PHP et JSON : lire et écrire des données

PHP et JSON

Pourquoi JSON sans BDD ?

Avant de connecter une base de données, on peut **stocker des données dans des fichiers JSON**.
C'est parfait pour :

- Des petits projets (moins de 1000 entrées)
- Apprendre sans installer MySQL
- Des formulaires de contact, des listes, des configurations

C'est quoi JSON ?

JSON = JavaScript Object Notation. C'est un format texte universel pour stocker des données structurées.

```
[
    { "id": 1, "nom": "Alice", "email": "alice@test.be" },
    { "id": 2, "nom": "Bob",   "email": "bob@test.be" }
]
```

Un fichier JSON = du texte. PHP peut le lire et l'écrire.

PHP ↔ JSON : les deux fonctions clés

json_encode() — PHP vers JSON

```php
<?php
// Un tableau PHP associatif
$personne = [
    "nom"   => "Alice",
    "age"   => 17,
    "ville" => "Namur"
];

// On le convertit en chaîne JSON
$json = json_encode($personne);
echo $json;
// {"nom":"Alice","age":17,"ville":"Namur"}

// JSON bien indenté (pour la lisibilité)
echo json_encode($personne, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
```

json_decode() — JSON vers PHP

```php
<?php
$json = '{"nom":"Alice","age":17,"ville":"Namur"}';

// true = retourne un tableau associatif (recommandé)
$personne = json_decode($json, true);

echo $personne["nom"];    // Alice
echo $personne["age"];    // 17
```

Sans le `true`, json_decode retourne un objet. Avec `true` → tableau. **Toujours mettre `true`.**

Lire et écrire des fichiers

file_get_contents() — Lire un fichier

```php
<?php
// Lit tout le contenu du fichier et le retourne comme une chaîne
$contenu = file_get_contents("data/contacts.json");

// On convertit la chaîne JSON en tableau PHP
$contacts = json_decode($contenu, true);
```

file_put_contents() — Écrire dans un fichier

```php
<?php
// Notre tableau de données
$contacts = [
    ["nom" => "Alice", "email" => "alice@test.be"],
    ["nom" => "Bob",   "email" => "bob@test.be"],
];

// On convertit en JSON et on écrit dans le fichier
// Le fichier est créé s'il n'existe pas, remplacé s'il existe
file_put_contents("data/contacts.json", json_encode($contacts, JSON_PRETTY_PRINT));
```

Pattern complet : lire, modifier, réécrire

C'est le pattern qu'on utilisera pour tout : ajouter, modifier, supprimer.

```php
<?php

$fichier = "data/contacts.json";

// 1. LIRE — si le fichier n'existe pas, on part d'un tableau vide
```

```php
if (file_exists($fichier)) {
    $contacts = json_decode(file_get_contents($fichier), true);
} else {
    $contacts = [];
}

// 2. MODIFIER — ajouter un nouveau contact
$contacts[] = [
    "id"    => time(),              // timestamp comme identifiant unique
    "nom"   => "Charlie",
    "email" => "charlie@test.be"
];

// 3. RÉÉCRIRE — tout le tableau mis à jour
file_put_contents($fichier, json_encode($contacts, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE)

echo "Contact ajouté !";
```

`$contacts[] = [...]` ajoute un élément à la fin du tableau, comme `push` en JavaScript.

Le dossier data/

Par convention, les fichiers de données vont dans un sous-dossier `data/`.

```
site-modulaire/
├── data/
│   └── contacts.json     ← créé automatiquement par PHP
├── includes/
├── index.php
└── ...
```

⚠️ Crée le dossier `data/` manuellement. PHP peut créer des fichiers dans un dossier existant, mais pas créer le dossier lui-même (sauf avec `mkdir()`).

👉 Prochaine étape

Exercice : lire et afficher un fichier JSON →