UNIVERSITE VIRTUELLE DU SENEGAL

foo nekk foofu la

Echange de données JSON avec le serveur

Le format de données JSON

El hadji Mamadou NGUER Enseignant chercheur en Informatique à l'UVS

2019

Chapitre 4 : Echange de données JSON avec le serveur

Objectifs spécifiques : A la suite de ce chapitre, l'étudiant doit être capable de:

1. Générer des données JSON dans le serveur avec PHP
2. Echanger des données JSON entre le client et un serveur en utilisant PHP
3. Afficher dans le client des données JSON générées par PHP
4. Utiliser la technique JSONP d'échange de données JSON

Chapitre 4 : Echange de données JSON avec le serveur

Plan de la séquence :

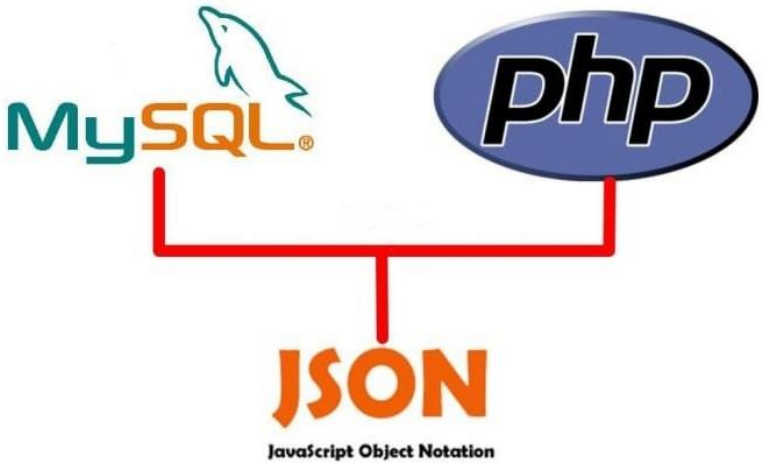
1. Introduction
2. JSON avec PHP
3. JSON avec PHP/MySQL
4. La technique JSONP

Introduction

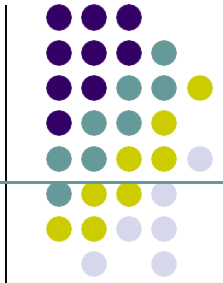
- Il est courant de lire des données JSON depuis un serveur Web et les afficher dans une page Web.

- On verra dans ce chapitre comment échanger des données JSON entre le client et le serveur en utilisant PHP.

- Notons que PHP est un langage de script côté serveur qui dispose de quelques fonctions intégrées pour gérer JSON.



JSON avec PHP



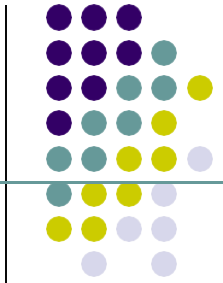
Conversion d'objets PHP en JSON.

- Un objet PHP peut être converti en objet JSON.
- Elle se fait en utilisant la fonction PHP json_encode ()
- L'exemple ci-dessous converti un objet PHP nommé $myObj en une donnée JSON.

```php
<?php
$myObj = new stdClass();
$myObj->nom = "Djiby";
$myObj->age = 30;
$myObj->ville = "Dakar";

$myJSON = json_encode($myObj);
echo $myJSON;
?>
```

JSON avec PHP



Affichage dans le client de données JSON générées par PHP

- Il se fait en utilisant un appel d'AJAX pour interroger le fichier PHP.
- JSON.parse () est utilisée pour convertir le résultat en un objet JavaScript:

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var myObj = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myObj.nom;
  }
};
xmlhttp.open("GET", "demo_file.php", true);
xmlhttp.send();
```

JSON avec PHP

Affichage dans le client de données JSON générées par PHP

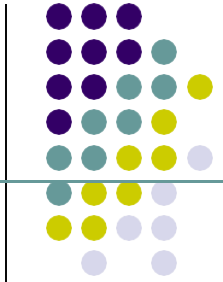
- Les tableaux en PHP seront également convertis en JSON lors de l'utilisation de la fonction PHP json_encode () :

```php
<?php
$myArr = array("Djiby", "Mary", "Pathé", "Saly");
$myJSON = json_encode($myArr);
echo $myJSON;
?>
```

- L'affichage des données -au niveau le client se fera comme suit :

