Conexión a API's: Probando GitHub API V3.0

Luego de haber leído la lectura de JSON aprenderemos como realizar una petición a un api de como github y crear a partir de este todo un cliente del mismo por medio de JSON.

Recordemos que en un API la mayoría de estos (como debe ser en el 2013) deben darnos la posibilidad de escoger entre xml y JSON, y cuando se habla de JavaScript es muy preferente escoger JSON que xml ya que si recordamos JSON nos guarda objetos de JavaScript ademas en el 2013 ya nadie usa xml.

Primero debemos hacer una petición al API de GitHub como en este caso GitHub tiene una restricción de peticiones por hora para cada IP usaremos LocalStorage para solucionar este problema.

Cuando hacemos una petición api debemos tener en cuenta tres caso de resultado, el primero es cuando todo sale mal y te responderá que el json no se pudo obtener, el segundo cuando excedes el límite de peticiones y te devuelve un JSON con un mensaje de error y por ultimo sera el maravilloso caso en donde todo sale bien y procedes a procesar la información.

Como dato importante de la forma en que se ejecuta una petición en JavaScript, siempre se realizará la petición y se seguirá ejecutando el código que está fuera de esta, cuando la petición tenga una respuesta de error o éxito se procederá con el código que corresponde según su solución, luego verán por que es importante saber esto.

Empecemos con código nosotros usamos .getJSON() en donde tendremos dos argumentos, el primero será la url a donde haremos la petición, el segundo será la función que ejecutará cuando se obtenga el JSON como un argumento de la función.

Haremos una petición a el API de GitHub con la organización de MejorandoLaClase de la siguiente forma:

```
org = "mejorandola";
$.getJSON("https://api.github.com/orgs/"+org+"?callback=?",org_info);
```

Lo cual nos devolverá el siguiente JSON:

```
{
  "login": "mejorandolaclase",
  "id": 2975064,
  "url": "https://api.github.com/orgs/mejorandolaclase",
  "repos_url": "https://api.github.com/orgs/mejorandolaclase/repos",
  "events_url": "https://api.github.com/orgs/mejorandolaclase/events",
  "members_url":
"https://api.github.com/orgs/mejorandolaclase/members{/member}",
  "public_members_url":
"https://api.github.com/orgs/mejorandolaclase/public_members{/member}",
  "avatar_url":
"https://secure.gravatar.com/avatar/df5413a54bf57ab18012f11af00ca4e5?d=ht
tps://a248.e.akamai.net/asse...",
  "name": "#MejorandoCurso",
  "company": null,
  "blog": "https://cursos.mejorando.la",
  "location": "https://cursos.mejorando.la",
  "email": "cursos@mejorando.la",
  "public_repos": 2,
  "public_gists": 0,
  "followers": 0,
  "following": 0,
  "html_url": "https://github.com/mejorandolaclase",
  "created_at": "2012-12-05T22:23:23Z",
  "updated_at": "2013-04-30T22:18:11Z",
  "type": "Organization"
}
```

De acá podemos hacer uso del nombre, el avatar, el número de repos públicas, el número de gist, el email y lo más importante la url del API de GitHub que me redirige a repositorios y miembros que serán las que más usemos entre otros datos random que usaremos a medida de lo que necesitemos.

Ahora crearemos la función org_info que recibió la información del JSON

y empezará a ejecutar el código que concierne a esta.

En este caso usaremos mustache.js para renderizar datos así que lo primero que debemos hacer es definir la función org_info(data) en donde data será el JSON y luego procedemos a definir un objeto llamado views en donde tendremos los datos que usaremos para renderizar, aparte agarramos la url de miembros y repos para saber la cantidad de miembros públicos que tiene la organización y para mostrar los repositorios que tiene esta.

Luego de esto haremos 2 peticiones, la primera será para la url de miembros en donde habrá una función llamada members_info y la segunda para la información de los repositorios.

```
function org_info (data){
    var num_members;
    data = data.data;
    view = {
        name : data.name,
        email : data.email,
        num_repos : data.public_repos,
        url : data.html_url
    }
    url_repos = data.repos_url;
    url_members = data.members_url.replace("{/member}","");
    $.getJSON(url_members, members_info);
    $.getJSON(url_repos, repo_info);
}
```

en donde sí vemos no hemos impreso nada al usuario ya que nos falta el número de miembros para imprimir la información general y luego cogeremos los repos y los imprimimos debajo.

