Les conditions if, if...else et if...elseif...else PHP

Cours complet PHP et MySQL

INTRODUCTION AU COURS PHP ET MYSQL

1. Introduction au cours : définitions et rôles du PHP et du MySQL

2. Client et serveur : définitions et interactions

3. Mise en place de notre environnement de travail

4. Créer, enregistrer et exécuter un script PHP

5. Afficher un résultat en PHP avec une instruction echo ou print

DÉCOUVERTE DES VARIABLES EN PHP

Nous savons désormais utiliser les opérateurs de comparaison. Il est donc temps d'entrer dans le vif du sujet et d'apprendre à créer nos premières structures conditionnelles.

Dans cette nouvelle leçon, nous allons étudier les structures de contrôle conditionnelles if, if...else et if... elseif... else.

La condition if en PHP

La structure de contrôle conditionnelle, ou plus simplement « condition » if est l'une des plus importantes et des plus utilisées dans l'ensemble des langages de programmation utilisant les structures de contrôle et en particulier en PHP.

La condition if est également la plus simple, puisqu'elle va nous permettre d'exécuter un bloc de code si et seulement si le résultat d'un test vaut true.

Créons immédiatement nos premières conditions if :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            $x = 4; //On affecte la valeur 4 à $x
            $y = 2; //On affecte la valeur 2 à $y

            if($x > 1){
                echo '$x contient une valeur supérieure à 1';
            }

            if($x == $y){
                echo '$x et $y contiennent la même valeur';
            }
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```

8. Opérateurs et concaténation en PHP

LES STRUCTURES DE CONTRÔLE EN PHP

9. Présentation des conditions et des opérateurs de comparaison

10. Les conditions if, if...else et if...elseif...else

11. Créer des conditions robustes avec les opérateurs logiques

12. Ecrire des conditions condensées avec les opérateurs ternaire et fusion null

13. L'instruction switch en PHP

14. Les boucles PHP et les opérateurs d'incrémentation et de décrémentation

15. Inclure des fichiers dans un autre en PHP avec include et require

DÉCOUVERTE DES FONCTIONS EN PHP

16. Introduction aux fonctions PHP

17. Contrôler le passage des arguments

18. Contrôler les valeurs de retour d'une fonction

19. La portée des variables en PHP

20. Constantes et constantes magiques en PHP

LES VARIABLES TABLEAUX EN PHP

21. Présentation des tableaux et tableaux numérotés en PHP

22. Les tableaux associatifs en PHP

23. Les tableaux multidimensionnels en PHP

MANIPULER DES DATES EN PHP

24. Le timestamp UNIX et la date en PHP

25. Obtenir et formater une date en PHP

26. Comparer des dates et tester la validité d'une date en PHP

Ici, j'ai créé deux conditions if avec une comparaison qui va être évaluée à true et une autre qui va être évaluée à false.

Commencez par noter la syntaxe de nos conditions qui est la suivante : if(test){code à exécuter si le test est validé}. Il convient de différencier ici « test » et « comparaison » : le test correspond à l'ensemble du code écrit dans les parenthèses et peut être composé de plusieurs comparaisons comme on va le voir par la suite.

Dans notre première condition, le résultat de la comparaison renvoyé par le PHP est true puisque notre variable $x stocke le chiffre 4 qui est bien supérieur à 1. Le code dans la condition est alors exécuté.

Dans notre deuxième condition, la comparaison est cette fois-ci évaluée à false car la valeur contenue dans $x n'est pas égale en valeur à la valeur contenue dans $y. Le code contenu dans la condition ne sera donc pas lu ni exécuté.

Inverser la valeur logique d'une condition

Dans les exemples ci-dessus, le code placé dans notre condition n'est exécuté que si le résultat de la comparaison est true.

Dans certaines situations, nous préférerons créer nos conditions de telle sorte à ce que le code dans la condition soit exécuté si le résultat de la comparaison est false.

Nous allons pouvoir faire cela de deux manières : soit en utilisant l'opérateur logique inverse ! que nous étudierons dans la leçon suivante, soit en comparant le résultat de notre comparaison à false.

Concentrons-nous pour le moment sur cette deuxième méthode qui demande de bien connaitre l'ordre de priorité des opérateurs et de bien comprendre l'associativité de ceux-ci.

Il y a deux choses à savoir ici :

- Les opérateurs <, <=, > et >= ont une priorité plus importante que les opérateurs ==, ===, !=, !==, <> et <> qui ont eux-mêmes une priorité plus importante que l'opérateur ??;
- Les opérateurs de comparaison sont non associatifs (à l'exception de l'opérateur ??

