
andreas.vogler@rocworks.at

Table of Contents

1 Introduction

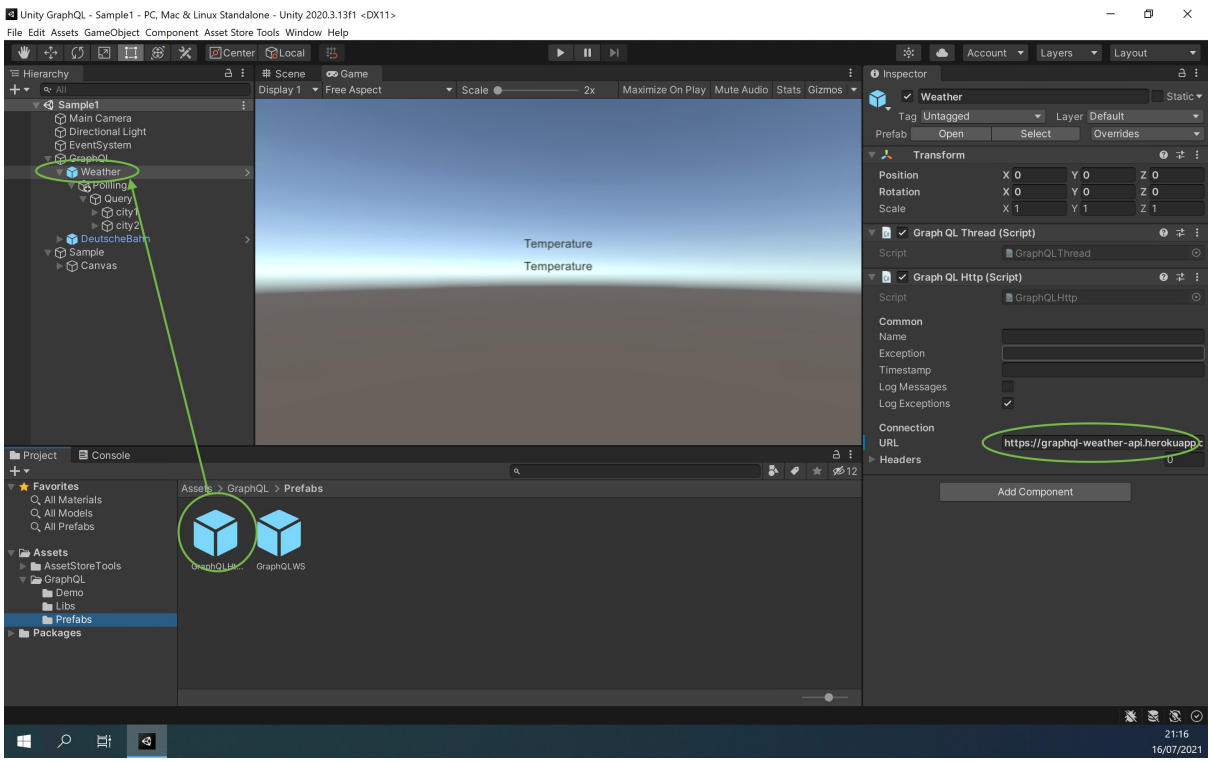
Simple GraphQL tools for Unity. GraphQL queries can be executed over a HTTP or a Websocket connection and the JSON result can be read directly from the Query object as a JSON variable or the result can be mapped to a structured set of GameObjects (JSON to GamesObjects). It supports Queries, Mutations and Subscriptions for real time updates.

2 How to use GraphQL for Unity

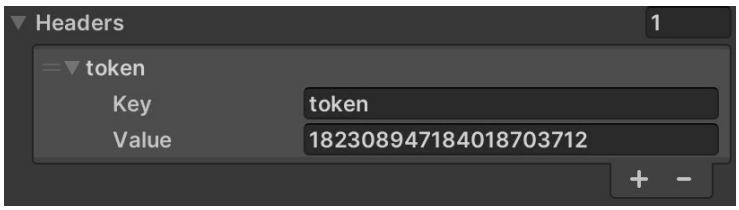
2.1 Connection

Drag from the Prefabs directory the Prefab "GraphQL Http" or "GraphQL WebSocket" to your scene and set the connection URL to your GraphQL Server. In the demo scene we use some public available GraphQL servers.

If you need GraphQL Subscriptions, then you must use a GraphQL WebSocket connection. With just HTTP it is not possible to get values unsolicited from the system.



If you need headers for your GraphQL server – for example for authorization – then you can set header variables and values in the section below the URL.

