RestClient for Unity 🤘

This **HTTP/REST** Client is based on Promises to avoid the Callback Hell ☠️ and the Pyramid of doom💩 working with **Coroutines** in **Unity** 🎮, example:

```
var api = "https://jsonplaceholder.typicode.com";
RestClient.GetArray<Post>(api + "/posts", (err, res) => {
  RestClient.GetArray<Todo>(api + "/todos", (errTodos, resTodos) => {
  RestClient.GetArray<User>(api + "/users", (errUsers, resUsers) => {
      //Missing validations to catch errors!
    });
  });
});
```

But working with **Promises** we can improve our code, yay! 👏

```
RestClient.GetArray<Post>(api + "/posts").Then(response => {
  EditorUtility.DisplayDialog("Success", JsonHelper.ArrayToJson<Post>(response, true),
"Ok");
  return RestClient.GetArray<Todo>(api + "/todos");
}).Then(response => {
  EditorUtility.DisplayDialog("Success", JsonHelper.ArrayToJson<Todo>(response, true),
"Ok");
  return RestClient.GetArray<User>(api + "/users");
}).Then(response => {
  EditorUtility.DisplayDialog("Success", JsonHelper.ArrayToJson<User>(response, true),
"Ok");
}).Catch(err => EditorUtility.DisplayDialog ("Error", err.Message, "Ok"));
```

Features 🎮

- Works out of the box 🎉
- Supports **HTTPS/SSL**
- Built on top of **UnityWebRequest** system
- Includes JSON serialization with **JsonUtility** (Other tools are supported!)
- Get **Arrays** Supported
- Default **HTTP** Methods **(GET, POST, PUT, DELETE, HEAD)**
- Generic **REQUEST** method to create any http request
- Based on **Promises** for a better asynchronous programming
- Handle HTTP exceptions in a better way

- Retry HTTP requests easily
- Open Source 🦄

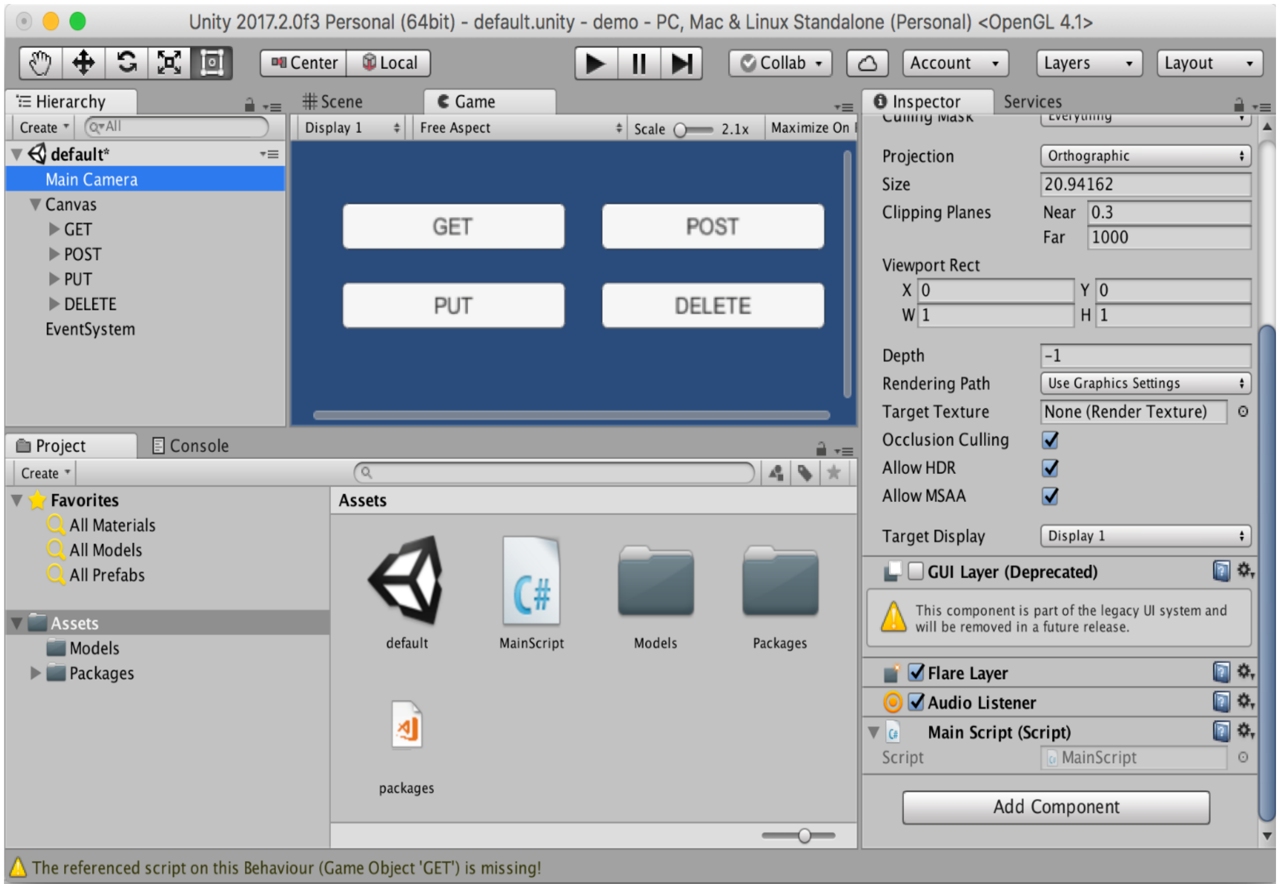
Supported platforms 📱 🖥️

The [UnityWebRequest]() system supports most Unity platforms:

