Wemakefuture Integromat v2.0 API Wrapper v3

Pass a config object to the create method.

Specify the root URL for the wrapped API with the root option. Create functions on the wrapper by passing a map of function names to path patterns for each HTTP METHOD.

```
350     // OAuth endpoint config
351     mainWrapper.oauth = create({
352         root: rootEndpoint,
353         get: {
354             authorize: '/oauth/auth/${connectionId}',
355             reauthorize: '/oauth/reauth/${connectionId}',
356             extend: '/oauth/extend/${connectionId}',
357             oauthCallback: '/oauth/cb/${connectionType}',
358             oauthCallbackId: '/oauth/cb/${connectionType}/${connectionId}'
359         }
360     });
```

Configuring the API calls is that easy. Now, wrapped API can be called like this:

```
597   let base = WeMakeFuture("api_key");
598
599   await base.oauth.authorize(
600       {
601           "connectionId": 0
602       }, {})
603       .then(res => res.json())
604       .then(json => console.log(json));
```

Method: GET, DELETE, HEAD
These maps will create functions on the returned wrapper that take **one parameter:** path argument. When those functions are called, they'll make requests with the corresponding HTTP method.

Method: POST, PATCH, PUT
These will create functions that take **two parameters:** path arguments and a request body.

Path patterns
Path patterns will be interpolated with the path arguments. These can correspond to either path variables (like **${pathVariable}**) or query params (separated by pipes like **?param1|param2|param3**).

Options

If you need to set more options for the request, for example HTTP headers, you can set these options with fetchOptions for the call in config.

```
204     // Auth endpoint config
205     mainWrapper.auth = create({
206         root: rootEndpoint,
207         post: {
208             authorize: {
209                 pathPattern: '/sso/authorize',
210                 fetchOptions: {
211                     headers: [
212                         [
213                             'content-type',
214                             'application/json'
215                         ],
216                         [
217                             'Authorization',
218                             "Token " + apiKey
219                         ]
220                     ]
221                 }
222             }
223         }
224     });
```

Default options
You can also set default options when you can use the same options for all calls.

```
// Hooks.Incoming endpoint config
mainWrapper.hooks.incomings = create({
    root: rootEndpoint,
    get: {
        list: '/hooks/${hookId}/incomings?pg[sortBy]|pg[offset]|pg[sortDir]|pg[limit]',
        detail:'/hooks/${hookId}/incomings/${incomingId}',
        stats: '/hooks/${hookId}/incomings/stats',
    },
    delete: {
        delete: '/hooks/${hookId}/incomings/'
    },
    fetchDefaults: {
        headers: [
            [
                'content-type',
                'application/json'
            ],
            [
                'Authorization',
                "Token " + apiKey
            ]
        ]
    }
});
```

Now initializing API wrapper with API Key and calling **base.hooks.incomings.list({ url & query parameters }, { request-body })** will make a request with the given parameters and headers.

```
let base = WeMakeFuture("api_key");

await base.hooks.incomings.list(
    {
        "hookId": 0,
        "pg[sortBy]": "string",
        "pg[offset]": 0,
        "pg[sortDir]": "string",
        "pg[limit]": 0
    }, {})
    .then(res => res.json())
    .then(json => console.log(json));
```

Custom options
You can also set custom options specifically to that call in 3rd parameter. This will overwrite fetchOptions and defaultOptions in config. To use customOptions in any method type of call, there should be 3 parameters and the custom options object needs to be 3rd one. For example if there is no url param & query params & body object, call should include 2 empty objects and the custom options.

```
await base.hooks.create(
    {
        "inspector": 0
    },
    {
        "name": "string",
        "teamId": 0,
        "typeName": "string",
        "__IMTCONN__": 0,
        "formId": "91282545501352"
    },
    {
        headers: [
            [
                'mode',
                'same-origin'
            ],
            [
                'cache',
                'force-cache'
            ],
            [
                'content-type',
                'application/json'
            ],
            [
                'Authorization',
                "Token " + "api_key"
            ]
        ]
    })
    .then(res => res.json())
    .then(json => console.log(json));
```

Note - options priority order: Custom Options **>** Fetch Options **>** Default Options