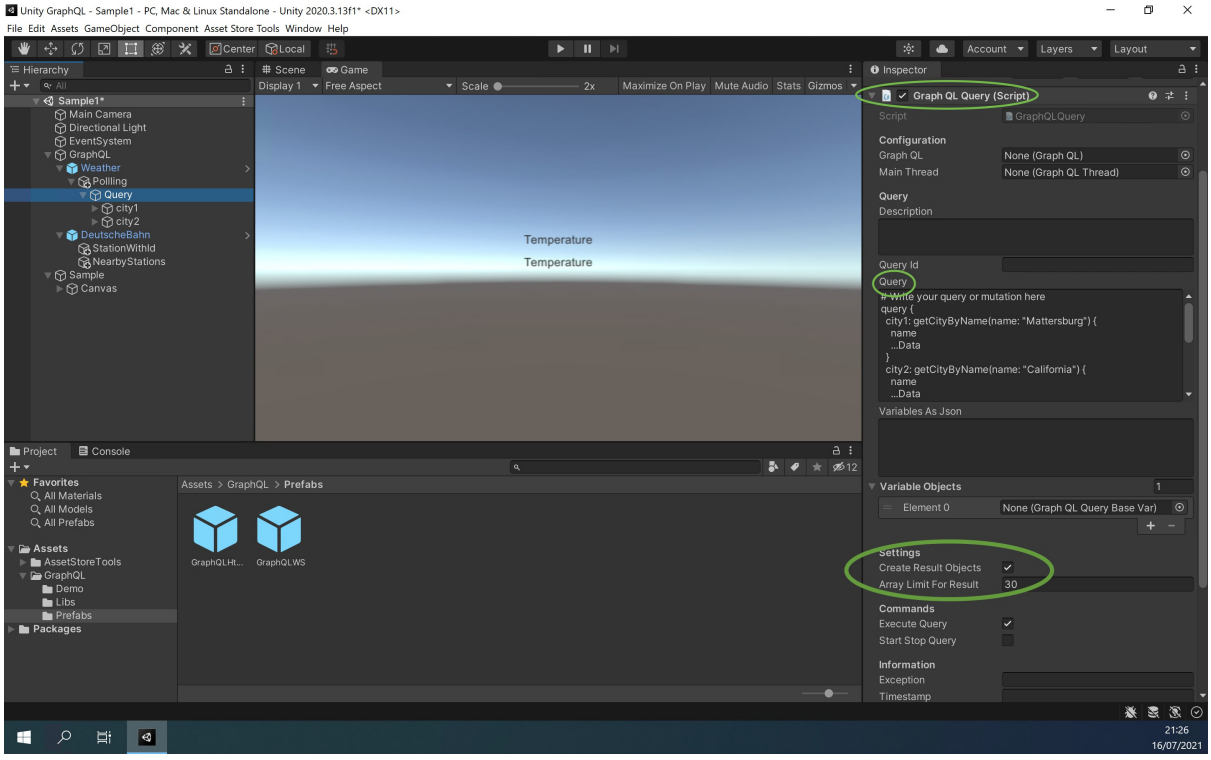


2.2 Query

Add an empty GameObject as a child of your previously created “Connection” GameObject to the scene and add the "GraphQL Query" script to it.

Then you must put your GraphQL Query into the Query field. You can pass a Query Id, but if you don’t pass it, the system will generate a unique id. The query description does not have any impact on the program, it is just for you and your documentation.

There is also a setting "Create Result Objects", if that option is set, the system will generate Unity GameObjects for the JSON Result of the query. You can start the scene, execute the query, copy the result objects, stop the scene and paste it below the query. So that you can use it now in your editor and code. When you start the scene and execute the query again, it will not create new objects, it will just update the values on the result objects.

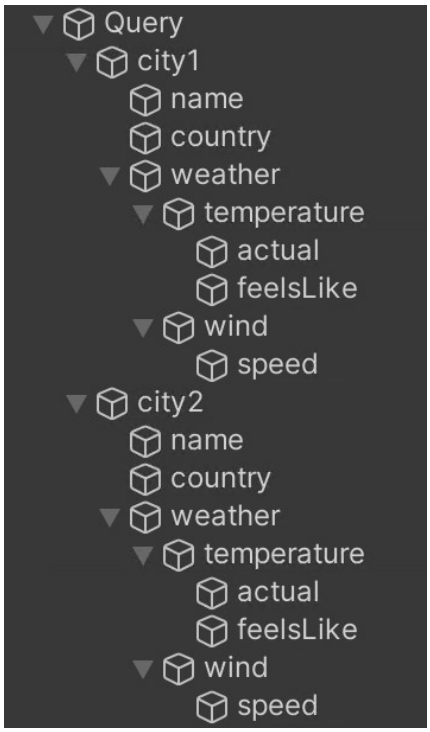


2.3 Example Query and Result Objects

Here we have an example query and how the result objects will look when you execute the query. You see how the query description matches the query result. One of the benefits of GraphQL "Get exactly that what you have asked for".

```
query {
  city1: getCityByName(name: "Mattersburg") {
    name
    ...Data
  }
  city2: getCityByName(name: "California") {
    name
    ...Data
  }
}

fragment Data on City {
  country
  weather {
    temperature {
      actual
      feelsLike
    }
    wind {
      speed
    }
  }
}
```