Entonces, procedemos a crear la función members_info.

```
function members_info (member) {
    view.num_members = member.length;
    template = "<a href={{url}}><h2>{{name}}</h2></a>"+
```

```
                "<p>Repositorios: {{num_repos}}</p>"+
                "<p>Miembros: {{num_members}}</p>"+
                "<p>email: <a href='mailto:{{email}}'>{{email}}</a></p>";
    output = Mustache.render(template, view);
    $("#repoinfo").html(output);
}
```

en donde el item que nos importa es la longitud del arreglo y luego lo agregamos a nuestro objeto view que sera el que rendereemos, ahora creamos nuestro template con doble musteache y el identificador del dato que queremos colocar, luego lo colocamos en la variable output un Mustache,render del template con la vista y lo colocamos en un id con jQuery.

Si nos dimos cuenta el render lo hicimos hasta que la los miembros estuvieran listos ya que es la ultima solicitud
que necesitábamos completar para imprimir los datos, de ser contrario se hubiera impreso todo excepto los miembros.

Ya impresa la información de la organización seguimos con los repositorios en donde en el caso de la organización de MejorandoLaClase nos debió haber devuelto el siguiente JSON:

```
[
  {
    "name": "MejorandoCurso",
    "html_url": "https://github.com/mejorandolaclase/MejorandoCurso",
    "description": "Contenido Original de los Cursos Presenciales y de la Plataforma.",
    "contents_url": "https://api.github.com/repos/mejorandolaclase/MejorandoCurso/contents/{+path}"
  },
  {
    "name": "mapa-turistico",
    "html_url": "https://github.com/mejorandolaclase/mapa-turistico",
    "description": "Aplicación web para localizar sitios de los que estemos orgullosos, por y para los alumnos de @mejorandola",
    "contents_url": "https://api.github.com/repos/mejorandolaclase/mapa-turistico/contents/{+path}"
  }
]
```

En este caso elimine mucho del JSON dejando solo los elementos que

nos interesan que son: el nombre, la url del repositorio, la descripción y la url del API de GitHub que nos devolverá un JSON con el contenido del repositorio.

Ahora haremos la función repo_info de la siguiente manera:

```
function repo_info(repos) {
    repos.forEach(function (repo){
        repo_contents = repo.contents_url.replace("{+path}","");
        view = {
            name : repo.name,
            git_url : repo.html_url,
            description : repo.description
        }
        if (view.description == ""){
            view.description = "Repositorio Oficial";
        }
        template = "<div class='repo' id='{{name}}'>"+
                "<h3>{{name}}</h3>"+
                "<a class='github' href='{{git_url}}'>Miralo en
Github</a>"+
                "<p>{{description}}</p></div>";
        output = Mustache.render(template, view);
        $("#repositorios").prepend(output);
    });
}
```

Acá lo que hacemos es recibir el JSON en la variable repos y la recorreremos con un forEach en donde repo asumirá cada uno de los repositorios de donde tendremos los datos, luego recogemos la url del api en donde se encuentra los contenidos del repo en cuestión y luego procedemos a crear nuestra vista con el nombre, la url en GitHub y por ultimo la descripción evaluamos si tiene contenido y de no ser así colocamos uno por default, luego creamos nuestro template y lo rendereamos como lo explique en el ejemplo de miembros, ahora vemos que con jQuery y su función prepend que se basa en añadir el elemento como primero sin borrar el contenido que ya se halla cargado.

En este caso se cargaran los 2 repositorios y quedara como tarea hacer la conexión con GitHub para mostrar los contenidos de un repositorio

en base a este.

Como mencione antes tenemos el problema de que el API de GitHub limita el numero de peticiones por hora y por IP, igualmente mencione que lo arreglaríamos con LocalStorage así que te dejo de tarea que con la lectura de LocalStorage resuelvas este problema y lo compartas con la comunidad.