```javascript
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var myObj = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myObj[2];
  }
};
xmlhttp.open("GET", "demo_file_array.php", true);
xmlhttp.send();
```

JSON avec PHP/MySQL



Couplage à une base de données MySQL

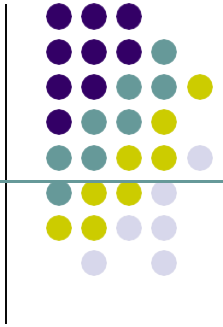
- PHP est un langage de programmation côté serveur, et peut être utilisé pour accéder à une base de données.

- Imaginons qu'on dispose d'une base de données sur un serveur et qu'on souhaite lui envoyer une requête à partir du client pour obtenir les 10 premières lignes d'un tableau appelé "clients".

- Pour cela, sur le client, on crée un objet JSON qui décrit le nom de la table et le nombre de lignes qu'on souhaite renvoyer.

- Avant d'envoyer la requête au serveur, on convertit l'objet JSON en chaîne et l'envoyer comme paramètre d'url de la page PHP:

JSON avec PHP/MySQL



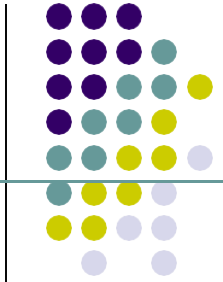
Couplage à une base de données MySQL

- **Envoi de la requête :** Elle se fait en utilisant JSON.stringify ()

```
obj = { "table":"clients", "limit":10 }; //1
dbParam = JSON.stringify(obj); //2
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {//4
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText; //5
  }
};
xmlhttp.open("GET", "json_demo_db.php?x=" + dbParam, true);
xmlhttp.send(); //3
```

- **Explication :** Cet exemple permet de :
  - Définir un objet contenant une propriété de table et une propriété de limite. (1)
  - Convertir l'objet en une chaîne JSON. (2)
  - Envoyer une requête au fichier PHP, avec la chaîne JSON comme paramètre. (3)
  - Attendre que la requête soit exécutée et le résultat obtenu (en JSON) (4)
  - Afficher le résultat reçu du fichier PHP. (5)

JSON avec PHP/MySQL



Couplage à une base de données MySQL

- **Traitement de la requête :** Il se fait dans le fichier json_demo_db.php :

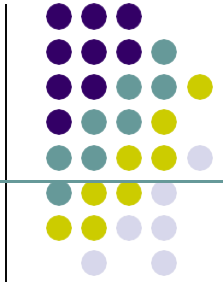
```php
<?php
header("Content-Type: application/json; charset=UTF-8");
$obj = json_decode($_GET["x"], false);

$conn = new mysqli("myServer", "myUser", "myPassword", "myDB");
$stmt = $conn->prepare("SELECT nom FROM ? LIMIT ?");
$stmt->bind_param("ss", $obj->table, $obj->limit);
$stmt->execute();
$result = $stmt->get_result();
$outp = $result->fetch_all(MYSQLI_ASSOC);

echo json_encode($outp);
?>
```

- **Explication :** Cet exemple permet de :
  - Convertir la requête en objet, en utilisant la fonction PHP json_decode () .
  - Accéder à la base de données et remplir un tableau avec les données reçues.
  - Ajouter le tableau à un objet et renvoyez l'objet au format JSON à l'aide de la fonction json_encode ()

JSON avec PHP/MySQL



Couplage à une base de données MySQL

- **Parcourt du résultat :** Le code suivant permet de convertir le résultat reçu du fichier PHP en un objet JavaScript, ou dans ce cas, un tableau JavaScript:

```
...
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    myObj = JSON.parse(this.responseText);
    for (x in myObj) {
      txt += myObj[x].nom + "<br>";
    }
    document.getElementById("demo").innerHTML = txt;
  }
};
...
```

- **Explication :** Cet exemple permet de :
  - Convertir la requête en objet, en utilisant la fonction PHP json_decode () .
  - Accéder à la base de données et remplir un tableau avec les données reçues.
  - Ajouter le tableau à un objet et renvoyez l'objet au format JSON à l'aide de la fonction json_encode ()

JSON avec PHP/MySQL

Couplage à une base de données MySQL

- **Envoi de la requête avec la méthode POST :**

  Dans l'exemple précédent la méthode GET est utilisée lors de l'envoi de la requête, mais il est souvent préférable d'utiliser la méthode HTTP POST. Dans ce cas, on spécifie la méthode et l'en-tête correcte, puis on met les données à envoyer comme argument de la méthode send() :

```
obj = { "table":"clients", "limit":10 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    myObj = JSON.parse(this.responseText);
    for (x in myObj) {
      txt += myObj[x].nom + "<br>";
    }
    document.getElementById("demo").innerHTML = txt;
  }
};
xmlhttp.open("POST", "json_demo_db_post.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```

JSON avec PHP/MySQL

Couplage à une base de données MySQL

- **Traitement de la requête envoyée avec la méthode POST :**

  La seule différence dans le fichier PHP est la méthode utilisée pour récupérer les données transférées.

```php
<?php
header("Content-Type: application/json; charset=UTF-8");
$obj = json_decode($_POST["x"], false);

$conn = new mysqli("myServer", "myUser", "myPassword", "myDB");
$stmt = $conn->prepare("SELECT nom FROM ? LIMIT ?");
$stmt->bind_param("ss", $obj->table, $obj->limit);
$stmt->execute();
$result = $stmt->get_result();
$outp = $result->fetch_all(MYSQLI_ASSOC);

echo json_encode($outp);
?>
```

La technique JSONP

Qu'est ce que JSONP ?

