API Query Manager

Quick Start

1. Package Introduction
3. How to Use
4. API Settings
5. QueryName
6. APIQuery
7. API Manager
Variables
Functions
8. Conclusion
9. Support

1. Package Introduction

"APIQueryMaster" is a versatile and generic API system designed to streamline the creation, processing, and data retrieval from various APIs. The system is robust, offering a range of features that enhance its functionality and adaptability for different API types. Below are the key features of the APIQueryMaster system:

1. **Universal API Processing**: Capable of handling any type of API, providing a flexible solution for diverse data retrieval needs.
2. **Query Management**: Simplifies the creation and management of API queries, ensuring efficient processing.
3. **Retry Mechanism**: Supports multiple retries for API calls, enhancing reliability and ensuring data retrieval even in case of initial failures.
4. **Concurrent API Processing**: Allows multiple API requests to be processed simultaneously, optimizing performance and reducing wait times.
5. **Customizable Headers**: Enables the addition of various headers to API requests, providing the necessary flexibility to meet specific API requirements.
6. **Priority Handling**: Facilitates changing the priorities of API processing, ensuring critical queries are handled promptly.
7. **Flexible Query Parameters**: Supports APIs with query parameters, with or without names included in the parameters, accommodating a wide range of API designs.
8. **Form Data Handling**: Capable of processing APIs that require form data, making it suitable for diverse application scenarios.
9. **Custom API Requests**: Allows for sending custom API requests using the same system, offering a unified approach to API management.

The project hierarchy of APIQueryMaster reflects its modular and organized structure, designed to facilitate ease of use and maintainability.

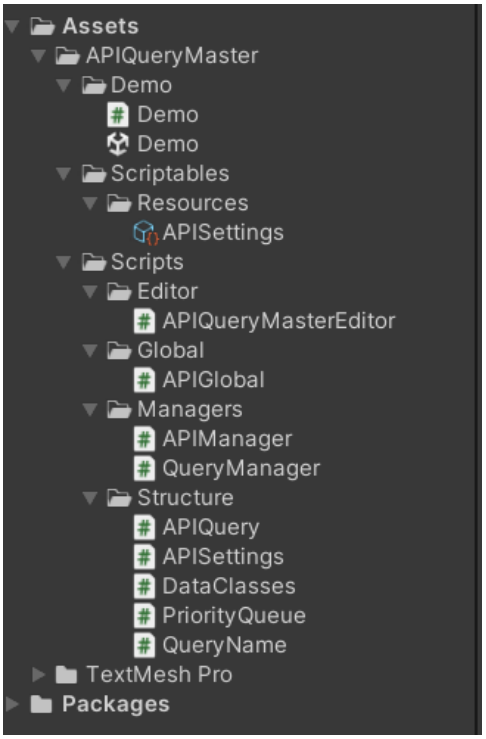
2. Package Hirerichy

The project hierarchy presented is for the "APIQueryMaster" module within a Unity project. At the top level, it includes several key directories: `Demo`, `Scriptables`, and `Scripts`.

1. **Demo**: Contains example scenes and `Demo.cs` script to demonstrate the usage of the APIQueryMaster system.
2. **Scriptables**: Houses scriptable objects that are pivotal for the project, including `APISettings.asset`, which can be accessed in code `APIGlobal.Settings` anywhere make it easily accessible for runtime default configuration.
3. **Scripts**: This directory is organized into multiple subfolders:
    - **Editor**: Contains editor-specific scripts like `APIQueryMasterEditor` to add MenuItem `APIQueryMaster/APISettings` for easy accessibility and selection of the `APISettings.asset`.
    - **Global**: Includes global scripts like `APIGlobal` that hold a global definition of `APISettings` object as settings and utilities used across the module and where every namespace is added.
    - **Managers**: Hosts manager scripts like `APIManager` and `QueryManager` responsible for handling API queries and managing their flow.

- **Structure**: Contains essential data structures such as `APIQuery`, `APISettings`, `PriorityQueue`, `QueryName`, and `DataClasses`, which are used to define and manage the data and queries within the system.
    i. `APIQuery` is the main data class used to take data from `QueryManager` and process it in the `APIManager`.
    ii. `APISettings` is a scriptable class of the `APISettings.asset`.
    iii. `PriorityQueue` is a data class used for the queue system with priority management.
    iv. `QueryName` is an enum where you can add query name to add their template data in API settings and can fetch them just using this enum in the code.

Additionally, the hierarchy includes a reference to `TextMeshPro`, indicating that the project utilizes this package for demo scene presentation.



