Instructions de branchement et boucles

Transparent 1

Bonjour, dans cette vidéo nous allons aborder les structures de contrôle qui permettent de choisir les instructions à exécuter selon qu'une ou plusieurs conditions sont remplies ou pas.

Il s'agit des instructions de branchement, de tests et de boucle.

Transparent 2

Nous commencerons tout d'abord avec la structure Si-Alors-Sinon :

- elle se compose au minimum d'un bloc IF,
- contenant une expression entre parenthèses qui renvoie TRUE (vrai) ou FALSE (faux),
- elle est suivie des instructions à exécuter entre deux accolades.

Il est possible d'ajouter un ou plusieurs blocs ELSEIF après le bloc IF.

Les blocs sont alors traités dans l'ordre et uniquement si les précédents ont renvoyé FALSE.

Chaque bloc ELSEIF contient donc :

- une expression entre parenthèses qui renvoie TRUE (vrai) ou FALSE (faux),
- suivie des instructions à exécuter entre deux accolades.

Un bloc ELSE peut être ajouté après le bloc IF et d'éventuels blocs ELSEIF, il ne sera exécuté que si toutes les expressions précédentes renvoient FALSE.

Le bloc ELSE ne contient donc pas d'expression à évaluer, mais uniquement les instructions à exécuter entre accolades.

Dans l'exemple fourni, les variables *$nom* et *$connecte* sont déclarées et initialisées.

Sur la ligne 6, on teste la variable *$connecte*. Lorsque cette variable est vraie, la ligne 7 est exécutée.

Dans le cas contraire, la ligne 9 est testée (à savoir si la variable *$nom* est initialisée, c'est à dire ne vaut pas null) dans ce cas la ligne 10 est exécutée. Enfin, si le second test est également négatif, c'est alors la ligne 13 qui est exécutée.

Dans tous les cas, la ligne 15 affiche la valeur de *$message* qui a été mise à jour dans une des instructions de branchement précédentes.

Vous pouvez reproduire et tester cet exemple en changeant les valeurs des variables *$nom* et *$connecte*.

Transparent 3

Il existe une autre structure de contrôle, la structure SWITCH (Selon en français).

Elle permet de tester l'égalité entre une unique variable et un grand nombre de valeurs, et peut donc se substituer dans certains cas à plusieurs IF/ELSEIF/ELSE.

La structure SWITCH :

- comprend la variable à évaluer entre parenthèses,
- est suivie d'autant de clauses CASE que de valeurs à tester,
- se termine éventuellement par une instruction DEFAULT.

Les instructions d'un CASE sont exécutées si la variable testée est égale à la valeur qui suit.

Si c'est le cas et que l'on ne souhaite pas évaluer les CASE suivants, il est nécessaire d'ajouter un BREAK après les instructions afin de sortir du SWITCH. Sinon, en l'absence de BREAK, les instructions suivantes sont exécutées sans que les Case ne soient évalués.

Pour finir, si aucune correspondance n'a été trouvée et qu'on a défini un DEFAULT, ce sont les instructions du DEFAULT qui sont exécutées.

Dans l'exemple fourni, la variable *$messages* est définie à *50* (ligne 2), puis incrémentée de *10* (ligne 3).

En ligne 5, on précise que la variable *$messages* doit être évaluée.

Le CASE de la ligne 6 teste l'égalité entre *$messages* et l'entier *50*.

Comme le test est faux, on passe directement au CASE suivant, ligne 9.

L'égalité entre *$messages* et *60* étant vraie, les instructions sont alors exécutées : on affiche "*10 messages reçus.*".

Comme il n'y a pas de break, le CASE suivant est testé avec une comparaison entre l'entier *$messages* valant *60* et la chaîne de caractères "60", ce qui est vrai.

En effet, le SWITCH ne permet de comparer que les valeurs et pas les types, ce qui équivaut à l'opérateur de comparaison == et non ===.

Les instructions suivantes sont alors exécutées avec la décrémentation de $messages avant son affichage, puis le break fait sortir du switch. Le dernier CASE, qui aurait fait l'objet d'une évaluation positive, n'est donc pas testé.

Vous pouvez reproduire et tester cet exemple en changeant la valeur de la variable *$messages*.

Transparent 4

Il existe des structures de contrôle autres que les tests.

