Untuk parsing data JSON maupun XML sudah didukung oleh berbagai bahasa pemrograman seperti Javascript dan PHP. Untuk Javascript, XML diperlakukan seperti element HTML sehingga model parsingnya sama seperti parsing element HTML (menggunakan DOM)

Sedangkan untuk JSON, Javascript dapat langsung mengubahnya ke tipe objek (terdapat fungsi bawaan), sehingga dapat langsung diperlakukan seperti objek Javascript pada umumnya.

Building (membuat JSON dan XML)

Pada Javascript, membuat dokumen JSON sangat mudah karena sudah terdapat buit in function yaitu JSON.Stringify atau dengan bantuan library JQuery.

Sedangkan untuk membuat dokumen XML agak ribet karena sama seperti membuat dokumen HTML sehingga perlu menjalankan berbagai fungsi yang berkaitan dengan DOM.

Dengan demikian, dapat disimpulkan bahwa pada Javascript, JSON lebih mudah di gunakan dibanding XML.

Penutup

Saat ini JSON menjadi format standar untuk pertukaran data dan lebih banyak digunakan untuk pertukaran data via internet (protokol HTTP) karena load time yang lebih cepat dan bandwith yang diperlukan lebih kecil.

Dengan **struktur yang simpel**, format JSON lebih mudah digunakan dibanding format yang lain, terutama jika data yang di pertukarkan jumlahnya besar.

Kita akan lebih banyak merasakan manfaat JSON ketika menggunakannya bersama Javascript, seperti singkatannya (**Javacript** Object Notation).

Demikian pembahasan mengenai “Memahami JSON” semoga dapat bermanfaat.

Memahami JSON Pada PHP

Agus Prawoto Hadi, 15-04-2018, http://jagowebdev.com/memahami-json-pada-php/

Hi, sobat JWD, kali ini kita akan membahas JSON pada PHP.

Sebelumnya, pada artikel Panduan Lengkap Memahami JSON kita sudah membahas secara umum apa itu JSON, nah pada artikel ini, kita akan membahas implementasi JSON pada PHP.

PHP sendiri sudah mengakomodir JSON sejak PHP versi 5.2 dan terus disempurnakan pada PHP versi berikutnya.

Berikut ini contoh implementasi JSON pada PHP dan Javascript.

Environment

Sebelum melangkah lebih jauh, perlu sobat ketahui, bahwa ketika menulis artikel ini, saya menggunakan PHP versi 5.6 yang berjalan di Windows 10.

Dengan demikian, jika sobat menggunakan versi yang berbeda, bisa jadi hasil yang sobat dapatkan berbeda dengan yang ada di artikel ini.

I. Membuat JSON Pada PHP

Untuk membuat JSON di PHP maupun bahasa pemrograman lain, **tidak disarankan** untuk membuatnya secara manual, kenapa? karena berisiko terjadi error, sebaliknya, selalu gunakan fungsi bawaan (built-in) dalam hal ini fungsi `json_encode()`

Kenapa demikian? karena setiap fungsi bawaan, termasuk PHP akan memperhatikan standar yang ada, sehingga JSON yang dihasilkan bisa dipastikan valid dan memenuhi standar yang ada (ECMA-404)

Contoh penggunaan fungsi `json_encode()` sebagai berikut:

```
$config = array(
                'site_title' =>'Jagowebdev'
                ,'site_url'=>'http://jagowebdev.com'
                ,'themes'=>'crystal'
        );
echo json_encode($config);
// HASIL
{"site_title":"Jagowebdev","site_url":"http:\/\/jagowebdev.com","themes":"crystal"}
```

I.1. json_encode() – Format Penulisan

Secara umum format penulisan fungsi `json_encode()` adalah sebagai berikut:

```
json_encode ( $value [, int $options = 0 [, int $depth = 512 ]] )
```

Argumen:

- Argumen `$value` bisa berisi semua type data termasuk objek dan array, namun tidak termasuk resource. Adapun tipe data yang sering digunakan adalah array. Sebagai tambahan, argumen ini **wajib diencode**menggunakan UTF-8, hal ini sering menjadi penyebab terjadinya error ketika membuat JSON dari data database
- Argumen `$options`, berupa berbagai opsi yang dapat digunakan untuk mengubah parameter default (argumen ini kita bahas pada bagian bawah).
- `$depth` merupakan tingkat kedalaman (child) dari argemen `$value`.