- All versions of the Editor and Standalone players
- WebGL
- Mobile platforms: iOS, Android
- Universal Windows Platform ([RSG.Promise_standard.dll]() is required)
- PS4 and PSVita
- XboxOne
- HoloLens
- Nintendo Switch

Demo ⏯️

Do you want to see this beautiful package in action? Download the demo [here]()



Projection: Orthographic
Size: 20.94162
Clipping Planes: Near 0.3, Far 1000
Viewport Rect: X 0, Y 0, W 1, H 1
Depth: -1
Rendering Path: Use Graphics Settings
Target Texture: None (Render Texture)
Occlusion Culling: ✓
Allow HDR: ✓
Allow MSAA: ✓
Target Display: Display 1
GUI Layer (Deprecated)
This component is part of the legacy UI system and will be removed in a future release.
Flare Layer ✓
Audio Listener ✓
Main Script (Script)
Script: MainScript

The referenced script on this Behaviour (Game Object 'GET') is missing!

Installation 👨‍💻

Unity package

Download and install the **.unitypackage** file of the latest release published here.

Nuget package

Other option is downloading this package from **NuGet** with **Visual Studio** or using the **nuget-cli**, a **NuGet.config** file is required at the root of your **Unity Project**, for example:

```xml
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <config>
    <add key="repositoryPath" value="./Assets/Packages" />
  </config>
</configuration>
```

The package to search for is **Proyecto26.RestClient**.

Getting Started 📚

The default methods **(GET, POST, PUT, DELETE, HEAD)** are:

```csharp
RestClient.Get("https://jsonplaceholder.typicode.com/posts/1").Then(res => {
  EditorUtility.DisplayDialog("Response", res.Text, "Ok");
});
RestClient.Post("https://jsonplaceholder.typicode.com/posts", newPost).Then(res => {
  EditorUtility.DisplayDialog("Status", res.StatusCode.ToString(), "Ok");
});
RestClient.Put("https://jsonplaceholder.typicode.com/posts/1", updatedPost).Then(res => {
  EditorUtility.DisplayDialog("Status", res.StatusCode.ToString(), "Ok");
});
RestClient.Delete("https://jsonplaceholder.typicode.com/posts/1").Then(res => {
  EditorUtility.DisplayDialog("Status", res.StatusCode.ToString(), "Ok");
});
RestClient.Head("https://jsonplaceholder.typicode.com/posts").Then(res => {
  EditorUtility.DisplayDialog("Status", res.StatusCode.ToString(), "Ok");
});
```

Generic Request Method

And we have a generic method to create any type of request:

```
RestClient.Request(new RequestHelper {
  Uri = "https://jsonplaceholder.typicode.com/post",
  Method = "POST",
  Timeout = 10,
 Params = new Dictionary<string, string> {
    { "param1", “Query string param..." }
  },
  Headers = new Dictionary<string, string> {
    { "Authorization", "Bearer JWT_token..." }
  },
  Body = newPhoto, //Serialize object using JsonUtility by default
  BodyString = SerializeObject(newPhoto), //Use it instead of 'Body' to serialize
using other tools
  BodyRaw = CompressToRawData(newPhoto), //Use it instead of 'Body' to send raw data
directly
  FormData = new WWWForm(), //Send files, etc with POST requests
  SimpleForm = new Dictionary<string, string> {}, //Content-Type: application/x-www-
form-urlencoded
  FormSections = new List<IMultipartFormSection>() {}, //Content-Type: multipart/form-
data
  CertificateHandler = new CustomCertificateHandler(),
  UploadHandler = new UploadHandlerRaw(bytes), //Send bytes directly if it's required
  DownloadHandler = new DownloadHandlerFile(destPah), //Download large files
  ContentType = "application/json", //JSON is used by default
  Retries = 3, //Number of retries
  RetrySecondsDelay = 2, //Seconds of delay to make a retry
  RetryCallback = (err, retries) => {}, //See the error before retrying the request
  EnableDebug = true, //See logs of the requests for debug mode
  IgnoreHttpException = true, //Prevent to catch http exceptions
  ChunkedTransfer = false,
  UseHttpContinue = true,
  RedirectLimit = 32,
  DefaultContentType = false, //Disable JSON content type by default
  ParseResponseBody = false //Don't encode and parse downloaded data as JSON
}).Then(response => {
  //Get resources via downloadHandler to get more control!
  Texture texture = ((DownloadHandlerTexture)response.Request.downloadHandler).textu-
re;
  AudioClip audioClip =
((DownloadHandlerAudioClip)response.Request.downloadHandler).audioClip;
  AssetBundle assetBundle = ((DownloadHandlerAssetBundle)response.Request.download-
Handler).assetBundle;

  EditorUtility.DisplayDialog("Status", response.StatusCode.ToString(), "Ok");

});
```