- Demander un fichier à partir d'un autre domaine peut entraîner des problèmes, en raison des règles inter-domaines.
- Alors que la demande d'un script externe à partir d'un autre domaine n'a pas ce problème.
- C'est cette possibilité qui est utilisé dans JSONP qui signifie JSON with Padding.
- Donc JSONP est une technique JavaScript pour envoyer des données JSON sans se soucier des problèmes inter-domaines.
- Elle n'utilise pas l' objet XMLHttpRequest mais la balise <script> à la place.

```
<script src="demo_jsonp.php">
```

La technique JSONP

Le fichier du serveur

- Le fichier sur le serveur encapsule le résultat dans un appel de fonction:

```
<?php
$myJSON = '{ "nom":"Djiby", "age":30, "ville":"Dakar" }';
echo "myFunc(".$myJSON.");";
?>
```

- Le résultat renvoie un appel à une fonction nommée "myFunc" avec les données JSON comme paramètre.

- Il faut s'assurer que la fonction existe sur le client.

Le fichier dans le client

- La fonction nommée "myFunc" est située sur le client, et prête à gérer les données JSON:

```
function myFunc(myObj) {
  document.getElementById("demo").innerHTML = myObj.nom;
}
```

La technique JSONP

Création à la volée d'une balise de script

- L'exemple précédent exécute la fonction "myFunc" lors du chargement de la page, en fonction de l'endroit où la balise de script est placée, ce qui n'est pas très satisfaisant.
- La balise de script ne doit être créée qu'en cas de besoin.
- Exemple : (Création et insertion de la balise <script> lorsqu'un bouton est cliqué)

```
<button onclick="clickButton()">Click sur moi!</button>
<p id="demo"></p>

<script>
function clickButton() {
  var s = document.createElement("script");
  s.src = "demo_jsonp.php";
  document.body.appendChild(s);
}

function myFunc(myObj) {
  document.getElementById("demo").innerHTML = myObj.nom;
}
</script>
```

La technique JSONP

Résultat JSONP dynamique

- Le code suivant rend l'exemple dynamique en envoyant les données JSON au fichier php et laisse le fichier php renvoyer un objet JSON en fonction des informations qu'il obtient.

```php
<?php
header("Content-Type: application/json; charset=UTF-8");
$obj = json_decode($_GET["x"], false);

$conn = new mysqli("myServer", "myUser", "myPassword", "myDB");
$result = $conn->query("SELECT nom FROM ".$obj->$table." LIMIT ".$obj->$limit);
$outp = array();
$outp = $result->fetch_all(MYSQLI_ASSOC);
echo "myFunc(".json_encode($outp).")";
?>
```

Explication : L'exemple ci-dessus permet de :

- Convertir la requête en objet, en utilisant la fonction PHP json_decode () .
- Accéder à la base de données et remplir un tableau avec les données obtenues.
- Ajouter le tableau à un objet.
- Convertir le tableau en JSON à l'aide de la fonction json_encode () .
- Envelopper "myFunc ()" autour de l'objet de retour.

La technique JSONP

Résultat JSONP dynamique

- Voici le code de la fonction "myFunc" qui est appelée à partir du fichier php :

```
function clickButton() {
  var obj, s
  obj = { table: "produits", limit: 10 };
  s = document.createElement("script");
  s.src = "jsonp_demo_db.php?x=" + JSON.stringify(obj);
  document.body.appendChild(s);
}
function myFunc(myObj) {
  var x, txt = "";
  for (x in myObj) {
    txt += myObj[x].nom + "<br>";
  }
  document.getElementById("demo").innerHTML = txt;
}
```

La technique JSONP

Fonction de rappel

- Lorsque vous n'avez aucun contrôle sur le fichier serveur, comment faire en sorte que le fichier du serveur appelle la fonction correcte?

- Parfois, le fichier serveur propose une fonction de rappel en paramètre:

```
function clickButton() {
  var s = document.createElement("script");
  s.src = "jsonp_demo_db.php?callback=myDisplayFunction";
  document.body.appendChild(s);
}
```

- Le fichier php appellera la fonction passée en tant que paramètre de rappel: