Formation Développement Web : Backend : PHP

Partie : 1

I. Backend

A. Introduction

Le "backend" (en français, "côté serveur") fait référence à la partie d'une application web ou logicielle qui s'exécute du côté du serveur. Il est responsable du traitement des demandes des utilisateurs, de la gestion des données, de la logique métier et de la communication avec la base de données, le cas échéant. Le backend travaille en arrière-plan pour rendre une application web fonctionnelle et répondre aux besoins des utilisateurs.

B. Langage de programmation Backend

Pour le développement du backend, il existe une multitude de langages de programmation parmi lesquels vous pouvez faire votre choix. Chacun de ces langages possède ses avantages et inconvénients spécifiques. Parmi les options disponibles, on compte PHP, JavaScript, Java, C#, Python, C, etc.

Dans le cadre de cette formation, nous mettrons l'accent sur l'utilisation de **PHP**.

II. PHP

A. Introduction PHP

PHP, est l'un des langages de programmation les plus populaires et largement utilisés pour le développement web. Il a été spécialement conçu pour créer des pages web dynamiques et interactives. Dans ce cours, nous allons explorer les bases de PHP et comment l'utiliser pour développer des site web.

B. Pourquoi PHP

- **Gratuit et Open Source** : PHP est un langage open source, ce qui signifie qu'il est gratuit à télécharger, à utiliser et à modifier selon vos besoins.
- **Large communauté** : PHP dispose d'une grande communauté de développeurs à travers le monde, ce qui facilite l'accès à de nombreuses ressources et bibliothèques.
- **Facilité d'apprentissage** : PHP est relativement simple à apprendre, en particulier pour les débutants en programmation, grâce à une syntaxe claire et concise.
- **Polyvalence** : PHP peut être utilisé pour développer une variété d'applications web, des sites web statiques aux applications web complexes.

III. Syntaxe PHP

A. Balise PHP

En PHP, vous utilisez des balises spéciales pour encadrer votre code PHP. Les balises de base sont **<?php** pour ouvrir une section PHP et **?>** pour la fermer.

1. Exemple

```
<?php
   // Votre code PHP va ici
?>
```

B. Déclaration des variables

En PHP, vous pouvez déclarer une variable en utilisant le signe **'$'**. Les noms de variables sont sensibles à la casse et commencent généralement par une lettre ou un tiret bas _. PHP prend en charge divers types de données.

1. Exemple

```
<?php
    $nombre = 42;
    $pi = 3.14;
    $nom = "John";
    $estVrai = true;
?>
```

C. Opération numérique

En PHP, vous pouvez effectuer diverses opérations arithmétiques pour manipuler les valeurs numériques. Pour illustrer ces opérations, nous allons utiliser une variable **$resultat** initialisée à 0 pour les exemples suivants :

```
<?php
    $resultat = 0;
?>
```

1. Addition « + »

```
<?php
    $num1 = 5;
    $num2 = 7;
    $resultat = $num1 + $num2; // $resultat vaut 12
?>
```

2. Soustraction « - »

```
<?php
    $num1 = 16;
    $num2 = 6;
    $resultat = $num1 - $num2; // $resultat vaut 10
?>
```

3. Multiplication « * »

```
<?php
    $num1 = 6;
    $num2 = 5;
    $resultat = $num1 * $num2; // $resultat vaut 30
?>
```

4. Division « / »

```
<?php
    $num1 = 10;
    $num2 = 5;
    $resultat = $num1 / $num2; // $resultat vaut 2
?>
```

5. Modulo (reste de la division) « % »

```
<?php
    $num1 = 7;
    $num2 = 4;
    $resultat = $num1 % $num2; // $resultat vaut 3
?>
```

6. Puissance (Exposant) « ** »

```
<?php
    $num1 = 5;
    $num2 = 2;
    $resultat = $num1 ** $num2; // $resultat vaut 25
?>
```

D. Opérateurs d'affectation

En PHP, les opérateurs d'affectation permettent d'attribuer des valeurs aux variables avec diverses opérations simplifiées. Voici comment utiliser ces opérateurs en PHP :

1. Addition avec affectation « += »

```
<?php
    $num = 5;
    $num += 7; // Cette opération équivaut à "$num = $num + 7" et la valeur de `$num` est maintenant : 12
?>
```

2. Soustraction avec affectation « -= »

```
<?php
    $num = 7;
    $num -= 1; // Cette opération équivaut à "$num = $num - 1" et la valeur de `$num` est maintenant : 6
?>
```

3. Multiplication avec affectation « *= »

```
<?php
    $num = 5;
    $num *= 6; // Cette opération équivaut à "$num = $num * 6" et la valeur de `$num` est maintenant : 30
?>
```

4. Division avec affectation « /= »

```
<?php
    $num = 6;
    $num /= 2; // Cette opération équivaut à "$num = $num / 2" et la valeur de `$num` est maintenant : 3
?>
```

5. Modulo avec affectation « += »

```
<?php
    $num = 5;
    $num %= 3; // Cette opération équivaut à "$num = $num % 3" et la valeur de `$num` est maintenant : 2
?>
```

6. Puissance avec affectation « **= »

```
<?php
    $num = 3;
    $num **= 2; // Cette opération équivaut à "$num = $num ** 2" et la valeur de `$num` est maintenant : 9
?>
```

7. Incrémentation « ++ »

```
<?php
    $num = 5;
    $num ++; // Cette opération équivaut à "$num = $num + 1" et la valeur de `$num` est maintenant : 6
?>
```

8. Décrémentation «--»

```
<?php
    $num = 6;
    $num --; // Cette opération équivaut à "$num = $num - 1" et la valeur de `$num` est maintenant : 5
?>
```

E. Opérateurs de comparaison en PHP

En PHP, les opérateurs de comparaison évaluent des expressions pour déterminer si elles sont vraies ou fausses en comparant les valeurs et les types de données. Pour illustrer ces opérations, nous allons utiliser une variable **'$resultat'** initialisée à false pour les exemples suivants :

```
<?php
    $resultat = false;
?>
```

1. Égal à (comparaison des valeurs) « == »

Cet comparaison compare la valeur de deux variables, dans l'exemple ci-dessous résultat contiendra 'true' car même si **'$var1'** est de type numérique et **'$var2'** de type string mais les deux contient le même valeur

```
<?php
    $var1 = 5;
    $var2 = "5";
    $resultat = ($var1 == $var2); // $resultat vaut True
?>
```

2. Égal à (comparaison de la valeur et du type) « === »

Cet comparaison compare la valeur de deux variables et aussi leur type, dans l'exemple ci-dessous résultat contiendra 'false' car **'$var1'** et **'$var2'** ont la même valeur mais pas le même type de variable

```
<?php
    $var1 = 5;
    $var2 = "5";
    $resultat = ($var1 === $var2); // $resultat vaut False
?>
```

3. Différent de (Pas la même valeur) « != »

Cet comparaison compare la valeur de deux variables, dans l'exemple ci-dessous résultat contiendra 'true' car **'$var1'** et **'$var2'** ont pas la même valeur

```
<?php
    $var1 = 12;
    $var2 = 2 ;
    $resultat = ($var1 != $var2); // $resultat vaut True
?>
```

4. Différent de (Pas la même valeur ou le même type) « !== »

Cet comparaison compare la valeur de deux variables et leur type , dans l'exemple ci-dessous résultat contiendra 'true' car même si **'$var1'** et **'$var2'** ont la même valeur mais pas le même type

```
<?php
    $var1 = 12;
    $var2 = "12";
    $resultat = ($var1 !== $var2); // $resultat vaut True
?>
```

5. Supérieur à « > »

Cet comparaison compare la valeur de deux variables, dans l'exemple ci-dessous résultat contiendra 'true' car **'$var1'** est supérieur de **'$var2'**

```
<?php
    $var1 = 12;
    $var2 = 2 ;
    $resultat = ($var1 > $var2); // $resultat vaut True
?>
```

6. Inférieur à « < »

Cet comparaison compare la valeur de deux variables, dans l'exemple ci-dessous résultat contiendra 'true' car **'$var1'** est inférieur de **'$var2'**

```
<?php
    $var1 = 6;
    $var2 = 9;
    $resultat = ($var1 < $var2); // $resultat vaut True
?>
```

7. Supérieur ou égal « >= »

Cet comparaison compare la valeur de deux variables, dans l'exemple ci-dessous résultat contiendra 'true' car **'$var1'** est supérieur de **'$var2'**

```
<?php
    $var1 = 12;
    $var2 = 2 ;
    $resultat = ($var1 >= $var2); // $resultat vaut True
?>
```

8. Inférieur ou égal« <= »

Cet comparaison compare la valeur de deux variables, dans l'exemple ci-dessous résultat contiendra 'true' car **'$var1'** est égal à **'$var2'**

```
<?php
    $var1 = 6;
    $var2 = 6;
    $resultat = ($var1 <= $var2); // $resultat vaut True
?>
```

F. Opérateurs logiques en PHP

En PHP, les opérateurs logiques combinent ou inversent des valeurs booléennes pour évaluer des conditions complexes et prendre des décisions basées sur la logique. Pour illustrer ces opérations, nous allons utiliser une variable **'$resultat'** initialisée à `false` pour les exemples suivants :

```
<?php
    $resultat = false;
?>
```

1. ET Logique « && »

L'opérateur logique "ET" renvoie vrai si les deux expressions sont vraies, sinon il renvoie faux.

Expression 1	Expression 2	Opération logique	Résulta
True	True	Expression1 && Expression2	True
True	False		False
False	True		False
False	False		False

a) Exemple1

```
<?php
    $v1 = 5;
    $v2 = "5";
    $v3 = "text";
    $v4 = 0;
    $c1 = ($v1 == $v2); // $c1 vaut True
    $c2 = ($v3 != $v4); // $c2 vaut True
    $resultat = ($c1 && $c2); // $resultat vaut True car $c1 et $c2 sont tous les deux True
?>
```

b) Exemple2

```
<?php
    $v1 = 5;
    $v2 = "5";
    $v3 = "text";
    $v4 = 0;
    $c1 = ($v1 == $v2); // $c1 vaut True
    $c2 = ($v3 == $v4); // $c2 vaut False
    $resultat = ($c1 && $c2); // $resultat vaut False car $c1 est True mais $c2 est False
?>
```

2. OU Logique « || »

L'opérateur logique "OU" renvoie vrai si l'un ou les deux expressions sont vraies, sinon il renvoie faux.

Expression 1	Expression 2	Opération logique	Résulta
True	True	Expression1|| Expression2	True
True	False		True
False	True		True
False	False		False

a) Exemple1

```
<?php
    $v1 = 5;
    $v2 = "5";
    $v3 = "text";
    $v4 = 0;
    $c1 = ($v1 == $v2); // $c1 vaut True
    $c2 = ($v3 == $v4); // $c2 vaut False
    $resultat = ($c1 || $c2); // $resultat vaut True car $c1 est True et $c2 est False
?>
```

b) Exemple2

```
<?php
    $v1 = 5;
    $v2 = "5";
    $v3 = "text";
    $v4 = 0;
    $c1 = ($v1 != $v2); // $c1 vaut False
    $c2 = ($v3 == $v4); // $c2 vaut False
    $resultat = ($c1 || $c2); // $resultat vaut False car $c1 et $c2 sont False
?>
```

3. NON Logique(Inversseur) « ! »

L'opérateur logique "NON" renvoie vrai si l'expression est fausse , et faut si l'expression est vrai

Expression 1	Opération logique	Résulta
True	!Expression1	False
False		True

a) Exemple1

```
<?php
    $v1 = 5;
    $v2 = "5";
    $c1 = ($v1 == $v2); // $c1 vaut True
    $resultat = (!$c1); // $resultat vaut False car $c1 est True
?>
```

b) Exemple2

```
<?php
    $v3 = "text";
    $v4 = 0;
    $c2 = ($v3 == $v4); // $c2 vaut False
    $resultat = (!$c2); // $resultat vaut True car $c2 est False
?>
```

G. Afficher un texte sur une page web

Pour afficher les résultats de vos opérations sur une page web à l'aide de PHP, vous pouvez utiliser la fonction **'echo'** pour générer du contenu HTML contenant les résultats.

1. Exemple

```
<?php
    echo "Bonjour tout le monde";
?>
```