Karena umumnya yang digunakan adalah array, maka agar pembahasan tidak terlalu panjang, pada bagian ini kita hanya akan membahas penggunaan fungsi `json_encode()` pada array.

I.2. json_encode() – Dasar Penggunaan

Fungsi `json_encode()` dapat digunakan baik untuk indexed array, maupun associative array, maupun gabungan keduanya, sebagai contoh kita memiliki tabel di database dengan nama config dengan data sebagai berikut:

id	option	value
1	site_title	Jagowebdev
2	site_description	Komunitas Jagowebdev (JWD) akan membantu anda menjadi “Jago” dibidang Web Development
3	site_url	http://jagowebdev.com
4	path	user\htdocs\jwd
5	themes	jwd

Contoh 1: indexed array

```php
$conn = mysqli_connect('localhost', 'root', '', 'test');
$query = mysqli_query($conn, 'SELECT * FROM config');
```

```
while ($row = mysqli_fetch_assoc($query)) {
        $json[] = $row['value'];}
echo '<pre>'; print_r($json); echo '</pre>';
$json = json_encode($json);
echo $json;
```

Hasilnya adalah:

```
// ARRAY
Array
(
    [0] => Jagowebdev
    [1] => Komunitas Jagowebdev (JWD) akan membantu anda menjadi "Jago" dibidang Web Development
    [2] => http://jagowebdev.com
    [3] => user\htdocs\jwd
    [4] => jwd
)
// JSON
["Jagowebdev","Komunitas Jagowebdev (JWD) akan membantu anda menjadi \"Jago\" dibidang Web Development","http:\/\/jagowebdev.com","user\\htdocs\\jwd","jwd"]
```

Contoh 2: Associative array:

```
$conn = mysqli_connect('localhost', 'root', '', 'test');
$query = mysqli_query($conn, 'SELECT * FROM config');

while ($row = mysqli_fetch_assoc($query)) {
        $json['site_option'][ $row['option'] ] = $row['value'];
}
$json['site_db'] = array(
                        'host' => 'localhost'
                        , 'user' => 'root'
                        , 'pass' => 'defult');
echo '<pre>'; print_r($json); echo '</pre>';
$json = json_encode($json);
echo $json;
```

Hasil yang kita peroleh adalah:

```
Array
(
    [site_option] => Array
        (
            [site_title] => Jagowebdev
            [site_description] => Komunitas Jagowebdev (JWD) akan membantu
anda menjadi "Jago" dibidang Web Development
            [site_url] => http://jagowebdev.com
            [base_path] => user\htdocs\jwd
            [themes] => jwd
        )

    [site_db] => Array
        (
            [host] => localhost
            [user] => root
            [pass] => defult
        )

)
{"site_option":{"site_title":"Jagowebdev","site_description":"Komunitas
Jagowebdev (JWD) akan membantu anda menjadi \"Jago\" dibidang Web
Development","site_url":"http:\/\/jagowebdev.com","base_path":"user\\htdocs\\
jwd","themes":"jwd"},"site_db":{"host":"localhost","user":"root","pass":"defu
lt"}}
```

Jika array berbentuk indexed array, maka JSON yang dihasilkan juga berbentuk array, misal:

```php
$config = array('localhost', 'root', 'pass');
echo json_encode($config);
// Hasil: ["localhost", "root", "pass"]
```

Jika kita ingin membuatnya menjadi objek maka kita dapat menggunakan opsi `JSON_FORCE_OBJECT`, misal:

```
$config = array('localhost', 'root', 'pass');
echo json_encode($config, JSON_FORCE_OBJECT);
{"0":"localhost","1":"root","2":"pass"}
```

Lebih lanjut tentang array dan objek pada JSON, dapat membaca kembali artikel sebelumnya: Panduan Lengkap Memahami JSON

Error Ketika Membuat JSON dari Data Database

JSON sering digunakan untuk menampilkan data dari database, seperti contoh pada bagian sebelumnya.

