

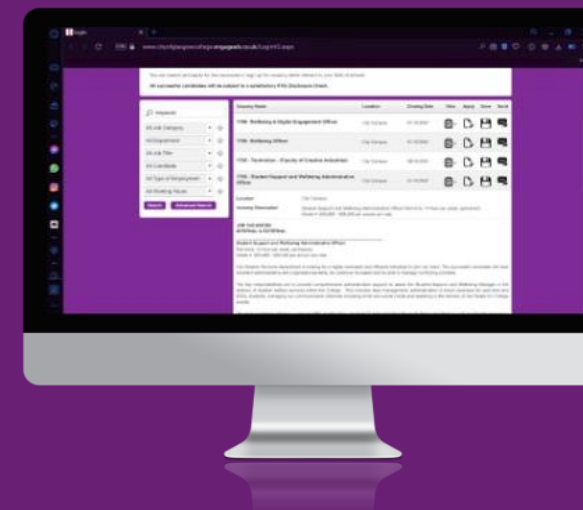


AJAX Call

# ¿Que es?

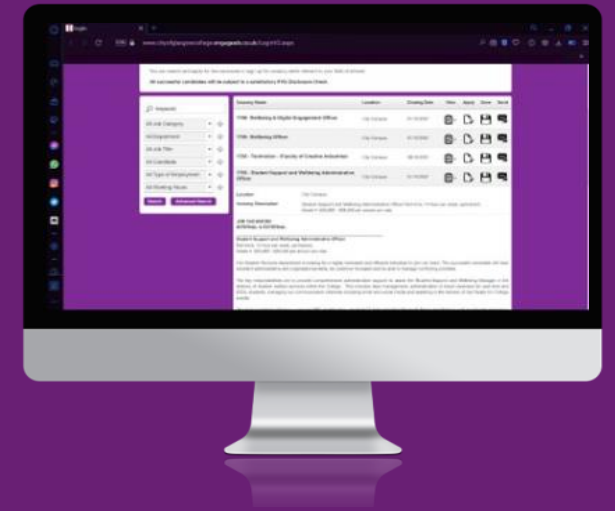
AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

Es un método utilizado en el desarrollo web con el fin de hacer peticiones para actualizar, renderizar o cambiar contenido en aplicaciones sin necesidad de hacer un refresh de la página, con esto conseguimos traer datos dinámicamente sin necesidad de crear un flujo mayor dentro del servidor y reducimos los tiempos de carga.



# ¿Para qué sirve en Indexación?

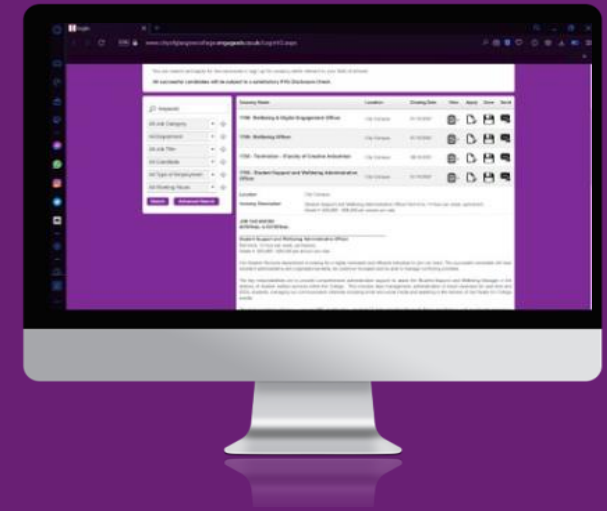
Sirven para traer vía petición los trabajos que vengan organizados en archivos JSON, permitiendo así consumir esos datos, haciendo muchísimo más efectivo el proceso de indexación reduciendo los tiempos de carga al requerir contenido.



# Además...

En muchos casos, la paginación se puede realizar desde las configuraciones del JSON, pudiendo paginar desde el Step Extract directamente para agilizar el proceso.

NOTA: Esto se realiza a riesgo de la cantidad de datos procesados en una sola corrida del Extract, ya que es mucha data, el boo podría arrojar error de timeout.