- Example downloading an audio file:

```
var fileUrl = "https://bit.ly/2ZUpqTc";
var fileType = AudioType.OGGVORBIS;
RestClient.Get(new RequestHelper {
  Uri = fileUrl,
  DownloadHandler = new DownloadHandlerAudioClip(fileUrl, fileType),
}).Then(res => {
  AudioSource audio = GetComponent<AudioSource>();
  audio.clip = ((DownloadHandlerAudioClip)res.Request.downloadHandler).audioClip;
  audio.Play();
});
```

With all the methods we have the possibility to indicate the type of response, in the following example we're going to create a class and the **HTTP** requests to load **JSON** data easily:

```
[Serializable]
public class User
{
  public int id;
  public string name;
  public string username;
  public string email;
  public string phone;
  public string website;
}
```

- **GET JSON**

```
var usersRoute = "https://jsonplaceholder.typicode.com/users";
RestClient.Get<User>(usersRoute + "/1").Then(firstUser => {
  EditorUtility.DisplayDialog("JSON", JsonUtility.ToJson(firstUser, true), "Ok");
});
```

- **GET Array (JsonHelper is an extension to manage arrays)**

```
RestClient.GetArray<User>(usersRoute).Then(users => {
  EditorUtility.DisplayDialog("Array", JsonHelper.ArrayToJsonString<User>(users,
true), "Ok");
});
```

Also, we can create different classes for custom responses:

```
[Serializable]
public class CustomResponse
{
  public int id;
}
```

- **POST**

```
RestClient.Post<CustomResponse>(usersRoute, newUser).Then(customResponse => {
  EditorUtility.DisplayDialog("JSON", JsonUtility.ToJson(customResponse, true), "Ok");
});
```

- **PUT**

```
RestClient.Put<CustomResponse>(usersRoute + "/1", updatedUser).Then(customResponse =>
{
  EditorUtility.DisplayDialog("JSON", JsonUtility.ToJson(customResponse, true), "Ok");
});
```

Custom HTTP Headers, Params and Options 💥

HTTP Headers, such as `Authorization`, can be set in the **DefaultRequestHeaders** object for all requests

```
RestClient.DefaultRequestHeaders["Authorization"] = "Bearer ...";
```

Query string params can be set in the **DefaultRequestParams** object for all requests

```
RestClient.DefaultRequestParams["param1"] = “Query string value...";
```

Also we can add specific options and override default headers and params for a request

```
var currentRequest = new RequestHelper {
  Uri = "https://jsonplaceholder.typicode.com/photos",
  Headers = new Dictionary<string, string> {
    { "Authorization", "Other token..." }
  },
  Params = new Dictionary<string, string> {
    { "param1", "Other value..." }
  }
};
RestClient.GetArray<Photo>(currentRequest).Then(response => {
  EditorUtility.DisplayDialog("Header", currentRequest.GetHeader("Authorization"),
"Ok");
});
```

And we can know the status of the request and cancel it!

```
currentRequest.UploadProgress; //The progress by uploading data to the server
currentRequest.UploadedBytes; //The number of bytes of body data the system has uploa-
ded
currentRequest.DownloadProgress; //The progress by downloading data from the server
currentRequest.DownloadedBytes; //The number of bytes of body data the system has
downloaded
currentRequest.Abort(); //Abort the request manually
```

Later we can clean the default headers and params for all requests

```
RestClient.CleanDefaultHeaders();

RestClient.CleanDefaultParams();
```

Example

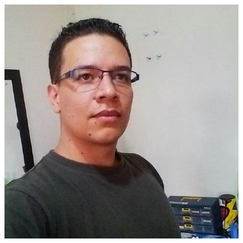
- Unity as Client

```
[Serializable]
public class ServerResponse {
  public string id;
  public string date; //DateTime is not supported by JsonUtility
}
[Serializable]
public class User {
  public string firstName;
  public string lastName;
}
RestClient.Post<ServerResponse>("www.api.com/endpoint", new User {
  firstName = "Juan David",
  lastName = "Nicholls Cardona"
}).Then(response => {
  EditorUtility.DisplayDialog("ID: ", response.id, "Ok");
  EditorUtility.DisplayDialog("Date: ", response.date, "Ok");
});
```

- NodeJS as Backend (Using Express)

```
router.post('/', function(req, res) {
  console.log(req.body.firstName)
  res.json({
    id: 123,
    date: new Date()
  })
});
```

Collaborators 🥇

			
Juan Nicholls	Diego Ossa	Nasdull	

Credits 👍

- **Promises library for C#:** Real Serious Games/C-Sharp-Promise

Supporting 🍻

I believe in Unicorns 🦄 Support me, if you do too.

Happy coding 💯

Made with ❤️