Nah, terkait data pada database, datanya bisa beragam, tergantung darimana data tersebut berasal (sebelum masuk ke database) dan bisa jadi tidak diencode menggunakan UTF-8 sehingga mengakibatkan error:

```
Malformed UTF-8 characters, possibly incorrectly encoded
```

Pembahasan lebih jauh tentang masalah ini dapat dibaca pada artikel: JSON Error Pada PHP – Membuat JSON Dari Database

Pentingnya escape slash (\/)

Secara default, Mulai PHP versi 5.4.0, fungsi `json_encode()` otomatis akan meng-*escape* setiap karakter slash, yaitu dari / menjadi \/

Dengan model seperti itu, untuk url, kita akan mendapati bentuk yang agak aneh, misal:
http://jagowebdev.com/images/article/json_logo.png
menjadi http://jagowebdev.com\/images\/article\/json_logo.png

Di standar JSON, escape ini sifatnya optional, tidak wajib, jika kita tidak menginginkannya, kita dapat menambahkan argumen `JSON_UNESCAPE_SLASH`.

Lantas, apa manfaat escape ini?

Tujuan utamanya adalah mengantisipasi jika didalam string json terdapat closing tag `</script>`, sehingga jika json tersebut diletakkan didalam tag `<script>` akan menyebabkan script utama berhenti dan dieksekusinya code javascript yang ada didalam JSON tersebut, misal:

```
<script type="text/javascript">
var json = {"content":"</script><script>alert(document.cookie)</script>"}
</script>
```

Pada contoh diatas, kode javascript yang ada didalam tag `<script>` yang ada pada data JSON akan dieksekusi, hal ini akan berbahaya dan berpotensi timbul serangan XSS.

Untuk itu sebaiknya biarkan karakter slash di-*escape*, atau kita dapat meng-*escape* tersebut jika **benar benar**tahu data JSON yang kita olah.

I.3 `json_encode` Argumen Kedua

Fungsi `json_encode()` menyediakan argumen kedua berupa opsi untuk mengubah parameter default, nilai opsi ini berupa bitmask dengan nilai 1, 2, 4, 8, 16, dst, agar lebih mudah, PHP telah menyediakan constant yang mewakili nilai tersebut, diantaranya:

1. `JSON_HEX_TAG` untuk mengubah tag HTML < dan > menjadi \u003C and \u003E
2. `JSON_HEX_APOS` mengubah single quote (‘) menjadi \u0027 Hal ini bermanfaat ketika meletakkan JSON pada atribut elemen HTML
3. `JSON_HEX_QUOT` mengubah double quote (”) menjadi \u0022
4. `JSON_HEX_AMP` mengubah (&) menjadi \u0026
5. `JSON_PRETTY_PRINT` memformat JSON dengan menambahkan spasi (bukan tab) sehingga menjadi mudah dibaca. Mulai ada di PHP versi 5.4.0. Contoh:

```
6.  $array = array('site_title' => 'Jagowebdev', 'site_url' =>
    'http://jagowebdev.com');
7.  echo json_encode($array);
8.  // Hasil: {"site_title":"Jagowebdev","site_url":"http:\/\/jagowebdev.com"}
9.  echo json_encode($array, JSON_PRETTY_PRINT);
10. // Hasil:
11. {
12.     "site_title": "Jagowebdev",
13.     "site_url": "http:\/\/jagowebdev.com"}
```

14. `JSON_UNESCAPED_SLASHES` membiarkan karakter (/) tidak diescape. Contoh:

```
15. $array = array('site_title' => 'Jagowebdev', 'site_url'
    =>'http://jagowebdev.com/blog/');
16. echo json_encode($array);
17. // Hasil:
    {"site_title":"Jagowebdev","site_url":"http:\/\/jagowebdev.com\/blog\/"}
18. echo json_encode($array, JSON_UNESCAPED_SLASHES);
    // Hasil: {"site_title":"Jagowebdev","site_url":"http://jagowebdev.com/blog/"}
```

