Json, Ajax, POO e PHP

Prof. Me. Hélio Esperidião

JSON - JavaScript Object Notation

É uma formatação leve de troca de dados.

Para seres humanos, é fácil de ler e escrever.

Para máquinas, é fácil de interpretar e gerar.

É baseado em um subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3a Edição -Dezembro - 1999.

JSON

01

JSON é em formato texto e completamente independente de linguagem

02

Formato ideal de troca de dados

03

é um formato compacto, de padrão aberto independente, de troca de dados simples e rápida (parsing) entre sistemas

ESTRUTURA

Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *object*, record, struct, dicionário, hash table, keyed list, ou arrays associativas.

Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência.

EXEMPLO - JavaScript

- myObjt é um objeto.
- O método stringify converte o objeto para texto json.

```
var myObj = { "name":"John", "age":31, "city":"New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

EXEMPLO - JavaScript

- myJson é uma String.
- O método parse converte de string para objeto;

```
var myJSON = '{ "name":"John", "age":31, "city":"New York" }';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Exemplo Array

Acesso ao carro na posição zero do vetor

```
<script>

var myObj, x,y;
myObj = {
    "name":"John",
    "age":30,
    "cars":[ "Ford", "BMW", "Fiat" ]
};
x = myObj.name;
y=myObj.cars[0];
document.getElementById("demo").innerHTML = x + " - " +y;

</script>
```

Varrendo todos as posições

```
<script>

var myObj, i, j, x = "";
myObj = {
    "name":"John",
    "age":30,
    "cars": [
        { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
        { "name":"BMW", "models":[ "320", "X3", "X5" ] },
        { "name":"Fiat", "models":[ "500", "Panda" ] }
    ]
}

for (i in myObj.cars) {
    x += "<h2>" + myObj.cars[i].name + "</h2>";
    for (j in myObj.cars[i].models) {
        x += myObj.cars[i].models[j] + "<br>";
    }
}


document.getElementById("demo").innerHTML = x;

</script>
```

Exemplo php

```php
<?php
$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";

$myJSON = json_encode($myObj);

echo $myJSON;
?>
```

Retorna um object ou um array associativo se o parâmetro opcional assoc é TRUE.

```php
<?php
$myArr = array("John", "Mary", "Peter", "Sally");

$myJSON = json_encode($myArr);

echo $myJSON;
?>
```

```php
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var_dump(json_decode($json));
var_dump(json_decode($json, true));

?>
```

XML (*e**X**tensible **M**arkup **L**anguage*)

A principal característica do XML, de criar uma infraestrutura única para diversas linguagens, é que linguagens desconhecidas e de pouco uso também podem ser definidas sem maior trabalho e sem necessidade de ser submetidas aos comitês de padronização.

O XML é um formato para a criação de documentos com dados organizados de forma hierárquica, como se vê, frequentemente, em documentos de texto formatados, imagens vetoriais ou bancos de dados.

Vantagens

- Separação do conteúdo da formatação
- Simplicidade e legibilidade, tanto para humanos quanto para computadores
- Possibilidade de criação de tags sem limitação
- Criação de arquivos para validação de estrutura (chamados DTDs)
- Interligação de bancos de dados distintos
- Concentração na estrutura da informação, e não na sua aparência

Exemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita nome="pão" tempo_de_preparo="5 minutos" tempo_de_cozimento="1 hora">
  <titulo>Pão simples</titulo>
  <ingredientes>
    <ingrediente quantidade="3" unidade="xícaras">Farinha de Trigo</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1.5" unidade="xícaras" estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, coloque numa bandeja e asse num forno.</passo>
  </instrucoes>
</receita>
```

XML em javaScript

```
<script>
var parser, xmlDoc;
var text = "<bookstore><book><title>Everyday Italian</title><author>Giada De Laurentiis</author>" +
"<year>2005</year></book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>
```

JSON VERSUS XML

O JSON pode ser considerado concorrente da XML na área de troca de informações.

Vejamos algumas das principais semelhanças e diferenças entre os modelos de marcação das informações:

SEMELHANÇAS

- Representam informações no formato texto.;
- Ambos podem ser utilizados para transportar informações em aplicações AJAX.
- Ambos são independentes de linguagem.
- Dados representados em XML e JSON podem ser acessados por qualquer linguagem de programação, através de API’s específicas.

DIFERENÇAS

Json Não é uma linguagem de marcação. (Não possui tags de abertura e de fechamento).

Json É tipicamente destinado para a troca de informações, enquanto XML possui mais aplicações.

Por exemplo: existem bancos de dados no formato XML e estruturados em SGBD XML nativo.

Exemplo completo

Php – json - ajax

Arquivos:

- Cliente.php
- Cliente_Controle_Get_Cliente.php
- Formulario.html

Formulario.html

```
function cadastrarCliente() {
    var nome =  document.getElementById('txtNome').value;
    var email= document.getElementById('txtEmail').value;
    var senha = document.getElementById('txtSenha').value;
    const ajax = new XMLHttpRequest();
    ajax.onload = function() {
        var objJsonCliente = JSON.parse(this.responseText);
        var nomeServidor  = objJsonCliente.nome;
        var emailServidor = objJsonCliente.email;
        var senhaServidor = objJsonCliente.senha;
        document.getElementById("divResposta").innerHTML += nomeServidor + " - " + emailServidor;
        document.getElementById("divResposta").innerHTML += senhaServidor;
    }
    ajax.open("POST", "Cliente_Controle_Get_Cliente.php");
    ajax.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    var variaveis = "";
    variaveis +="&txtNome= " + nome;
    variaveis +="&txtEmail=" + email;
    variaveis +="&txtSenha=" + senha;
    ajax.send(variaveis);
}
```

Cliente_Controle_Get_Cliente.php

- Recebe os dados do Formulario.html
- Instancia um cliente
- Converte o cliente para String json