27. Les variables superglobales PHP

28. Création et gestion des cookies en PHP

29. Définir et utiliser les sessions en PHP

MANIPULER DES FICHIERS EN PHP

30. Introduction à la manipulation de fichiers en PHP

31. Ouvrir, lire et fermer un fichier en PHP

32. Créer et écrire dans un fichier en PHP

33. Autres opérations sur les fichiers en PHP

UTILISER LES EXPRESSIONS RÉGULIÈRES OU RATIONNELLES EN PHP

34. Introduction aux expressions rationnelles ou expressions régulières

35. Les fonctions PCRE PHP

36. Les classes de caractères des regex

37. Les métacaractères des regex PHP

38. Les options des expressions régulières disponibles en PHP

PROGRAMMATION ORIENTÉE OBJET (POO) PHP : CONCEPTS DE BASE

39. Introduction à la programmation orientée objet PHP : classes, instances et objets

40. Propriétés et méthodes en PHP orienté objet

41. Les méthodes PHP constructeur et destructeur

42. Encapsulation et visibilité des propriétés et méthodes PHP

43. Classes étendues et héritage en PHP orienté objet

44. Surcharge d'éléments et opérateur de résolution de portée en PHP

45. Les constantes de classe en PHP objet

droit d'écrire la comparaison 2 < 4 > 2 par exemple en PHP car les opérateurs < et > ont le même ordre de priorité et sont non associatifs.

Ainsi, si on écrit par exemple 2 < 4 == false, PHP va d'abord effectuer la comparaison 2 < 4 puis comparer le résultat de cette comparaison (qui est ici true car 2 est bien inférieur à 4) à false. Ici, true est différent de false et donc le test va finalement échouer.

Pour inverser la valeur logique d'une condition, c'est-à-dire pour exécuter le code de la condition uniquement lorsque notre première comparaison est évaluée à false, il suffit donc de comparer le résultat de cette première comparaison à la valeur false.

Si notre première comparaison n'est pas vérifiée et est évaluée à false, alors le test de notre condition va devenir if(false == false) ce qui va être finalement évalué à true et donc le code de notre condition va bien être exécuté !

Notez qu'on va également pouvoir utiliser les parenthèses pour forcer la priorité d'un opérateur sur un autre, ce qui peut être utile lorsqu'on souhaite utiliser plusieurs opérateurs de comparaison qui ont une priorité équivalente par défaut.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            $x = 4; //On affecte la valeur 4 à $x
            $y = 2; //On affecte la valeur 2 à $y

            if(($x <= 1) == false){
                echo '$x contient une valeur supérieure à 1';
            }

            if(($x != $y) == false){
                echo '$x et $y contiennent la même valeur';
            }
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```

47. Les méthodes et les classes abstraites en PHP objet

48. Les interfaces en PHP orienté objet

49. Les méthodes magiques en orienté objet PHP

PROGRAMMATION ORIENTÉE OBJET PHP : NOTIONS AVANCÉES

50. Le chainage de méthodes en PHP

51. Les closures et les classes anonymes en PHP objet

52. L'auto chargement des classes en PHP

53. Le mot clef final en PHP objet

54. La résolution statique à la volée ou late static bindings en PHP

55. Utiliser les traits en orienté objet PHP

56. L'interface Iterator et le parcours d'objets en PHP

57. Le passage d'objets en PHP : identifiants et références

58. Le clonage d'objets et la méthode magique PHP __clone()

59. La comparaison d'objets PHP

ESPACES DE NOMS, FILTRES ET GESTION DES ERREURS EN PHP

60. Les espaces de noms PHP

61. Présentation des filtres PHP

62. Filtres de validation, de nettoyage et drapeaux de l'extension PHP Filter

63. Utilisation pratique des filtres en PHP

64. Définition et gestion des erreurs en PHP

65. Déclenchement, capture et gestion des exceptions PHP : try, throw, catch

INTRODUCTION AUX BASES DE DONNÉES, AU SQL ET À MYSQL

66. Introduction aux bases de données, au SQL et au MySQL

Ici, notre variable $x stocke la valeur 4 qui est supérieure à 1. La partie $x <= 1 de notre première condition va donc être évaluée à false. On compare ensuite le résultat de la comparaison à false. On obtient donc ici if(false == false) qui va être évalué à true puisque c'est le cas et donc le code dans notre condition va bien être exécuté.

On utilise le même procédé pour notre deuxième condition que je vous laisse décortiquer seul pour que vous puissiez vous entrainer.

La condition if...else en PHP

Avec la condition if, nous restons relativement limités puisque cette condition nous permet seulement d'exécuter un bloc de code selon que le résultat d'un test soit évalué à true.