# ¿A qué nos referimos con "petición"?

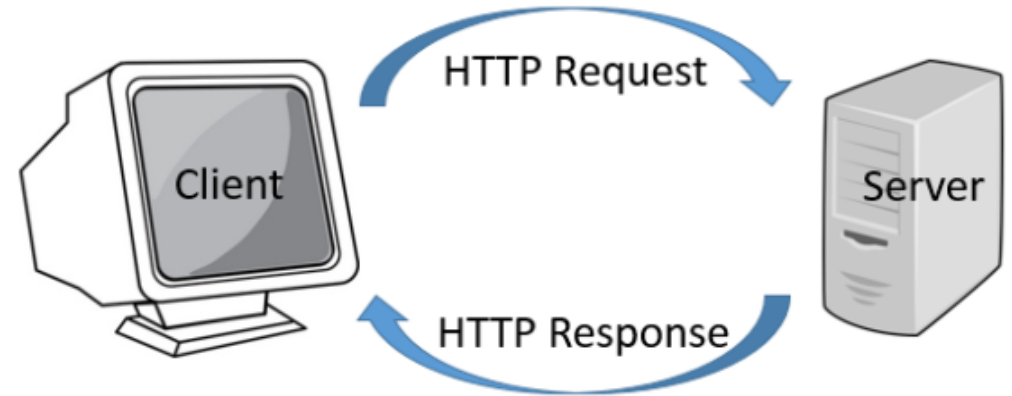
Es una solicitud ya sea de tipo GET o POST desde nuestro código como clientes, mediante el protocolo HTTP a un servidor con el fin de retornar datos específicos según los parámetros indicados.



# Protocolo HTTP

HTTP = "Hypertext Transfer Protocol".

Es el conjunto de normas, o protocolo principal para el manejo de datos en la web, nos permite realizar envíos o peticiones de datos en la misma. Su estructura se basa en la forma Cliente-Servidor.



# Petición de tipo GET

Request que debe ser usada principalmente para realizar solicitudes al servidor más no para en envío de información, ya que sus parámetros pasan vía URL.

```
var json;  
do {  
    var data = {};  
    $.ajax({  
        url : '',  
        headers: {  
            "Content-Type": "application/json;charset=UTF-8"  
        },  
        type : 'GET',  
        data : JSON.stringify(data),  
        dataType: "json",  
        async : false,  
        success: function(data) {  
            json = data;  
        }  
    });  
}
```

# Petición de Tipo POST

Se usa para comunicarse con el servidor, enviando información (Carga útil) que solo es visible por el mismo, con el fin de recibir o responder con datos según los parámetros indicados en ella.

```
var data = {};  
$.ajax({  
  url : '',  
  headers: {  
    "Content-Type": "application/json;charset=UTF-8"  
  },  
  type : 'POST',  
  data : JSON.stringify(data),  
});
```



# Métodos para su uso en Indexación

Estas peticiones HTTP que sirven para traer jobs contenidos dentro de archivos json, se realizan mediante el uso de la función AJAX dentro de la librería de jQuery.