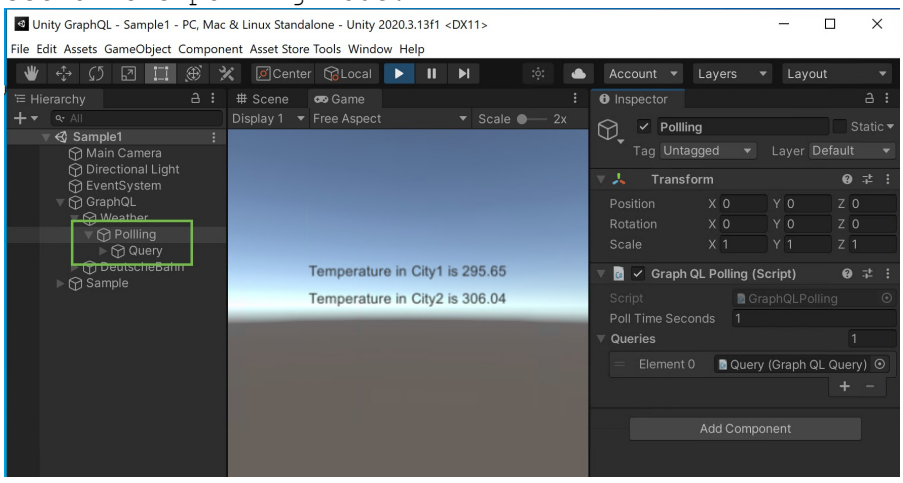


2.4 Variables

You can also use GraphQL Variables in your queries. You can set the values of the variables in the "Variables As Json" Field like this: {"Arg1": "value1"}. Or you can use the field "Variable Objects", there you can add a list of variables and you can drag and drop a GameObject of with a component "Graph QL Query Float Var" or "… Text Var" on it.

2.5 Polling

You can create a new empty GameObject and add the "GraphQL Polling" script to it and drag the previously pasted query nodes below this polling node. The queries will then be executed based on the interval set on the polling node.



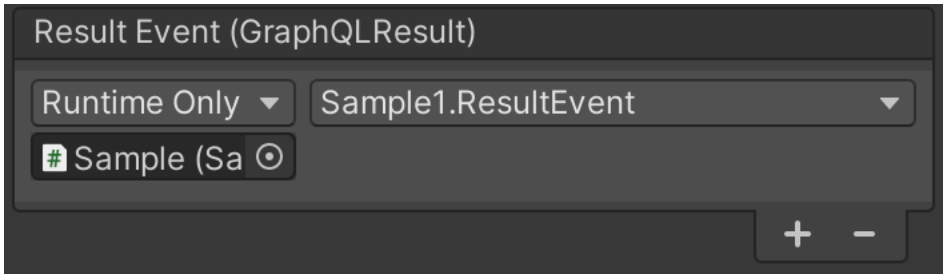
2.6 Result Event

On the GraphQL Query GamesObject there is a "Result Event" property. This is a UnityEvent<GraphQLResult>. You can set here one ore more functions which will be called every time when the query delivers a result. The query result of type GraphQLResult is passed as an argument to the function.

Example:

```
public class Sample1 : MonoBehaviour
{
    public void ResultEvent(GraphQLResult result)
    {
        Debug.Log("Data is here! "+result.Data.ToString());
    }
}
```

Create a GameObject with "Sample1" as component and drag and drop it the "Result Event" property of your GraphQLQuery and choose the function "ResultEvent".



2.7 Native JavaScript WebSocket Support

If you need a WebSocket connection in a WebGL build, then you must use the Script "GraphQLWebsocketWebGL.cs", which you will find in the Scripts directory. It will use a native JavaScript implementation for the WebSocket connection. See also the Sample2-WebGL demo scene in the scene's directory.
Limitation: By using the native WebSocket feature it is currently not possible to pass Headers.

2.8 AWS AppSync WebSocket Protocol

The package can now also handle the AWS AppSync WebSocket Protocol, which is different to the Apollo Protocol. You can now subscribe to real-time value changes via a GraphQL WebSocket connection with AWS AppSync.
The GraphQL WebSocket GameObject now has two options which you must set. One is a Checkbox to enable AWS AppSync WebSocket protocol and the second one "Aws App Sync Auth" is a string where you must put in the authentication string for your AWS AppSync service, which is a JSON

string, and depends on the authentication methos which you use in your AWS AppSync service.

Example for a token-based authentication:

Url: wss:// xxxxxxxxxxxxxxx.appsync-realtime-api.eu-central-1.amazonaws.com/graphql

Auth-String: { "host": "xxxxxxxxxxxxxxx.appsync-api.eu-central-1.amazonaws.com", "x-api-key": "zzz-zzzzzzzzzzzzzzzzz" }