Gestion des données JSON en PHP

JSON (JavaScript Object Notation) est un format léger et facilement lisible pour échanger des données entre un serveur et un client.

PHP fournit des fonctions intégrées comme `json_encode` et `json_decode` pour manipuler les données en JSON.

1 - Présentation générale

- JSON est utilisé pour représenter des objets JavaScript sous forme de chaînes de texte.
- En PHP, on convertit des tableaux ou des objets en JSON avec `json_encode`, et on transforme des chaînes JSON en tableaux ou objets PHP avec `json_decode`.

Exemple de données JSON :

```json
{
 "nom": "Alice",
 "age": 25,
 "email": "alice@email.com"
}
```

Fonction `json_encode`

Permet de convertir des tableaux ou objets PHP en chaîne JSON. C'est souvent utilisé pour envoyer des données structurées à un client via une API.

Exemple 1 : Convertir un tableau PHP en JSON

```php
$data = [
 "nom" => "Alice",
 "age" => 25,
 "email" => "alice@email.com"
];

$json = json_encode($data);
echo $json; // {"nom":"Alice","age":25,"email":"alice@email.com"}
```

2.1 - Options utiles pour `json_encode`

2.1.1 - `JSON_PRETTY_PRINT`

Formate le JSON pour le rendre lisible (indentation).

Exemple :

```php
echo json_encode($data, JSON_PRETTY_PRINT);
/*
   {
      "nom": "Alice",
      "age": 25,
      "email": "alice@email.com"
   }
   */
```

2.1.2 - `JSON_UNESCAPED_UNICODE`

Empêche l'échappement des caractères Unicode. Utile pour conserver les accents et caractères spéciaux lisibles.

Exemple :

```php
$data = ["texte" => "C'était génial!"];
echo json_encode($data, JSON_UNESCAPED_UNICODE);
// {"texte":"C'était génial!"}
```

2.1.3 - Combiner plusieurs options

On peut combiner plusieurs options avec l'opérateur bitwise `|` .

Exemple :

```php
echo json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
```

Fonction `json_decode`

> Transforme une chaîne JSON en tableau ou objet PHP.
>
> Cette fonction est utile pour traiter des données reçues via une API ou un formulaire JSON.

Exemple 1 : JSON en tableau PHP

```php
$json = '{"nom":"Alice","age":25,"email":"alice@email.com"}';
$data = json_decode($json, true); // true pour tableau associatif
print_r($data);
/*
   Array
   (
      [nom] => Alice
      [age] => 25
      [email] => alice@email.com
   )
*/
```

Exemple 2 : JSON en objet PHP

```php
$data = json_decode($json); // Pas de true, données sous forme d'objet
echo $data->nom; // Affiche "Alice"
```

3.1 - Tester et gérer les erreurs

3.1.1 - Détecter les erreurs avec `json_last_error`

Si `json_encode` ou `json_decode` échoue, vous pouvez récupérer des informations sur l'erreur avec `json_last_error()` .

Exemple :

```php
$json = '{"nom":"Alice","age":25";"email":alice@email.com}'; // Erreur JSON
$data = json_decode($json);

if (json_last_error() !== JSON_ERROR_NONE) {
  echo "Erreur JSON : " . json_last_error_msg();
}
```

3.2 - Décodage partiel de JSON

Récupérer uniquement certaines informations depuis une chaîne JSON en la décodant.

Exemple :

```php
$json = '{"nom":"Alice","details":{"age":25,"email":"alice@email.com"}}';
$data = json_decode($json, true);
echo $data['details']['email']; // alice@email.com
```

Cas pratiques

4.1 - Travailler avec des fichiers JSON

4.1.1 - Écrire JSON dans un fichier

Enregistrement d'un tableau sous forme de fichier JSON.

```php
$data = [
 "produit" => "Livre",
 "prix" => 19.99,
 "stock" => 20
];

file_put_contents('produit.json', json_encode($data, JSON_PRETTY_PRINT));
```

4.1.2 - Lire JSON depuis un fichier

Chargement et traitement des données.

```php
$json = file_get_contents('produit.json');
$data = json_decode($json, true);
print_r($data);
```

4.2 - Filtrer des données JSON

On peut manipuler des données après décodage.

Exemple pour extraire uniquement certaines informations :

```php
$json = '[{"nom":"Alice","age":25},{"nom":"Bob","age":30}]';
$data = json_decode($json, true);

foreach($data as $personne) {
 echo "Nom : " . $personne['nom'] . ", Âge : " . $personne['age'] . "\n";
}
```

À RETENIR

Fonction / Option	Description	Exemple
`json_encode`	Encode un tableau ou objet PHP en JSON	`json_encode($data);`
`JSON_PRETTY_PRINT`	Formate le JSON pour améliorer la lisibilité	`json_encode($data, JSON_PRETTY_PRINT);`
`JSON_UNESCAPED_UNICODE`	N'échappe pas les caractères Unicode (conserve les accents)	`json_encode($data, JSON_UNESCAPED_UNICODE);`
`json_decode`	Décode une chaîne JSON en tableau (avec `true`) ou en objet (par défaut)	`$data = json_decode($json, true);`
`json_last_error`	Vérifie les erreurs après encodage ou décodage	`json_last_error_msg();`
Lire un fichier JSON	Charger un JSON depuis un fichier pour le traiter	`$json = file_get_contents('file.json');`
Écrire un fichier JSON	Sauvegarder un tableau PHP sous forme de fichier	`file_put_contents('file.json', json_encode($data));`

5.1 - Conseils pratiques :

1. Vérifiez toujours les erreurs avec `json_last_error()` après décodage.
2. Utilisez `JSON_PRETTY_PRINT` pour un formatage lisible lors du débogage ou des exports.
3. Combinez `json_encode` et `json_decode` avec les fonctions de fichiers (`file_get_contents` , `file_put_contents`) pour manipuler efficacement des fichiers JSON dans vos projets.