RestClient for Unity

> Supported Unity versions 2017.2 or higher

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

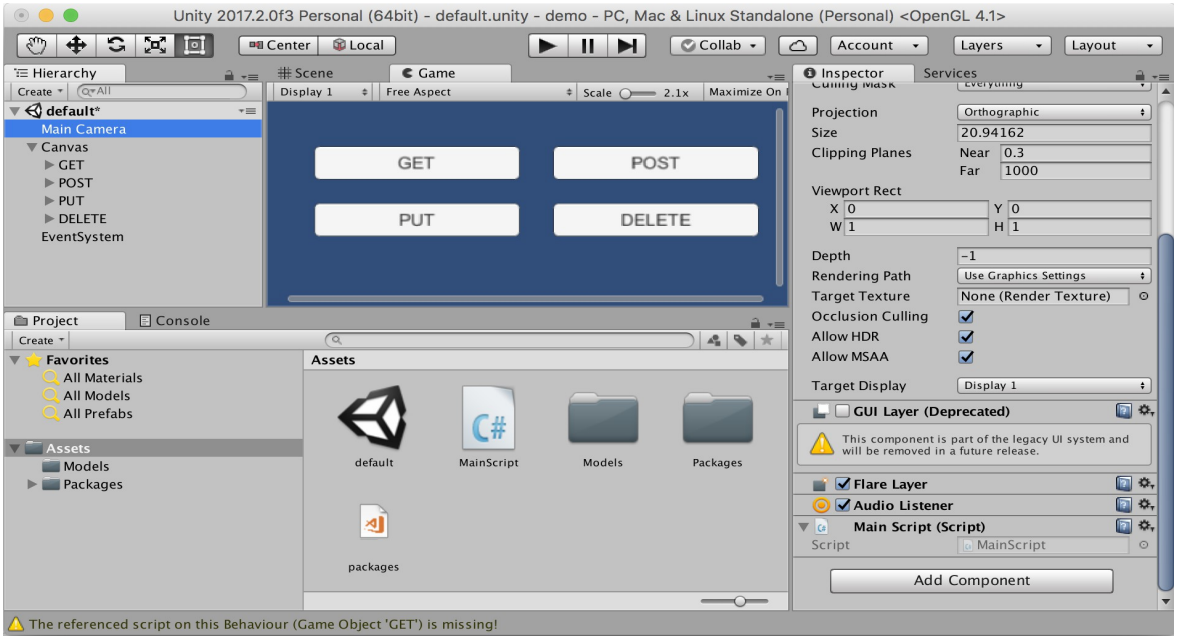

```
var api = "https://jsonplaceholder.typicode.com";
RestClient.GetArray<Post>(api + "/posts", (err, res) => {
   RestClient.GetArray<Todo>(api + "/todos", (errTodos, resTodos) => {
      RestClient.GetArray<User>(api + "/users", (errUsers, resUsers) => {
         //Missing validations to catch errors!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

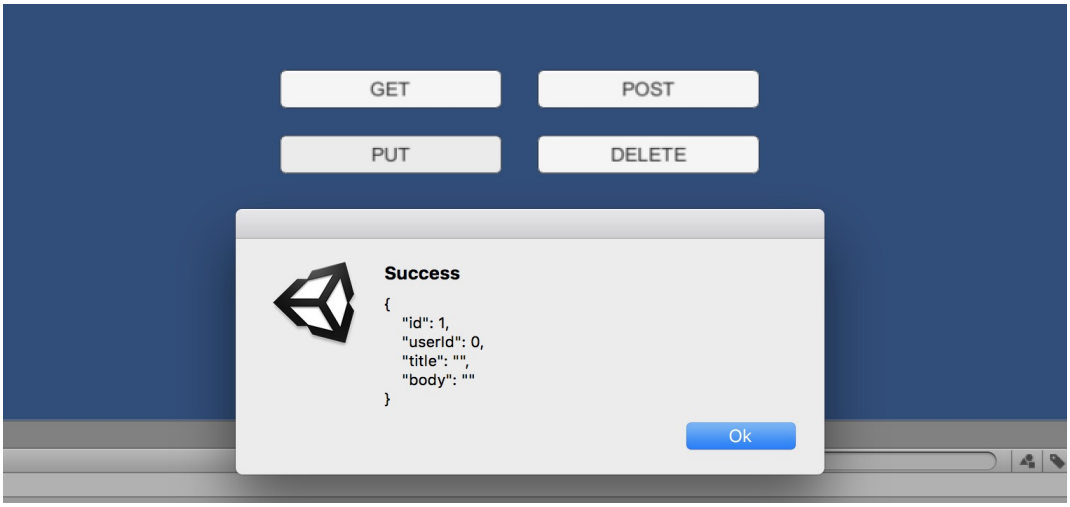
But working with **Promises** we can improve our code, yay!

```
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"));
```

Demo

Do you want to see this beautiful package in action? Download the demo here





Installation

Unity package

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

Nuget package

Other option is download 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 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)** are:

```
RestClient.Get("https://jsonplaceholder.typicode.com/posts/1").Then(response => {
   EditorUtility.DisplayDialog("Response", response.text, "Ok");
})

RestClient.Post("https://jsonplaceholder.typicode.com/posts", newPost).Then(response => {
   EditorUtility.DisplayDialog("Status", response.statusCode.ToString(), "Ok");
})

RestClient.Put("https://jsonplaceholder.typicode.com/posts/1", updatedPost).Then(response => {
   EditorUtility.DisplayDialog("Status", response.statusCode.ToString(), "Ok");
```

```
})


RestClient.Delete("https://jsonplaceholder.typicode.com/posts/1").Then(response => {
   EditorUtility.DisplayDialog("Status", response.statusCode.ToString(), "Ok");
})
```

But we are going to create a class **"User"** 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**

```
RestClient.GetArray<User>(usersRoute).Then(allUsers => {
   EditorUtility.DisplayDialog("JSON Array", JsonHelper.ArrayToJsonString<User>(allUsers, 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 and Options

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

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

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

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

And later we can clean the default headers for all requests

```
RestClient.CleanDefaultHeaders();
```

Collaborators

Diego Ossa

Juan Nicholls

Credits

- Real Serious Games/C-Sharp-Promise

Supporting

I believe in Unicorns, aupport me, if you do too.

Happy coding

Made with <3