Il s'agit des boucles, qui permettent de répéter les mêmes instructions un certain nombre de fois.

Celles utilisées en PHP sont proches du C/C++ ou du JAVA.

La boucle WHILE permet d'exécuter plusieurs fois des instructions, tant qu'une condition est respectée. On pourrait la traduire par TANT QUE ... FAIRE.

Elle comprend :

- une expression entre parenthèses qui renvoie TRUE ou FALSE,
- suivie des instructions à exécuter entre deux accolades.

Il est nécessaire que :

- les variables évaluées dans la condition soient initialisées au préalable,
- et que leur valeur soit modifiée dans la boucle, afin qu'elle ne soit pas infinie.

Il existe une variante du WHILE, la boucle DO/WHILE.

Elle est similaire, mais les instructions sont exécutées une 1ère fois avant que la condition ne soit évaluée.

On pourrait la traduire par FAIRE... TANT QUE

Elle comprend donc :

- d'abord les instructions à exécuter entre deux accolades.
- puis une expression entre parenthèses qui renvoie TRUE (vrai) ou FALSE (faux),

Là encore, il est nécessaire que les variables évaluées dans la condition soient initialisées au préalable et que leur valeur soit modifiée dans la boucle, pour éviter une boucle infinie.

Dans cet exemple, la variable *$messages* est initialisée à *1* (ligne 2).

À l'entrée dans la boucle (ligne 3), la condition est évaluée à TRUE donc les instructions en lignes 4 et 5 sont exécutées : on affiche la variable *$messages*, puis *$messages* est incrémentée.

Arrivée à la fermeture de l'accolade en ligne 6, la boucle reprend en ligne 4.

La condition est à nouveau évaluée et les instructions exécutées, ainsi de suite, tant que *$messages* reste inférieur à 4.

Dans la seconde partie, *$messages* est réinitialisée à 3 (ligne 9).

A l'entrée dans la boucle, les instructions sont exécutées, quelle que soit la valeur de *$messages*.

On affiche donc la variable, puis *$messages* est décrémentée.

Arrivée à la fermeture de l'accolade (ligne 13), la variable *$messages* n'est plus supérieure à *0* au bout de 3 itérations, la condition est donc fausse et la boucle s'arrête.

Ce qu'il est important de comprendre ici, c'est que le test est effectué avant les instructions dans le cas du While, alors qu'il est effectué après dans le cas du DoWhile. Ainsi, avec une condition négative dans le cas d'un While, les instructions peuvent même ne pas être exécutées une seule fois.

Ainsi, les instructions de la dernière boucle ne sont jamais effectuées puisque la condition de test du while n'est pas remplie initialement.

Un DoWhile à la place aurait permis au moins un passage.

Vous pouvez reproduire et tester cet exemple, en changeant la valeur de la variable *$messages*.

Transparent 5

FOR est un autre type de boucle dont la syntaxe est différente :

- les parenthèses contiennent à la fois l'initialisation, la condition et l'incrémentation,
- elles sont suivies des accolades qui contiennent les instructions.

Au 1er passage dans la boucle, la variable est initialisée.

A chaque passage dans la boucle, la condition est évaluée, puis si elle est vraie les instructions sont exécutées, et ensuite l'incrémentation est effectuée.

Les boucles FOREACH sont des variantes de la boucle FOR, et particulièrement adaptées au parcours de tableaux.

Nous aborderons les tableaux ainsi que la syntaxe précise des ForEach ultérieurement, mais l'exemple ci-contre illustre néanmoins leur intérêt.

Cet exemple donne le même résultat que la boucle while vue dans les transparents précédents.

Au 1er passage ligne 2, la variable *$nb_messages* est initialisée à *1*.

Ensuite la condition est évaluée, *$nb_messages* est inférieure à *4*, donc l'instruction (ligne 3) est exécutée.

Après quoi, *$nb_messages* est incrémentée et prend la valeur *2*. La condition est à nouveau évaluée, les instructions exécutées et l'incrémentation effectuée, jusqu'à ce que *$nb_messages* soit égale à *4*.

La condition devient alors fausse et on sort de la boucle.

Dans un deuxième temps, on crée un tableau contenant 3 messages (*message 1*, *message 2* et *message 3*). Pour chaque message du tableau, on affiche le message correspondant.