# Configuración

## Spider Config

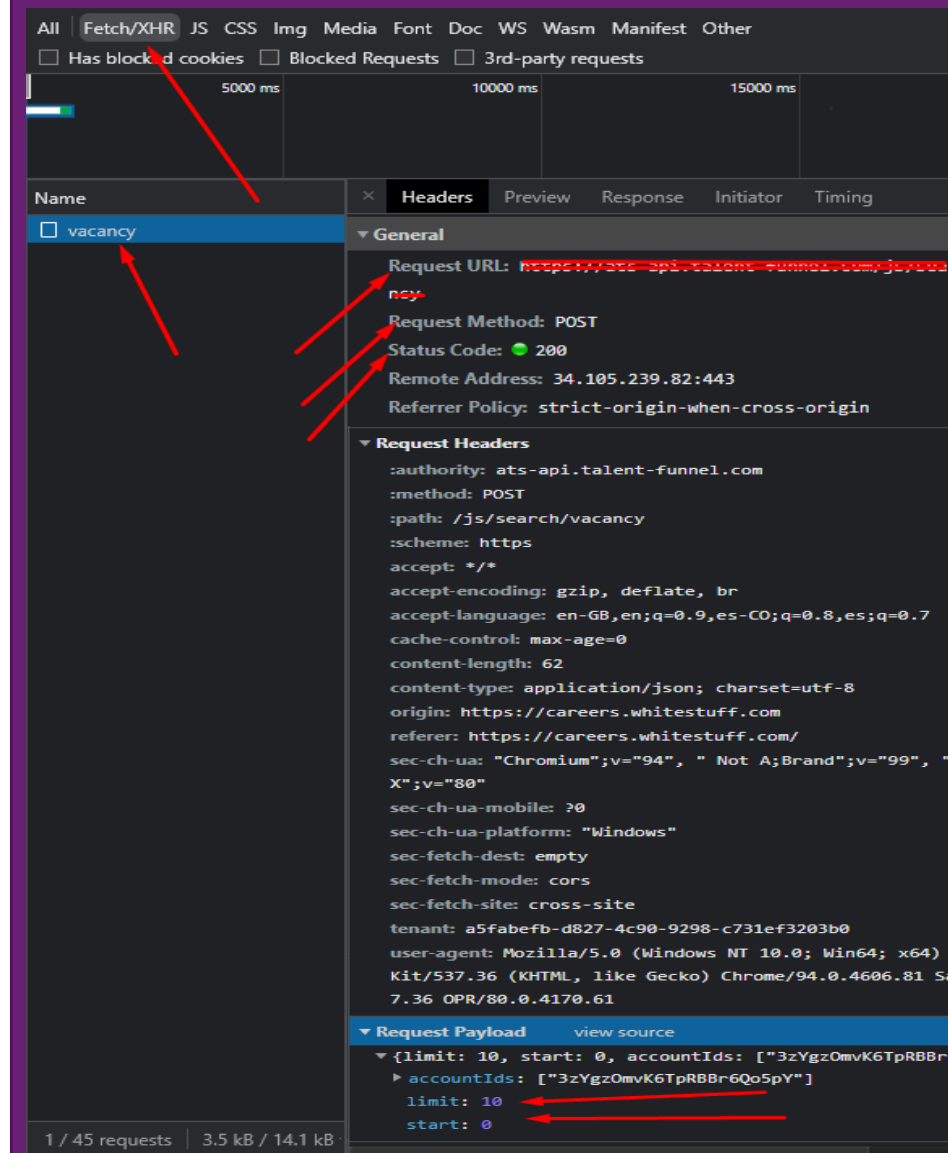
```
{
  "options": {
    "inactivateJQuery": false,
    "ignoreLoadErrors": false,
    "waitForPageLoadEvent": false,
    "waitForResources": false
  },
  "noimage": true,
  "skipResources": false,
  "noUnnecessaryResources": false
}
```

# Análisis

# Analizar la request en el jobsite.

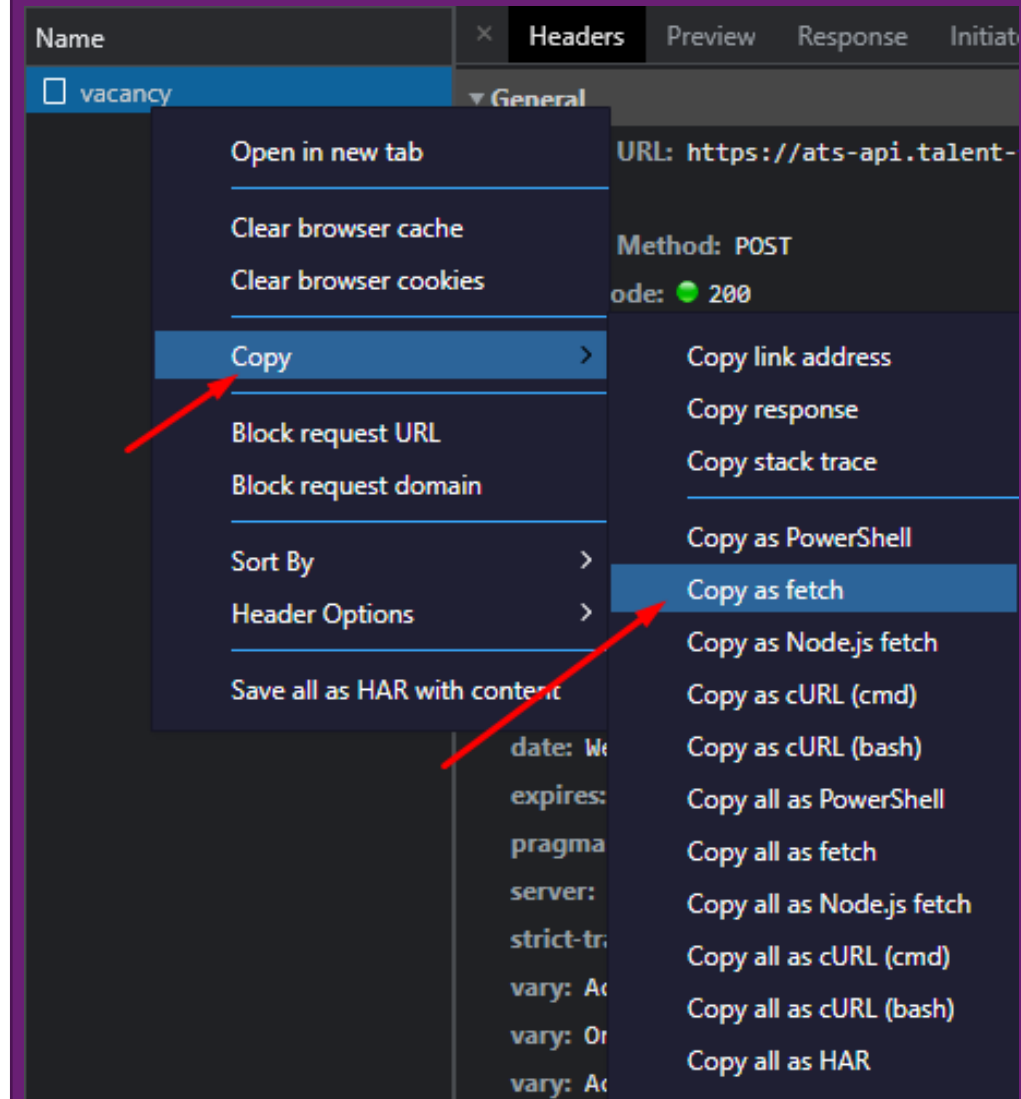
Varios items captan nuestra atención en un análisis preliminar del JSON, por ejemplo:

1. Metodo (GET/POST)
2. La formacion de la URL
3. Request Payload: Esta es la info que ponemos en la variable 'data' sobre todo si esta es presentada en forma de JSON





## Extraer el Fetch.


Al extraer el fetch, podemos obtener más directamente los 'headers' que vamos a requerir para la indexacion. Además de esto, el 'body', que es el 'Request Payload' y en los casos en que se exprese en forma de Query podríamos solo agregarlo a la URL.




# Extract

 Variable 'data' que almacena datos adicionales para la request.

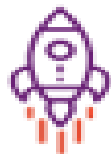
 "headers" configuraciones que requiere la request.

 Se organiza la extracción de los datos de los jobs, según sea el caso de cada jobsite.

 Con el uso de un ciclo Do-While podemos configurar una paginacion segun nos permita el site.

```
10 (function () {
11     var jobs = [];
12     var out = {};
13     var counter = 0;
14     var limit = 0;
15     var json;
16
17     do {
18         var data = { "filters": [{ "page": counter, "limit": 100 }] }; //datos adicionales para el request
19         $.ajax({
20             url: 'https://www.url.com', //link del json
21             headers: {
22                 "accept": "application/json, text/plain, */*"
23             }, //se obtienen con el fetch
24             type: 'POST',
25             data: JSON.stringify(data), //convierte a tipo String la variable 'data' cuando se usa en forma de JSON
26             dataType: "json",
27             async: false,
28             success: function (result) {
29                 json = result.data;
30                 limit = result.total;
31                 for (elem of json) {
32                     var job = {};
33                     job.title = elem.title;
34                     job.location = elem.location;
35                     job.url = elem.url;
36                     job.temp = "96";
37                     jobs.push(job);
38                 }
39                 counter += 1;
40             },
41             error: function (error) {
42                 msg(error);
43             }
44         });
45     } while (counter < limit);
46
47     out["jobs"] = jobs;
48     return out;
49 })();
```