La structure conditionnelle if...else (« si... sinon » en français) va être plus complète que la condition if puisqu'elle va nous permettre d'exécuter un premier bloc de code si un test renvoie true ou un autre bloc de code dans le cas contraire.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            $x = 4; //On affecte la valeur 4 à $x
            $y = 2; //On affecte la valeur 2 à $y

            if($x > 1){
                echo '$x contient une valeur stric. supérieure à 1 <br>';
            }else{
                echo '$x contient une valeur inférieure ou égale à 1 <br>';
            }

            if($x == $y){
                echo '$x et $y contiennent la même valeur <br>';
            }else{
                echo '$x et $y contiennent des valeurs différentes <br>';
            }
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```

68. Se connecter à une base de données MySQL en PHP

69. Créer une base de données MySQL et une table dans la base

MANIPULER DES DONNÉES DANS DES BASES MYSQL AVEC PDO

70. Insérer des données dans une table MySQL

71. Les requêtes MySQL préparées avec PDO PHP

72. Modifier les données d'une table MySQL ou sa structure

73. Supprimer des données, une table ou une base de données MySQL

74. Sélection simple de données dans une table MySQL en PHP

75. Utiliser des critères de sélection pour sélectionner des données dans une table MySQL

76. Utiliser les fonctions d'agrégation et les fonctions scalaires SQL

JOINTURES, UNION ET SOUS REQUÊTES

77. Présentation des jointures SQL

78. Création de jointures SQL

79. L'opérateur SQL UNION

80. Les opérateurs de sous requête SQL

GESTION DES FORMULAIRES HTML AVEC PHP

81. Rappels sur les formulaires HTML

82. Récupérer et manipuler les données des formulaires HTML en PHP

83. Sécurisation et validation des formulaires en PHP

CONCLUSION DU COURS PHP ET MYSQL

84. Conclusion du cours complet PHP et MySQL

Notez la syntaxe de la condition if...else : on place notre comparaison et on effectue notre test dans le if mais pas dans le else.

En effet, la structure else est une structure qui a été créée pour prendre en charge tous les cas nos gérés précédemment. Ainsi, on ne précisera jamais de condition au sein d'un else puisque par défaut cette structure prend en charge tous les autres cas (tous les cas non gérés par le if ici).

Si le test de notre condition est validé, le code dans le if va s'exécuter et le code dans le else va alors être ignoré.

Si au contraire le test n'est pas validé alors le code dans le if va être ignoré et c'est le cas dans le else qui va être exécuté.

La condition if...elseif...else en PHP

La condition if...elseif...else (« si...sinon si...sinon ») est une structure conditionnelle encore plus complète que la condition if...else puisqu'elle va nous permettre cette fois-ci de générer autant de cas que l'on souhaite.

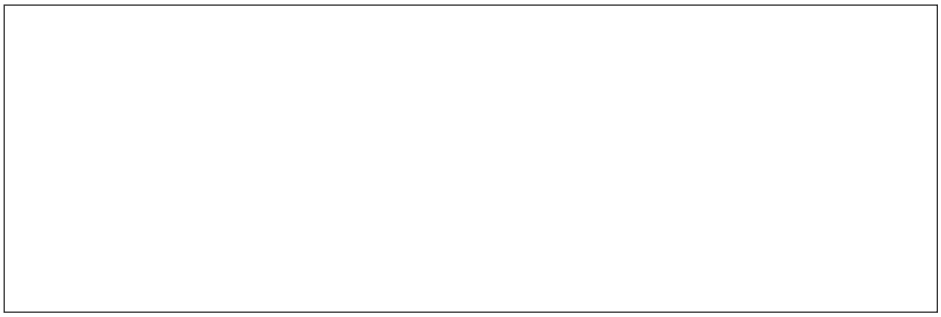
En effet, nous allons pouvoir écrire autant de elseif (en un seul mot, attention) que l'on veut dans notre condition if...elseif...else, chacun avec un test différent.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            $x = 4; //On affecte la valeur 4 à $x
            $y = 2; //On affecte la valeur 2 à $y

            if($x > 1){
```

```
19.
20.                 echo '$x contient une valeur stric. inférieure à 1 <br>';
21.             }
22.
23.             if($x == $y){
24.                 echo '$x et $y contiennent la même valeur <br>';
25.             }elseif($x < $y){
26.                 echo '$x contient une valeur stric. inférieure à $y <br>';
27.             }elseif($x > $y){
28.                 echo '$x contient une valeur stric. supérieure à $y <br>';
29.             }else{
30.                 echo 'Les valeurs de $x et $y n\'ont pas pu être comparées';
31.             }
32.         ?>
33.         <p>Un paragraphe</p>
34.     </body>
35. </html>
```



Comme vous pouvez le constater, les elseif occupent un rôle similaire au if de départ puisque chacun d'entre eux va posséder son propre test (qui est obligatoire).

Notez que dans le cas où plusieurs elseif possèdent un test qui va être évalué à true, seul le code du premier elseif rencontré sera exécuté. En effet, dès qu'un test va être validé, le PHP va ignorer les tests suivants.

Notez également qu'on devra toujours obligatoirement terminer notre condition if...elseif...else avec un else qui servira à gérer toutes les issues (ou les cas) non pris en charge par le if ou par les elseif.

Précédent Suivant

Laisser un commentaire

Vous devez vous connecter pour publier un commentaire.