AJAX v.1.2

Julio Alejandro Santos Corona

12 de noviembre de 2020

La clase es una simplificación para realizar peticiones ajax, para ello utiliza la api web fetch de javascript.

1. Instancia

Para crear una instancia basta con crear un objeto a través del constructor de la función AJAX.main; a este se le puede pasar un booleano (default: true) que definirá el comportamiento de las rutas configuradas:

```
1  var ajax = new AJAX.main; five
2  ajax.route (``localhost:9001/index.js"); //Entrada al backend
```

2. Peticiones

La clase cuenta con métodos para realizar peticiones de forma fácil y con una estructura similar.

2.1. GET, POST, PUT, DELETE

Los verbos más utilizados en un api rest y la estructura de uso es similar para todos los casos, haciendo una distinción en el método get que el argumento de valores es opcional y dichos valores son transmitidos en la URL.

Argumento	Tipo	Opcional	Descripción
callback	function	No	Función a ejecutar luego de la resolución de la petición, recibe dos argumentos, res y status. Res contiene los datos en formato json y status el estado http de respuesta.
body	object	No	Objeto js que contiene los valores que serán pasados a través del body; en caso del método get este argumento es opcional y los valores son transmitidos en la URL.
url	string	No	Ruta a la que se hará la petición, el comportamiento está dado por el modo en que se creó la instancia.
middle	boolean	Si	Permite o niega la ejecución de un middleware. (Default: true)

Ejemplo:

```
1  //Peticiones get con y sin valores
2  ajax.get (function (res, status) {
3      //codigo
4  }, { var1: valor1 }, "/APP1");
5
6  ajax.get (function (res, status) {
7      //codigo
8  }, "/APP1");
9
10 ajax.post (function (res, status) {
11     //codigo
12 }, { var1: valor1 }, "/APP");
13
14 ajax.put (function (res, status) {
15     //codigo
16 }, {}, "/APP");
```

```
17
18  ajax.delete (function (res, status) {
19      //codigo
20  }, {}, "/APP", false);
```

2.2. FORM y TEXT

La clase cuenta con dos métodos extra que tienen la finalidad de realizar peticiones especiales.

El método **form** se puede utilizar cuando se cuente con una vista en la que exista una estructura html form, la petición se enviará siempre a través de POST y el body serán los objetos input, textarea, etc. contenidos en el.

Argumento	Tipo	Opcional	Descripción
callback	function	No	Función a ejecutar luego de la resolución de la petición, recibe dos argumentos, res y status.
form	object/string	No	El argumento puede recibir un objeto FormData o el id de la etiqueta form.
url	string	No	Ruta a la que se hará la petición.
middle	boolean	Si	Permite o niega la ejecución de un middleware. (Default: true)

HTML

```
1  <form id="formulary">
2      <input type="text" name="name">
3      <input type="number" name="age">
4  </form>
```

JS

```
1  var ajax = new AJAX.main;
2  ajax.route ("localhost:9001/index.js");
3
4  let form = new FormData (document.getElementById ("formulary"));
5
6  ajax.form (function (res, status) {
7      //codigo
8  }, form, "/FORMULARIO");
9
10 //El metodo tambien permite pasar solo el id del formulario
11 ajax.form (function (res, status) {
12     //codigo
13 }, "formulary", "/FORMULARIO");
```

El método **text** se utilizará para obtener el contenido de un archivo, para éste método no se ejecuta middleware, su definición es la siguiente:

Argumento	Tipo	Opcional	Descripción
callback	function	No	Función a ejecutar luego de la resolución de la petición, recibe dos argumentos, res y status.
url	string	No	Ruta al archivo.
content	string	Si	Header Content-Type, por default TEXT/PLAIN

Ejemplo:

```
1  ajax.text (function (text, status) {
2      //codigo
3  }, "/VIEWS/view.html", "TEXT/HTML");
```

3. Manejo de rutas

El comportamiento de las rutas se define durante el constructor, por default URLStatic tiene un valor true.

Las rutas se configuran a través del método **route** y **use**. La recomendación es configurar a través de **route** la ubicación del punto de entrada al backend y posteriormente definir la entrada de app con use.

Cuando URLStatic es true la ruta se forma con **route** + **use** + **url** de lo contrario será **use** + **url**.

Ejemplo:

```
1  var ajax = new AJAX.main;
2
3  ajax.route ("localhost:9001"); //Configuracion del punto de entrada a backend
4  ajax.use ("/APP"); //Configuracion del punto de entrada de app
5
6  //Peticion POST a ruta localhost:9001/APP/CONFIG?idUser=193
7  ajax.post (function (res, status) {
8      //codigo
9  }, { mail: "user@server.com" }, "/CONFIG?idUser=193");
```

4. Otros

4.1. Manejador de errores

La clase por default cuenta con un manejo de errores estándar que puede ser sustituído y adaptado a las necesidades del aplicativo en caso de existir un problema con la petición.

```
1  ajax.setErrors (function (err) {
2      if (err.status = 404) {
3          console.log ({ type: "Error", error: "err" });
4      }
5  });
```

4.2. Middleware

Durante las peticiones es posible configurar un middleware que se ejecutará antes de la función resolutora, dicho middleware recibe como argumentos la propia respuesta de la petición y el estatus de la misma.

```
1  ajax.setMiddleware (function (res, status) {
2      //codigo
3  });
```

4.3. Headers

Es posible, en caso de requerirlo, configurar las cabeceras que se incluirán en las peticiones a través del objeto **Headers** y del que se tiene acceso directo en ajax.**header**.