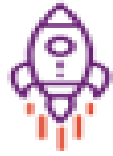


En casos donde el request es el tipo HTML, o el JSON trae un variable que contiene el HTML, se configura de una manera especial para tratar los datos como en extracción normal por selectores.



```
1 (function () {
2   var jobs = [];
3   var out = {};
4   var counter = 1;
5   var limit = 0;
6   var json;
7   $.ajax({
8     url: window.location.origin + '/jm-ajax/get_listings/?per_page=1000&page=1',
9     headers: {
10      "accept": "*/*"
11    },
12    type: 'POST',
13    dataType: "json",
14    async: false,
15    success: function (result) {
16      json = document.createElement('div');
17      json.innerHTML = result.html;
18      limit = result.max_num_pages;
19      var html_jobs = json.querySelectorAll('li[class*="job_listing"]');
20      for (var elem of html_jobs) {
21        if (typeof elem == "function") continue;
22        if (typeof elem == "number") continue;
23        var job = {};
24        job.reqid = elem.getAttribute('class').split(' ').shift().split('-').pop();
25        job.title = elem.querySelector('h3').textContent.trim();
26        job.url = elem.querySelector('a').href.trim();
27        job.location = elem.querySelector('div[class="location"]').textContent.trim();
28        job.dateposted_raw = elem.querySelector('time').getAttribute('datetime').trim();
29        job.dateposted_raw = job.dateposted_raw.split('-')[1] + '/' + job.dateposted_raw.split('-')[2] + '/' + job.dateposted_raw.split('-')[0];
30        if (elem.querySelector('li[class*="job-type"]')) {
31          job.source_jobtype = elem.querySelector('li[class*="job-type"]').textContent.trim();
32        }
33        job.temp = 96;
34        jobs.push(job);
35      }
36      counter = counter + 1;
37    },
38    error: function (error) {
39      msg(error);
40    }
41  });
42  out["jobs"] = jobs;
43  return out;
44 })();
```

# Pagination





En los casos donde se requiera (cantidad de jobs o data) se puede configurar una paginación que alivie la carga del extract evitando errores como el "Time-out" generado por el Boo.



```
726 (function () {  
727     var out = {};  
728     out["pass_it"] = pass_it;  
729     out.pass_it.offSet += 50  
730     if (out.pass_it.offSet < out.pass_it.limit) {  
731         out["has_next_page"] = true;  
732     } else {  
733         out["has_next_page"] = false;  
734     }  
735     out["wait"] = false;  
736     return out;  
737 })();
```

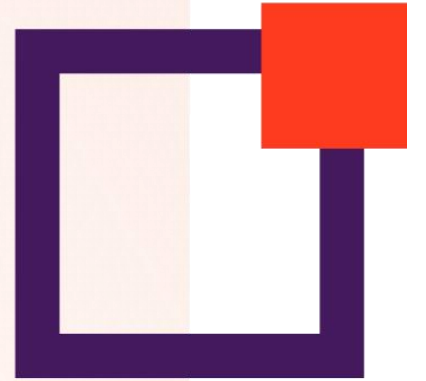
# Anotaciones Adicionales

-  El Extract no debe ser sobrecargado con información, es decir, una cantidad grande de jobs(>5k jobs aprox.), o una cantidad media de jobs (>2k jobs aprox.) pero con descripciones.
-  Este tipo de request también se pueden aplicar en las descripciones cuando sea necesario. (Tiempos de carga - bloqueos)

# ¡Muchas gracias Team!

[sebastian.galeano@talent.com](mailto:sebastian.galeano@talent.com)

[ramses.almanza@talent.com](mailto:ramses.almanza@talent.com)



Crawling page: 18	<a href="#">add-step.php?style=dark&amp;id=203235:2454</a>
No code for the step 'before-extract'	<a href="#">add-step.php?style=dark&amp;id=203235:2457</a>
>> extract code executed	<a href="#">add-step.php?style=dark&amp;id=203235:2457</a>
Found 16 new jobs!.	<a href="#">add-step.php?style=dark&amp;id=203235:2457</a>
Total: 356	
	<a href="#">add-step.php?style=dark&amp;id=203235:2451</a>
Test results stored	<a href="https://www.talent.com/private/tools/content/process/spiderResults/?crawlId=8df1d6b...">https://www.talent.com/private/tools/content/process/spiderResults/?crawlId=8df1d6b...</a>
<u>run time: 53s</u>	<a href="#">add-step.php?style=dark&amp;id=203235:2457</a>
Jobs Extracted: 356	<a href="#">add-step.php?style=dark&amp;id=203235:2457</a>
Found 356!	<a href="#">add-step.php?style=dark&amp;id=203235:2457</a>
Closing chrome	<a href="#">add-step.php?style=dark&amp;id=203235:2457</a>
Test Finished	<a href="#">add-step.php?style=dark&amp;id=203235:2451</a>
Unsubscribed to queue	<a href="#">add-step.php?style=dark&amp;id=203235:2448</a>
	<a href="#">add-step.php?style=dark&amp;id=203235:2448</a>
Disconnected from rabbit	