3. How to Use

Here is the `APIExample` script that demonstrates how to utilize the `APIQuery` class within the APIQueryMaster system to set up and execute an API request in Unity. This script showcases initializing an API query, configuring its parameters and settings, and defining callbacks for handling the response.

```
using APIQueryMaster.Structure;
using UnityEngine;

public class APIExample : MonoBehaviour
{
    void Start()
```

```
    {
        // Initialize APIQuery with default settings
        APIQuery query = new APIQuery(QueryName.SomeQueryName);

        // Set query parameters
        query.SetQueryData(new { param1 = "value1", param2 = "value2" });

        // Set form data
        query.SetFormData(new { field1 = "data1", field2 = "data2" });

        // Add custom header
        query.AddHeader("Authorization", "Bearer token");

        // Set callbacks
        query.SetSuccessCallBack(OnSuccess);
        query.SetFailCallBack(OnFailure);

        // Set timeout and retries
        query.SetTimeOut(30);
        query.SetRetries(3);

        // Finally, Enqueue the query to the API manager send the request)
        APIManager.EnqueueQuery(query);
    }

    void OnSuccess(string response)
    {
        Debug.Log("Success: " + response);
    }

    void OnFailure(string error)
    {
        Debug.LogError("Error: " + error);
    }
}
```

Script Explanation:

- **Initialization**:

An instance of `APIQuery` is created using a predefined query name (`QueryName.SomeQueryName`). This initializes the query with default settings from the `APISettings.asset` configuration. Another constructor with parameters is (`QueryType queryType, string URL, string endPoint = "", int priority = 0`) can be used to process a separate API that is out of the main system.

- **Setting Query Parameters**:

```
query.SetQueryData(new { param1 = "value1", param2 = "value2" });
```

or

```
DummySendData sendData = new DummySendData{ param1 = "value1", param2 =
"value2" };
query.SetQueryData(sendData, false);
```

To set query data to pass your data to set parameters in the query. There is an option parameter of `IncludeName` which has a default value of true.

```
https://dummy.restapiexample.com/endpoint?param1=value1&param1=value2
```

or

```
https://dummy.restapiexample.com/endpoint/value1/value2
```

- **Setting Form Data**:

```
query.SetFormData(new { param1 = "value1", param2 = "value2" });
```

or

```
DummySendData sendData = new DummySendData{ param1 = "value1", param2 =
"value2" };
query.SetFormData(sendData);
```

To send data in the body of API requests use `SetFormData` It will get serialized into JSON.

- **Adding Custom Header**:

```
query.AddHeader("Authorization", "Bearer token");
```

To add a custom header use the AddHeader function. The header name is "`Authorization`" and its value is "`Bearer token`". Set all headers if you are creating a custom API Query request.

- **Setting Callbacks**:

```
query.SetSuccessCallBack(OnSuccess);
```

Registers the `OnSuccess` method as the callback to be executed upon successful completion of the API request.

```
query.SetFailCallBack(OnFailure);
```

Registers the `OnFailure` method as the callback to be executed if the API request fails.

- **Setting Timeout and Retries**:

```
query.SetTimeOut(30);
```

Sets the timeout for the API request to 30 seconds.

```
query.SetRetries(3);
```

Configures the query to retry up to 3 times in case of failure.

- **Executing the Query**:

Now the Query will be enqueued to the APIManager priority queue using `APIManager.EnqueueQuery(query);`.

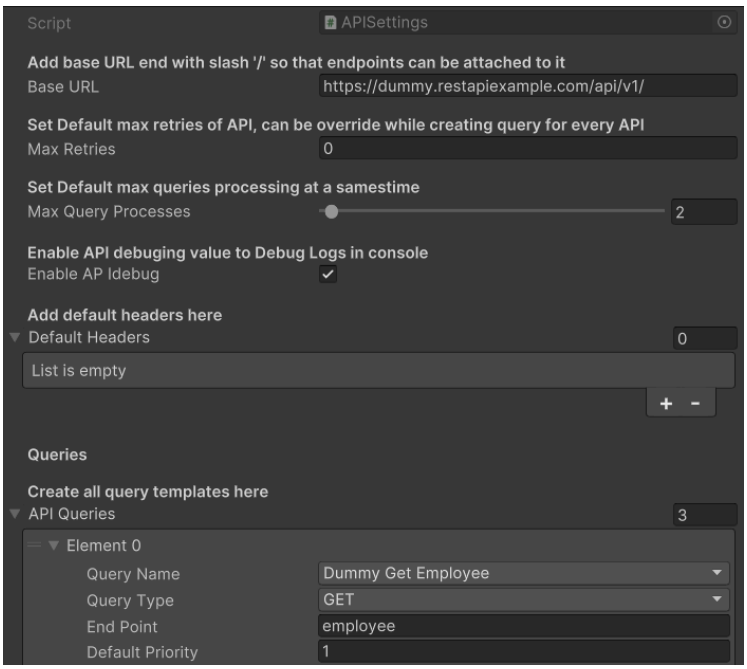
Callback Methods:

- **OnSuccess**:
  - This method is called when the API request is successful. It logs the response to the console.
- **OnFailure**:
  - This method is called when the API request fails. It logs the error message to the console.

4. API Settings

The API Settings Inspector for the APIQueryMaster system provides a user-friendly interface for configuring various parameters essential for API management.

