¿Qué es JSON y JSONP?

Teléfono: 900 103 253
www.hostalia.com

HOSTALIA .com

En el mundo del diseño y programación web, llega un momento en el que es necesario gestionar gran cantidad de información de forma sencilla y eficaz, información que luego podrá ser mostrada en ese sitio o bien almacenada en base de datos. Uno de los sistemas más utilizados para llevar a cabo esta acción es el uso de JSON o en su defecto JSONP cuando se realiza intercambio de información entre servidores remotos. A lo largo de este White Paper explicaremos en qué consisten estos dos métodos.

¿Qué es JSON?

JSON es el acrónimo para JavaScript Object Notation y aunque por su nombre pueda parecer que forma parte del lenguaje de programación JavaScript, no tiene nada que ver con él, de hecho es un estándar basado en texto plano que se utiliza para el intercambio de información.

Básicamente JSON describe los datos con una determinada sintaxis que se utiliza para identificar y gestionar los datos y que es totalmente independiente de cualquier plataforma. Esto permite utilizar este sistema para intercambiar información entre diferentes aplicaciones, independientemente de lenguaje utilizado para su desarrollo. Por ejemplo, el emisor podría estar desarrollado en Java mientras que el receptor podría estar desarrollado con algún framework de PHP. Esto es posible gracias a que cada lenguaje tiene su propia librería para codificar y descodificar este tipo de cadenas en formato JSON.

En los últimos años, JSON se ha ido imponiendo al sistema XML, el utilizado hasta ahora para el intercambio de información, ya que el primero ofrece un sistema mucho más sencillo que el segundo, en el que la información debería ir almacenada entre etiquetas (<info>123</ info ><info >321</info><info>787</info>) mientras en JSON podemos almacenarlo todos seguidos como si fuera un array simple ([123, 321,787]).

¿Cómo se forma una cadena en formato JSON?

Crear una cadena de información basada en JSON es muy sencillo, únicamente hay que seguir una serie de indicaciones para que esté correctamente formada, siendo la primera de ellas que la información debe estar entre llaves ({ y }) que es lo que define el inicio y el final del objeto.

Teniendo claro esto, es necesario que nos familiaricemos con la sintaxis que se utiliza para su creación. Lo que tenemos que tener claro es que se trata de un sistema que está basado en el principio de "clave-valor" y que para asignar un valor a una clave, deberemos utilizar los dos puntos (:).

{"Empresa": "Hostalia"}

Lo siguiente que debemos tener claro son los tipos de datos que podemos representar.

- **Número**: tiene que tratarse de un entero o un float. Por ejemplo: 2, 3.14, 35…
- **String**: Texto representado entre comillas. Por ejemplo “Hostalia”.
- **Booleano**: Dos posibles valores, true o false.
- **Array**: Se representa por medio de corchetes []. En su interior se puede almacenar tantos pares de claves-valor como queramos, además de objetos.
- **Objetos**: En este caso, los objetos representan una serie de datos y están definidos entre llaves.
- **Valor nulo**: se representa con la cadena Null e indica que no tiene ningún valor asignado a esa clave.

Pasemos a continuación a ver un ejemplo completo de cómo representar una determinada información utilizando JSON. Para ello supondremos que tenemos un listado de empresas con sus nombres y el número de empleados de cada una de ellas.

- Empresa1 tiene 10 trabajadores
- Empresa2 tiene 345 trabajadores
- Empresa3 tiene 44 trabajadores

Esa información pasada a formato JSON quedaría de la siguiente manera

```
{
“Empresas”: [
{“NombreEmpresa”:”Empresa1”, “NumEmpleados”:10},
{“NombreEmpresa”:”Empresa2”, “NumEmpleados”:345},
{“NombreEmpresa”:”Empresa3”, “NumEmpleados”:44},
]
}
```

En el ejemplo que hemos dejado sobre estas líneas nos encontramos gran parte de los tipos de datos que se pueden utilizar. Cada una de las empresas, junto con el número de sus empleados, forma parte de un mismo objeto, de ahí que estén representados entre llaves.

```
{“NombreEmpresa”:”Empresa1”, “NumEmpleados”:10}
```

Por otro lado, también tenemos un array compuesto de tres objetos que están delimitados entre corchetes. Este array está asignado a la clave “Empresas”, siguiendo el principio clave-valor que hemos comentado anteriormente.

```
“Empresas”: [
{“NombreEmpresa”:”Empresa1”, “NumEmpleados”:10},
{“NombreEmpresa”:”Empresa2”, “NumEmpleados”:345},
```

{“NombreEmpresa”:”Empresa3”, “NumEmpleados”:44},

]

Lo que hemos visto aquí corresponde a un sencillo ejemplo, pero esta estructura se podía complicar tanto como quisiéramos.