19. JSON_FORCE_OBJECT Jika inputan berupa non-aassociative array, JSON yang dihasilkan tetap berbentuk objek bukan array. Hal ini bermanfaat jika penerima mensyaratkan bahwa JSON harus berbentuk objek. Contoh:

```
20.  $array = array('Jagowebdev', 'http://jagowebdev.com/blog/');
21.  echo json_encode($array);
22.  // Hasil array: ["Jagowebdev","http:\/\/jagowebdev.com\/blog\/"]
23.  echo json_encode($array, JSON_FORCE_OBJECT);
     // Hasil objek: {"0":"Jagowebdev","1":"http:\/\/jagowebdev.com\/blog\/"}
```

List constant diatas adalah yang paling sering digunakan, jika Sobat penasaran ingin tahu semua daftar constant yang ada, sobat dapat mengunjungi: PHP: Predefined Constant

Jika ingin menggunakan opsi lebih dari satu, gunakan bitwise OR (|), misal:

```
$json = array('html' => '<div class="title">Contoh HTML</div>');
$decode = json_encode($json, JSON_HEX_TAG | JSON_HEX_QUOT);
```

II. Parsing JSON di PHP

Setelah sebelumnya telah kita bahas bagaimana cara membuat JSON di PHP, pada bagian ini, kita bahas bagaimana mengubah JSON tersebut menjadi array maupun object.

Untuk mengubah JSON menjadi array atau object di PHP, kita gunakan fungsi `json_decode()`, fungsi ini memiliki format penulisan sebagai berikut:

```
json_decode( string $json [, bool $assoc = FALSE [, int $depth = 512 [, int $options
= 0 ]]] )
```

Jika argumen kedua dari fungsi ini, yaitu `$assoc` bernilai `true`, maka output yang dihasilkan adalah array, jika tidak, maka yang dihasilkan adalah objek, misal kita memiliki data JSON sebagai berikut:

```
$json = '{"site_option":
                {"site_title":"Jagowebdev"
                  ,"site_description":"Komunitas Jagowebdev (JWD) akan membantu anda
menjadi \"Jago\" dibidang Web Development"
                  ,"site_url":"http://jagowebdev.com"
                  ,"themes":"jwd"
```

```
                }
          ,"site_db":
                {"host":"localhost"
                  ,"user":"root"
                  ,"pass":"defult"
                }}';
```

Contoh 1: Hasil berupa objek:

```
$decode = json_decode($json);
var_dump($decode);

/*Hasil:
object(stdClass)#1 (2) {
  ["site_option"]=>
  object(stdClass)#2 (4) {
    ["site_title"]=>
    string(10) "Jagowebdev"
    ["site_description"]=>
    string(85) "Komunitas Jagowebdev (JWD) akan membantu anda menjadi "Jago" dibidang
Web Development"
    ["site_url"]=>
    string(21) "http://jagowebdev.com"
    ["themes"]=>
    string(3) "jwd"
  }
  ["site_db"]=>
  object(stdClass)#3 (3) {
    ["host"]=>
    string(9) "localhost"
    ["user"]=>
    string(4) "root"
    ["pass"]=>
    string(6) "defult"
  }
}
*/
```

Contoh 2: Hasil berupa array:

```
$decode = json_decode($json, true);
echo '<pre>'; print_r($decode); echo '</pre>';

/*Hasil:
Array
(
    [site_option] => Array
        (
            [site_title] => Jagowebdev
            [site_description] => Komunitas Jagowebdev (JWD) akan membantu
anda menjadi "Jago" dibidang Web Development
            [site_url] => http://jagowebdev.com
            [themes] => jwd
        )
    [site_db] => Array
        (
            [host] => localhost
            [user] => root
            [pass] => defult
        )
)
*/
```

Pilih Objek atau Array? hal ini tergantung dari situasi yang ada, umumnya bentuk yang sering digunakan adalah array.

