Le templating natif en PHP

> Le templating natif en PHP permet de mélanger du code PHP avec du HTML de manière simple et élégante grâce à une syntaxe simplifiée.
>
> Ce style améliore la lisibilité et diminue la complexité dans la structure des fichiers.

1 - Introduction

Lorsque vous développez des applications dynamiques, vous mélangez souvent HTML et PHP pour afficher des données.

Cependant, utiliser des structures de contrôle classiques (`if` , `for` , etc.) avec des accolades `{}` et des balises PHP peut mener à un code difficile à lire. En utilisant le templating natif en PHP, vous pouvez simplifier considérablement la présentation.

Exemple, code classique :

```php
 <?php
if ($isConnected) {
    echo '<p>Bonjour !</p>';
} else {
    echo '<p>Veuillez vous connecter.</p>';
}
?>
```

Vers un code natif plus lisible :

```
 <?php if ($isConnected): ?>
    <p>Bonjour !</p>
<?php else: ?>
    <p>Veuillez vous connecter.</p>
<?php endif; ?>
```

Les structures de contrôle natives

2.1 - Les conditions : `if` , `else` , `elseif` , `endif`

Les conditions affichent des parties de votre page HTML en fonction de critères définis.

Exemple :

```
 <?php if ($isConnected): ?>
    <p>Bienvenue, utilisateur connecté !</p>
<?php else: ?>
    <p>Veuillez vous connecter.</p>
<?php endif; ?>
```

2.2 - Boucles : `for` , `endfor`

Avec `for` , vous pouvez afficher dynamiquement plusieurs éléments HTML.

Exemple : Génération d'une liste numérotée :

```
 <ul>
    <?php for ($i = 1; $i <= 5; $i++): ?>
        <li>Élément numéro <?php echo $i; ?></li>
    <?php endfor; ?>
</ul>
```

2.3 - Boucles : `while` , `endwhile`

La boucle `while` exécute un bloc tant qu'une condition est remplie.

Exemple : Afficher des nombres pairs :

```
 <?php
$i = 2;
?>
<ul>
   <?php while ($i <= 10): ?>
      <li><?php echo $i; ?></li>
      <?php $i += 2; ?>
   <?php endwhile; ?>
</ul>
```

2.4 - Boucles : `foreach` , `endforeach`

`foreach` est idéale pour parcourir des tableaux ou afficher des collections.

Exemple : Liste des utilisateurs :

```
 <?php $utilisateurs = ["Alice", "Bob", "Charlie"]; ?>
<ul>
   <?php foreach ($utilisateurs as $user): ?>
      <li><?php echo $user; ?></li>
   <?php endforeach; ?>
</ul>
```

Comparaison : Code classique VS Templating natif

3.1 - Code classique

Voici un exemple de génération d'une liste avec du HTML et PHP classique :

```
 <?php
echo "<ul>";
for ($i = 1; $i <= 3; $i++) {
   echo "<li>Élément $i</li>";
}
echo "</ul>";
?>
```

3.2 - Code avec templating natif

Le même code réécrit de manière plus lisible avec le templating natif :

```
 <ul>
   <?php for ($i = 1; $i <= 3; $i++): ?>
      <li>Élément <?php echo $i; ?></li>
   <?php endfor; ?>
</ul>
```

3.3 - Analyse

- Classique : Le code PHP est entremêlé au HTML, ce qui le rend difficile à lire.
- Templating natif : Les blocs HTML sont immédiatement lisibles, et la logique PHP reste structurée.

Avantages pour la lisibilité HTML

1. Lecture intuitive : La séparation visuelle entre les blocs PHP et HTML permet une meilleure organisation.
2. Moins de syntaxe encombrante : Plus besoin de jongler avec des `<?php` ouverts/fermés fréquemment.
3. Concentration sur le HTML : Les développeurs front-end peuvent facilement comprendre et modifier le code.
4. Réduction des erreurs : Une syntaxe plus propre entraîne moins d'erreurs de coquilles (comme les balises fermées ou accolades oubliées).

Cas pratique : Génération d'un tableau HTML

Voici un tableau HTML généré dynamiquement :

```
 <?php
$data = [
   ["Nom" => "Alice", "Âge" => 25],
   ["Nom" => "Bob", "Âge" => 30],
   ["Nom" => "Charlie", "Âge" => 35]
];
?>
<table border="1">
   <thead>
      <tr>
         <th>Nom</th>
         <th>Âge</th>
      </tr>
   </thead>
   <tbody>
      <?php foreach ($data as $entry): ?>
         <tr>
            <td><?php echo $entry["Nom"]; ?></td>
            <td><?php echo $entry["Âge"]; ?></td>
         </tr>
      <?php endforeach; ?>
   </tbody>
</table>
```

À RETENIR

- Conditions :
  - Exemple :

```php
<?php if ($isConnected): ?>
  <p>Bienvenue</p>
<?php else: ?>
  <p>Veuillez vous connecter.</p>
<?php endif; ?>
```

- Boucles :
  - for:endfor

```php
<?php for ($i = 1; $i <= 3; $i++): ?>
  <p><?php echo $i; ?></p>
<?php endfor; ?>
```

  - while:endwhile

```php
<?php while ($i <= 10): ?>
  <p><?php echo $i++; ?></p>
<?php endwhile; ?>
```

  - foreach:endforeach

```php
<?php foreach ($list as $item): ?>
  <p><?php echo $item; ?></p>
<?php endforeach; ?>
```

- **Avantages** :
    1. Mieux structuré, plus lisible.
    2. Parfait pour des développeurs front-end et full-stack.
    3. Réduction des erreurs liées à la syntaxe encombrante.

Adopter le templating natif est une manière efficace d'organiser son code PHP tout en offrant une visibilité claire sur le contenu de vos pages. Cela rend vos projets dynamiques plus maintenables et professionnels !