REST API PLUGIN

This plugin will help you to call REST API easily even if you do not need to know the low level networking codes.

Features

- GET
- POST
- PUT
- DELETE
- Can pass Header, Body and Form data
- Possible to call Authentication API using Bearer Token.
- Call API Asynchronously. So no performance issue.
- Support callbacks.
- Can set the TimeOut variable (in ms) for api calling.
- Easy to Use

Documentation -

Basic idea About API Calling -

To make an API call you have to understand and do the following things.

1. First see the API documentation to understand the API type (POST/GET/PUT/DELETE) , API request parameter types (header/body/form) and API response (how API returns the result).

2. Then you have to create the request class and response class.

Request Class – First please see the API documentation to find out which data you need for that API calling. Generally request data are submitted through form, header and body.

Body - Body data mainly passes through raw json. So you just need to make the json data from that class object. For example please see the ToBody function of PutPostRequestModel class from ApiDataModels.cs script.

Header - Header is a dictionary type data. Please check the ToCustomHeader function of PutPostRequestModel class from ApiDataModels.cs script.

Form – This is WWWForm type data.Please check the ToFormData function of PutPostRequestModel class from ApiDataModels.cs script.

For better understanding please check here

Response Class – It is easier than creating a request class. You just need to create a POJO class from the response string. There are many tools available online for that.

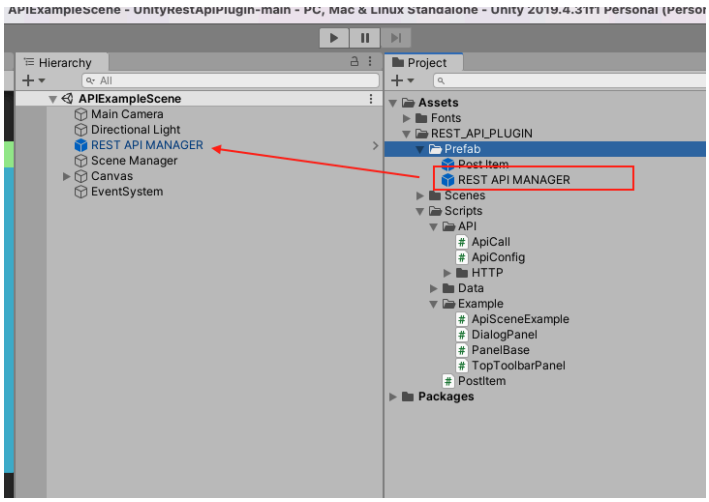
I personally use this website for creating POJO response classes. Just copy the api response from API

document or some API Client (like Postman) and then paste it and this website will create a C# Pojo class.
For this plugin please check Use Field option from setting and uncheck other options.

You get the primary idea now let's jump into the main part about API calling.

API CALLING-

Project Setup - Drag the REST API MANAGER from project hierarchy to game scene. Bingo! Your project is ready for api call now.



Get API Call -

```
ApiCall.instance.GetRequest<RESPONSE MODEL>( Url, Headers,  (result => {
// API Call Success. Do whatever
}),
(error => {
// Handle Error
}), JsonPrefix);
```

Here,

Url - is a string type data. Please put the API url here.

Header - It is a dictionary type data.

result - This block will execute only when the API call is successful. Result type is the response model. You have to write the API success logic in this blocks.

error - This block will execute only when the API call is failed. Error is a string type variable. You have to write the API failure logic in this block.

JsonPrefix - This is a string type value. It may be null or need to put value here depending on API response JSON.

If the response starts with [then you have to put some JsonPrefix value. Please remember that

Json Prefix would be the root of the Response Class so it would be mandatory to keep the same naming of JsonPrefix value and response class root variable name.

If response Json is not started with [then you do not need to add Json Prefix Value and make it nul.

For more Understanding please see the GetPosts function from ApiSceneExample.cs script.

POST API Call -

```
ApiCall.instance.PostRequest<ResponseModel>(Url, Header, Form, Body , (result =>
{
// API Call Success. Do whatever
}), (error =>
{
// Handle Error
}), JsonPrefix);
```

Here,

URL, Header, result, error, JsonPrefix - Please see the above Get API documentation.
Body - it is string type data.

For more Understanding please see the AddPost function from ApiSceneExample.cs script.

PUT API Call -

```
ApiCall.instance.PutRequest<ResponseModel>(Url, Header , Form ,Body ,(result =>
{
}), (error =>
{
}), JsonPrefix);
```

Here,

URL, Header, result, error, JsonPrefix, Body - Please see the above Post API documentation.

For more Understanding please see the PutPost function from ApiSceneExample.cs script.

DELETE API Call -

```
ApiCall.instance.DeleteRequest<ResponseModel>(Url, Header ,(result =>
{
}), (error =>
{
}), JsonPrefix);
```

Here,

URL, Header, result, error, JsonPrefix - Please see the above PUT API documentation.

For more Understanding please see the DeletePost function from ApiSceneExample.cs script.