JSON utilizado como sistema de almacenamiento

Aunque hemos comentado que la notación JSON fue creada en un principio para el intercambio de información entre diferentes aplicaciones, hoy en día se ha conseguido dar un uso algo diferente para lo que fue concebido, ya que es posible utilizarlo como sistema de base de datos donde poder almacenar información. Es el caso de MongoDB, un gestor de base de datos del tipo noSQL basado en clave-valor y que en vez de disponer las típicas tablas que utilizan los sistemas de bases de datos basados en SQL, disponen de colecciones que no son otra cosa que objetos JSON que podemos definir a nuestro antojo.

Ejemplo de uso de JSON con PHP

Como hemos comentado anteriormente en este White Paper, JSON puede ser utilizado con cualquier lenguaje de programación. El proceso de recorrer la información es igual de fácil independientemente del tipo de lenguaje con el que estemos trabajando. Ahora mostraremos un sencillo ejemplo de cómo se pueden recorrer estos archivos mediante PHP.

Partiremos del siguiente ejemplo que ha sido almacenado en un fichero con nombre “persona.json”.

```
{
"nombre":
    {
    "nombre":"Wilber",
```

```
        "apellido":"Lopez"
        },
    "profesion":
        {
        "profesion":"ingeniero",
        "gustos":"programacion"
        }
}
```

Lo primero que tendremos que hacer será leer el contenido del archivo "**persona.json**" y para ello utilizaremos la función de PHP "file_get_contents".

```
$datosPersona = file_get_contents("persona.json");
```

El siguiente paso será descodificar esa información para convertirla en un array y poder recorrerlo mediante un bucle.

```
$persona = json_decode($datosPersona);
```

Si imprimimos el valor de $persona, tendríamos que ver algo parecido a la siguiente imagen.

```
stdClass Object
(
    [nombre] => stdClass Object
        (
            [nombre] => wilber
            [apellido] => lopez
        )

    [profesion] => stdClass Object
        (
            [profesion] => ingeniero
            [gustos] => programacion
        )

)
```

Por último podríamos recorrerlo con la ayuda de un "foreach" para trabajar con la información que trae la cadena, ya sea para guardarlo en base de datos o hacer con ella lo que necesitemos.

```
foreach($empresas as $empresa){
    ……..
}
```

¿Qué es JSONP?

JSONP o JSON con padding se trata de una API utilizada para el intercambio de información entre sitios que pueden estar alojados en un mismo servidor o en alguna máquina remota. Su metodología es muy similar a JSON pero tiene una pequeña diferencia que es utilizada para paliar la limitación de AJAX entre dominios, que por motivos de seguridad, sólo permite hacer llamadas entre un mismo nombre de dominio.

El motivo por el que se puede acceder a la información que está alojada en otro servidor mediante JSONP y no con JSON es que los navegadores sí que permiten la carga de código JavaScript que traigas con la etiqueta SCRIPT y el atributo src de otro dominio.

La diferencia entre JSON y JSONP radica en su envoltura ya que en este segundo tipo, además de enviar la información, se manda una función a la que se le suele llamar callback, que funciona como un JavaScript englobando a la información. A continuación podréis ver un pequeño ejemplo donde se ve la diferencia de la que hablamos.

```
mifuncion({
    "datos": "del",
    "json": "con JSONP"
})
```

Gracias a esta propiedad, mediante el uso de JSONP, se puede acceder de forma remota a determinada información que se encuentre en otro servidor dedicado diferente. De esta forma, en nuestra página web podríamos tener el siguiente código, donde en el atributo src se le indica una url de un JSON que está en otro alojamiento.

```
<script type="text/javascript" src="http://dominio.com/datos.json?callback=parseJSON"></script>
```

Como podemos ver, en esa url se ha incluido la llamada a la función que forma parte del JSON por medio del atributo “callback”. Gracias a esto, mediante el uso de código, podríamos recuperar esa información y procesarla como hemos visto en un paso anterior.