```php
<?php
include "Cliente.php";

$vNome =     $_POST['txtNome'];
$vEmail =    $_POST['txtEmail'];
$vSenha =    $_POST['txtSenha'];

$cliente = new Cliente();

$cliente->setNome($vNome);
$cliente->setEmail($vEmail);
$cliente->setSenha($vSenha);

echo json_encode($cliente);

?>
```

Classe Cliente.

- Interface JsonSerializable
- Método jsonSerialize()
  - Deve ser implementado para converter o objeto em texto no formato json.

```php
<?php
    class Cliente implements JsonSerializable {
        private $nome;
        private $email;
        private $senha;
        public function getNome(){
            return $this->nome;
        }
        public function setNome($v){
            $this->nome = $v;
        }
        public function getEmail(){
            return $this->email;
        }
        public function setEmail($v){
            $this->email = $v;
        }
        public function getSenha(){
            return $this->senha;
        }
        public function setSenha($v){
            $this->senha = $v;
        }
         public function jsonSerialize(){
          return get_object_vars($this);
        }
    }
?>
```

Array de objetos

- Muitas das vezes é necessário trabalhar com um array de objetos
- Um caso muito simples é o resultado de uma busca de clientes no banco que pode retornar um array de clientes.

```php
<?php
include "Cliente.php";

$vNome =     $_POST['txtNome'];
$vEmail =    $_POST['txtEmail'];
$vSenha =    $_POST['txtSenha'];

$cliente[0] = new Cliente();
$cliente[0]->setNome($vNome);
$cliente[0]->setEmail($vEmail);
$cliente[0]->setSenha($vSenha);

echo json_encode($cliente);

?>
```

Vetor de objetos json

[{"nome":"
helio","email":"helio@gmail.com","senha":"123456"}]

[{"nome":" 1","email":“1","senha":“1"},
{"nome":" 2","email":"2","senha":“2"}]

Recebendo vetor de objetos

- O javascritp é capaz de detectar o tamanho do array e criar uma estrutura de repetição que passe por todos os elementos

```
function cadastrarCliente() {
    var nome =  document.getElementById('txtNome').value;
    var email= document.getElementById('txtEmail').value;
    var senha = document.getElementById('txtSenha').value;
    const ajax = new XMLHttpRequest();
    ajax.onload = function() {
        var objJsonCliente = JSON.parse(this.responseText);
        document.getElementById("divResposta").innerHTML += this.responseText;
        var tamanhoVetor = objJsonCliente.length;
        var i=0;
        for(i=0;i<tamanhoVetor;i++){
        var nomeServidor  = objJsonCliente[i].nome;
        var emailServidor = objJsonCliente[i].email;
        var senhaServidor = objJsonCliente[i].senha;
            document.getElementById("divResposta").innerHTML += nomeServidor + " - " + emailServidor;
            document.getElementById("divResposta").innerHTML += " - " + senhaServidor;
        }
    }
    ajax.open("POST", "Cliente_Controle_Get_Cliente.php");
    ajax.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    var variaveis = "";
    variaveis +="&txtNome= " + nome;
    variaveis +="&txtEmail=" + email;
    variaveis +="&txtSenha=" + senha;
    ajax.send(variaveis);
}
```

Exemplo com banco de dados e json

Banco.php

```php
<?php
header("Content-type: text/html; charset=utf-8");
class Banco {
    private $host="localhost";
    private $user="root";
    private $password="";
    private $connection=null;
    private $db="agenda";
    function __construct() {
        $this->connection = mysqli_connect($this->host,$this->user,$this->password,$this->db);
        mysqli_set_charset( $this->connection, "utf8");
    }

    function getConnection(){
        return $this->connection;
    }

}
```

Estado.php

```php
<?php
header("Content-type: text/html; charset=utf-8");
require_once("Banco.php");
class Estado{
    private $idEstado;
    private $estadoNome;
    private $banco;
    function __construct() {
        $this->banco = new Banco();
    }
    public function getAllFromEstado(){
        $result = $this->banco->getConnection()->query("SELECT * FROM estado");
        if($result){
            while ($row = $result->fetch_object()){
                $matrizResposta[] = $row;
            }
        }
        $result->close();

        return $matrizResposta;
    }
    public function setEstadoNome($estadoNome){
        $this->estadoNome=$estadoNome;
    }
    public function setIdEstado($idEstado){
        $this->idEstado=$idEstado;
    }
    public function getIdEstado(){
        return $this->idEstado;
    }
    public function getEstadoNome(){
        return $this->estadoNome;
    }
}
?>
```

Controler_getAllFromEstado.php

```php
<?php
    header("Content-type: text/html; charset=utf-8");
    require_once("Estado.php");
    $estado = new Estado();
    $resultado=$estado->getAllFromEstado();
    echo json_encode($resultado,JSON_UNESCAPED_UNICODE);

?>
```