CATATAN:

JSON bisa berasal dari berbagai sumber: baik dari database, file fisik (.txt, .json, dll), maupun dari url eksternal. Agar pembahasan tidak terlalu panjang, pembahasan mengenai topik ini dibahas pada artikel: JSON Pada Javascript

III. Handling Error Pada JSON

Ketika membuat dan memparsing data JSON baik dengan fungsi `json_encode()` maupun `json_decode()`, maka ketika terjadi error, keduanya tidak

memunculkan pesan error apapun. Fungsi `json_encode()` hanya menghasilkan nilai kosong, sedangkan pada fungsi `json_decode()` akan menghasilkan nilai `NULL`

Nah untuk mengetahui error yang terjadi, terdapat dua cara yang dapat kita lakukan yaitu menggunakan fungsi:

1. `json_last_error()`
   Fungsi ini mulai ada pada PHP versi 5.3.0, fungsi akan menghasilkan nilai integer yang memiliki arti tertentu sebagai hasil dari eksekusi fungsi `json_encode()` dan `json_decode()`
2. `json_last_error_msg()`
   Fungsi ini mulai ada pada PHP versi 5.5.0, fungsi akan langsung menghasilkan pesan yang mecerminkan hasil eksekusi fungsi `json_encode()` dan `json_decode()`

Cara paling mudah untuk mengetahui error yang terjadi adalah dengan menggunakan fungsi `json_last_error_msg()` karena yang dimunculkan langsung pesan errornya bukan kode , contoh:

```
$json = '{"site_option":
                {"site_title":"Jagowebdev"
                  ,"site_url":"http://jagowebdev.com"
                }
         }';
$decode = json_decode($json, true);
$last_error = json_last_error_msg();
if (strtolower($last_error) != 'no error') {
        echo 'ERROR: ' . $last_error; die;
}
```

NOTE: Terkait contoh diatas, jika tidak terjadi error, maka fungsi `json_last_error_msg()` akan menghasilkan string “No error”

Agar lebih fleksibel dan pesan error lebih terorganisir, kita dapat menggunakan fungsi json_last_error() fungsi ini akan **menghasilkan nilai integer** mulai dari 0 s.d 10

Nilai integer ini telah diterjemahkan ke bentuk constant sebagai berikut:

Kode Error Pada JSON

Constant	Integer	Arti	Versi PHP
JSON_ERROR_NONE	0	No error has occurred	
JSON_ERROR_DEPTH	1	The maximum stack depth has been exceeded	
JSON_ERROR_STATE_MISMATCH	2	Invalid or malformed JSON	
JSON_ERROR_CTRL_CHAR	3	Control character error, possibly incorrectly encoded	
JSON_ERROR_SYNTAX	4	Syntax error	
JSON_ERROR_UTF8	5	Malformed UTF-8 characters, possibly incorrectly encoded	PHP 5.3.3
JSON_ERROR_RECURSION	6	One or more recursive references in the value to be encoded	PHP 5.5.0
JSON_ERROR_INF_OR_NAN	7	One or more NAN or INF values in the value to be encoded	PHP 5.5.0
JSON_ERROR_UNSUPPORTED_TYPE	8	A value of a type that cannot be encoded was given	PHP 5.5.0
JSON_ERROR_INVALID_PROPERTY_NAME	9	A property name that cannot be encoded was given	PHP 7.0.0
JSON_ERROR_UTF16	10	Malformed UTF-16 characters, possibly incorrectly encoded	PHP 7.0.0

Sehingga jika kita mencetak konstanta tersebut, kita akan mendapatkan nilai integer nya, misal: jika kita jalankan `echo JSON_ERROR_SYNTAX` maka kita akan mendapatkan nilai 4
Dengan model seperti diatas, kita dapat mendefinisikan sendiri pesan error, misal:

```
$json = '{"title":"Memahami JSON Pada PHP - Part II}';
$decode = json_decode($json);
if (json_last_error() == JSON_ERROR_UTF8) {
        echo 'ERROR: Terdapat karakter diluar UTF-8';
}
```

atau

```
if (json_last_error() == 5) {
        echo 'ERROR: Encoding bukan UTF-8';
}
```