- It allows you to set the base URL (`Base URL`), ensuring that all endpoints can be attached correctly.
- You can define the default maximum retries (`Max Retries`) for API calls, which can be overridden for specific queries.
- The `Max Query Processes` setting lets you control the number of API queries processed simultaneously.
- Enabling `API Idebug` allows debugging information to be logged in the console.
- Additionally, you can specify default headers to be included in all API requests.
- Create query templates under `API Queries` for streamlined query management.



5. QueryName

`QueryName` is a simple enum but an essential part of the system used to identify and access `API Queries` templates by using it. It can be string but to avoid string matching issues, enum is best practice. You have go to the `QueryName` script to add APIs names. It is hassle but it worth time.

6. APIQuery

The `APIQuery` class is a core component of the APIQueryMaster system, responsible for organizing and managing the data sent and received during API interactions. This class provides a structured way to define API queries, handle query parameters, set callbacks for success and failure, and manage retries and timeouts. It facilitates the creation of flexible and robust API requests by encapsulating various aspects of query configuration and execution.

Variables:

- **`queryName`**: Name of the query to be added in the Query Name Script.
- **`queryType`**: Specifies the type of HTTP method (GET, POST, PUT, etc.).
- **`endPoint`**: Endpoint for the API request, excluding the starting slash.
- **`defaultPriority`**: Default priority for query execution, in ascending order.
- **`BaseURL`**: Base URL for the API request.
- **`IsSendingDataByQuery`**: Indicates if data is being sent via query parameters.
- **`IncludeNameForQuery`**: Indicates if parameter names should be included in the query.
- **`QueryData`**: Data to be sent as query parameters.
- **`IsSendingDataByForm`**: Indicates if data is being sent as form data.
- **`FormData`**: Data to be sent in the form/body of the request.
- **`RecieveDataCallback`**: Callback function for successful data reception.
- **`OnFailedCallback`**: Callback function for handling failed requests.
- **`Headers`**: List of headers to be included in the API request.
- **`Retries`**: Number of retries attempted for the query.
- **`AvailableRetries`**: Maximum number of retries allowed.
- **`Priority`**: Priority of the query.
- **`Timeout`**: Timeout value for the query.
- **`IsTimeoutSet`**: Indicates if a timeout is set for the query.

Functions:

- **`APIQuery(QueryName queryName)`**: Initializes an API query with default values.
- **`APIQuery(QueryType queryType, string URL, string endPoint, int priority)`**: Initializes an API query with custom values.
- **`GetCompleteURL()`**: Returns the complete URL with base URL and endpoint, including query parameters if applicable.
- **`SetQueryData(object QuerySendData, bool IncludeName = true)`**: Sets the data to be sent as query parameters.
- **`SetFormData(object FormSendData)`**: Sets the data to be sent in the form/body of the request.
- **`AddHeader(string headerName, string headerValue)`**: Adds a header to the API request.
- **`SetSuccessCallBack(Action<string> recieveDataCallback)`**: Registers a callback for successful data reception.

- `SetFailCallBack(Action<string> failedCallback)`: Registers a callback for handling failed requests.
- `SetTimeOut(int timeout)`: Sets the timeout value for the query.
- `SetRetries(int retries)`: Sets the maximum number of retries for the query.
- `IncrementRetries()`: Increments the retry count.
- `BuildQueryString()`: Builds the query string from the query data.

7. API Manager

The `APIManager` class is responsible for handling API calls in Unity, using a priority queue system to manage and process requests efficiently. It supports callbacks for both successful responses and failures, allowing for robust and flexible API communication.

Variables

- `currentQueryProcesses`: Tracks the number of queries currently being processed.
- `queryQueue`: A priority queue for managing API queries.
- `manager`: A reference to a `MonoBehaviour` for starting coroutines.

Functions

- `APIManager(MonoBehaviour manager)`: Constructor that initializes the API manager with a given `MonoBehaviour`.
- `void ClearQueue()`: Clears the priority queue, discarding all remaining API queries.
- `void EnqueueQuery(APIQuery query)`: Enqueues an API query into the priority queue for processing.
- `void ProcessNextQuery()`: Processes the next query in the priority queue.
- `void OnResponse(APIQuery query, string downloadedData)`: Handles the response of a processed query and invokes the appropriate callback.
- `IEnumerator SendRequest(APIQuery APIQuery, Action<APIQuery, string> actionToInvoke)`: Sends an API request, processes the response, and manages retries if necessary.

8. Conclusion

- The APIQueryMaster system provides a robust and flexible solution for managing API interactions within your Unity projects. By leveraging this system, developers can easily create, configure, and execute API queries with a high degree of customization. The APIQuery class, along with its supporting components, allows for efficient handling of query parameters, form data, headers, retries, and timeouts. This streamlined approach simplifies the process of integrating various APIs into your application, ensuring reliable and efficient data communication.

9. Support

If you have any questions or require further assistance with the APIQueryMaster system, please do not hesitate to contact our support team. I am here to help you make the most of this powerful tool and ensure your development experience is smooth and productive. Please also provide any feedback to improve this.

Support Contact Information:

- **Email**: ahsan.ijaz.official@gmail.com
- **LinkedIn**
- **Github**
- **Website**