SimpleHTTP

SimpleHTTP is a dead simple HTTP client for Unity, because HTTP shouldn't be that hard.

Installation

Download the .unitypackage file and inside Unity go to the menu `Assets -> Import New Assets...` and select the package for SimpleHTTP.

Features

Currently, SimpleHTTP supports: * HTTP and HTTPS * GET, POST, PUT and DELETE * Headers * Timeouts

All of these features with a simple and consistent API. In the future it will also support: * More validations * URL parsing * Delegates to indicate progress

Usage

SimpleHTTP is very, very, very, simple to use (really!) and it is asynchronous. You can send any type of request basically with three objects: `Request`, `Client` and `Response`. Consider that the general flow would be something like this: * Create a request * Use the client to send the request * Get the response * Profit!

To make your life even easier, SimpleHTTP support JSON serialization and deserialization, so you don't need to worry about that. To use it in your Unity scripts, you just need to add the following instruction at the top of your .cs scripts:

```
using SimpleHTTP;
```

Below are some examples that can help you understand how it works. Also you can refer to the `Examples` folder for a full working example of the library with a demo scene.

GET

```
IEnumerator Get() {
    // Create the request object
    Request request = new Request ("https://jsonplaceholder.typicode.com/posts/1");

    // Instantiate the client
    Client http = new Client ();
    // Send the request
    yield return http.Send (request);

    // Use the response if the request was successful, otherwise print an error
    if (http.IsSuccessful ()) {
        Response resp = http.Response ();
        Debug.Log("status: " + resp.Status().ToString() + "\nbody: " + resp.Body());
    } else {
        Debug.Log("error: " + http.Error());
    }
}
```

POST with JSON payload (raw)

As I mentioned before, SimpleHTTP supports JSON serialization and deserialization, so you just need to have serializable POJOs in your code. Let's say you want to fetch a post from a certain URL and your POJO looks like this:

```
[System.Serializable]
public class Post {
    private string title;
    private string body;
    private int userId;
```

```
    public Post(string title, string body, int userId) {
        this.title = title;
        this.body = body;
        this.userId = userId;
    }
}
```

Then you can send a POST to the server to create a new post.

```
IEnumerator Post() {
    // Let's say that this the object you want to create
    Post post = new Post ("Test", "This is a test", 1);

    // Create the request object and use the helper function `RequestBody` to create a body from JSON
    Request request = new Request ("https://jsonplaceholder.typicode.com/posts")
        .Post (RequestBody.From<Post> (post));

    // Instantiate the client
    Client http = new Client ();
    // Send the request
    yield return http.Send (request);

    // Use the response if the request was successful, otherwise print an error
    if (http.IsSuccessful ()) {
        Response resp = http.Response ();
        Debug.Log("status: " + resp.Status().ToString() + "\nbody: " + resp.Body());
    } else {
        Debug.Log("error: " + http.Error());
    }
}
```

SimpleHTTP will set the content type of this request automagically as “application/json”, so another thing to don’t worry about :).

POST with FormData

If the server only supports `x-www-form-urlencoded` you still can use SimpleHTTP to send your request. In this case, you just need to use the `FormData` helper to create the body of your request. Then you can send the POST as you would normally do (and SimpleHTTP will also take care of the content type for this request).

```
IEnumerator Post() {
    FormData formData = new FormData ()
        .AddField ("userId", "1")
        .AddField ("body", "Hey, another test")
        .AddField ("title", "Did I say test?");

    // Create the request object and use the helper function `RequestBody` to create a body from FormData
    Request request = new Request ("https://jsonplaceholder.typicode.com/posts")
        .Post (RequestBody.From (formData));

    // Instantiate the client
    Client http = new Client ();
    // Send the request
    yield return http.Send (request);

    // Use the response if the request was successful, otherwise print an error
    if (http.IsSuccessful ()) {
        Response resp = http.Response ();
        Debug.Log("status: " + resp.Status().ToString() + "\nbody: " + resp.Body());
    } else {
        Debug.Log("error: " + http.Error());
    }
}
```

Adding headers to your request

So, you also need to add certain headers to your request? Do not fear! SimpleHTTP also let you do that easily. Just use the `AddHeader` method in your request and you’re all set.

```
IEnumerator Get() {
    // Create the request object
    Request request = new Request ("https://jsonplaceholder.typicode.com/posts/1")
        .AddHeader ("Test-Header", "test")
        .AddHeader ("X-Fancy-Id", "some-fancy-id");

    // Instantiate the client
    Client http = new Client ();
    // Send the request
    yield return http.Send (request);

    // Use the response if the request was successful, otherwise print an error
    if (http.IsSuccessful ()) {
        Response resp = http.Response ();
        Debug.Log("status: " + resp.Status().ToString() + "\nbody: " + resp.Body());
    } else {
        Debug.Log("error: " + http.Error());
    }
}
```

PUT and DELETE

PUT requests will work exactly the same than POSTs, you just need to use `Put()` instead. And DELETEs will work similarly to GETs, just use `Delete()` for that and you're done.

License

SimpleHTTP is licensed under the Apache License, Version 2.0.

Donation

SimpleHTTP is free and open source because I think that HTTP is something fundamental and basic for games nowadays, and there are no simple and free solutions to perform basic tasks like GET or POST. However, I'm open to donations, and if you really love SimpleHTTP I'd really appreciate if you buy me a coffee to continue improving this small and simple client for the use of all of us.

https://paypal.me/satanas82