Untuk menghandle semua pesan error, kita dapat membuat sebuah array dengan format sebagai berikut:

```
$json_error = array {
        JSON_ERROR_DEPTH => 'Maksimum kedalaman JSON terlampaui'
        , JSON_ERROR_STATE_MISMATCH => 'Format JSON tidak valid'
        , JSON_ERROR_CTRL_CHAR => 'Control Character Error, Mungkin jenis encoding
tidak sesuai'
        , JSON_ERROR_SYNTAX => 'Syntax error, Apakah string mengandung karakter
escape?'
        , JSON_ERROR_UTF8 => 'Terdapat character non UTF-8, Mungkin jenis encoding
tidak sesuai'
        , JSON_ERROR_RECURSION => 'Terdapat recursive pada nilai yang di encode'
        , JSON_ERROR_INF_OR_NAN => 'Terdapat nilai INF atau NAN pada string'
        , JSON_ERROR_UNSUPPORTED_TYPE => 'Terdapat nilai yang tidak dapat di encode'
        , JSON_ERROR_INVALID_PROPERTY_NAME  => 'Terdapat nama property yang tidak
dapat di encode'
        , JSON_ERROR_UTF16 => 'Terdapat character non UTF-16, Mungkin jenis encoding
tidak sesuai'
}

$json = '{"title":"Memahami JSON Pada PHP - Part II}';
$decode = json_decode($json, true);

$last_error = json_last_error();
if ($last_error == 0) {
        print_r($decode);
} else {
        echo $json_error[$last_error];
}
```

IV. JSON dan XSS

Setiap string yang berhubungan dengan javascript dan HTML maka akan tidak lepas dari pembahasan tentang XSS, demikian juga JSON. XSS ini terjadi salah satunya ketika script HTML terdapat element `<script>` yang berasal dari **sumber tidak terpercaya** (seperti form input dimana pengunjung bebas memasukkan data).

Element script tersebut tersebut mengandung kode javascript yang akan mengekpose data pribadi user kemudian mengirim data tersebut ke pembuat script.

Terkait dengan JSON, pembuatan JSON dengan fungsi `json_encode()` **sudah aman**, karena (seperti telah disampaikan sebelumnya) fungsi tersebut sudah meng-escape slash (<\/script>) – mulai PHP versi 5.4.0 -, sehingga membuat kode javascript di dalam JSON tidak berjalan.

Terlebih lagi, perlindungan terhadap XSS sebenarnya tidak dari sisi JSON nya melainkan pada sisi inputan user, dimana ketika ada inputan dari luar yang akan langsung dieksekusi atau disimpan ke database, data inputan tersebut harus di sanitasi terlebih dahulu, misal menggunakan fungsi `htmlentities()` sehingga misal ketika dibuat menjadi JSON atau format lain, data tersebut sudah aman.

Apakah perlu HEX?

Pembahasan yang sering muncul adalah, jika json kita mengandung tag HTML, atau tag `<script>`, perlukah kita menggunakaan opsi JSON_HEX_XXX untuk mengubah elemen HTML menjadi hex? seperti berikut ini:

```
$array = array('html' => '</script><script>document.cookie</script>');
$json = json_encode($json, JSON_HEX_TAG | JSON_HEX_QUOT | JSON_HEX_APOS |
JSON_HEX_AMP);
```

Sekali lagi bahwa fungsi `json_encode` ini sudah aman sehingga penggunaan argumen JSON_HEX_xxx tidak diperlukan lagi terlebih jika sumber data sudah terpercaya, misal dari server kita sendiri dan kita sendiri yang membuat data tersebut.

V. Penutup

PHP menyediakan tools bawaan yang komplet untuk bekerja dengan JSON, hal yang perlu diperhatikan adalah versi PHP yang kita gunakan, karena pada PHP versi sebelum 5.6, terdapat beberapa bug yang merepotkan.

Demikian pembahasan mengenai JSON Pada PHP semoga bermanfaat.