Concevez votre site web avec PHP et MySQL

30 hours Medium License

Transmettre des données avec l'URL

Savez-vous ce qu'est une URL ? Cela signifie **Uniform Resource Locator**, et cela sert à représenter une adresse sur le web. Toutes les adresses que vous voyez en haut de votre navigateur, comme `http://www.siteduzero.com`, sont des URL.

Je me doute bien que vous ne passez pas votre temps à les regarder (il y a bien mieux à faire !), mais je suis sûr que vous avez déjà été surpris de voir que certaines URL sont assez longues et comportent parfois des caractères un peu curieux. Par exemple, après avoir fait une recherche sur Google, la barre d'adresse contient une URL longue qui ressemble à ceci :

```
http://www.google.fr/search?rlz=1C1GFR343&q=siteduzero
```

Dans cet exemple, les informations après le point d'interrogation sont des données que l'on fait transiter d'une page à une autre. Nous allons découvrir dans ce chapitre comment cela fonctionne.

Envoyer des paramètres dans l'URL

En introduction, je vous disais que l'URL permettait de transmettre des informations. Comment est-ce que ça fonctionne exactement ?

Former une URL pour envoyer des paramètres

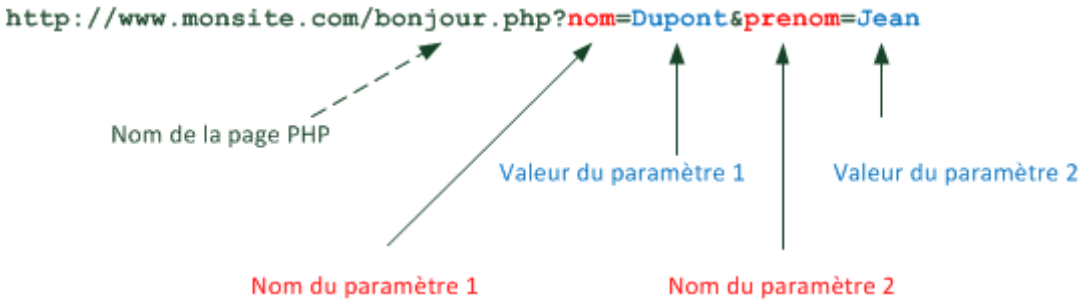
Imaginons que votre site s'appelle `monsite.com` et que vous avez une page PHP intitulée `bonjour.php`. Pour accéder à cette page, vous devez aller à l'URL suivante :

```
http://www.monsite.com/bonjour.php
```

Jusque-là, rien de bien nouveau. Ce que je vous propose d'apprendre à faire, c'est d'**envoyer** des informations à la page `bonjour.php`. Pour cela, on va ajouter des informations à la fin de l'URL, comme ceci :

```
http://www.monsite.com/bonjour.php?nom=Dupont&prenom=Jean
```

Ce que vous voyez après le point d'interrogation, ce sont des **paramètres** que l'on envoie à la page PHP. Celle-ci peut récupérer ces informations dans des variables. Voyez sur la figure suivante comment on peut découper cette URL.



Structure d'une URL

Le point d'interrogation sépare le nom de la page PHP des paramètres. Ensuite, ces derniers s'enchaînent selon la forme `nom=valeur` et sont séparés les uns des autres par le symbole `&`.

On peut écrire autant de paramètres que l'on veut ?

En théorie, oui. Il suffit de les séparer par des & comme je l'ai fait. On peut donc voir une URL de la forme :

`page.php?param1=valeur1&param2=valeur2&param3=valeur3&param4=valeur4…`

La seule limite est la longueur de l'URL. En général il n'est pas conseillé de dépasser les 256 caractères, mais les navigateurs arrivent parfois à gérer des URL plus longues. Quoi qu'il en soit, vous aurez compris qu'on ne peut pas non plus écrire un roman dans l'URL.

Créer un lien avec des paramètres

Maintenant que nous savons cela, nous pouvons créer des liens en HTML qui transmettent des paramètres d'une page vers une autre.

Imaginons que vous avez deux fichiers sur votre site :

- `index.php` (l'accueil) ;
- `bonjour.php` .

Nous voulons faire un lien de `index.php` qui mène à `bonjour.php` et qui lui transmet des informations dans l'URL, comme le schématise la figure suivante.

Lien entre index.php et bonjour.php

Pour cela, ouvrez `index.php` (puisque c'est lui qui contiendra le lien) et insérez-y par exemple le code suivant :

```html
<a href="bonjour.php?nom=Dupont&amp;prenom=Jean">Dis-moi bonjour !</a>
```

Comme vous le voyez, le `&` dans le code a été remplacé par `&` dans le lien. Ça n'a rien à voir avec PHP : simplement, en HTML, on demande à ce que les `&` soient écrits `&` dans le code source. Si vous ne le faites pas, le code ne passera pas la validation W3C.

Ce lien appelle la page `bonjour.php` et lui envoie deux paramètres :

- `nom` : Dupont ;
- `prenom` : Jean.

Vous avez sûrement deviné ce qu'on essaie de faire ici : on appelle une page `bonjour.php` qui va dire « Bonjour » à la personne dont le nom et le prénom ont été envoyés en paramètres.

Comment faire dans la page `bonjour.php` pour récupérer ces informations ? C'est ce que nous allons voir maintenant. ;-)

Récupérer les paramètres en PHP

Nous savons maintenant comment former des liens pour envoyer des paramètres vers une autre page. Nous avons pour cela ajouté des paramètres à la fin de l'URL.

Intéressons-nous maintenant à la page qui réceptionne les paramètres. Dans notre exemple, il s'agit de la page `bonjour.php`. Celle-ci va automatiquement créer un array au nom un peu spécial : `$_GET`. Il s'agit d'un array associatif (vous vous souvenez ? Nous avons étudié cela il y a peu !) dont les clés correspondent aux noms des paramètres envoyés en URL.

Reprenons notre exemple pour mieux voir comment cela fonctionne. Nous avons fait un lien vers `bonjour.php?nom=Dupont&prenom=Jean`, cela signifie que nous aurons accès aux variables suivantes :

Nom	Valeur
`$_GET['nom']`	Dupont
`$_GET['prenom']`	Jean

On peut donc récupérer ces informations, les traiter, les afficher, bref faire ce que l'on veut avec. Pour commencer, essayons tout simplement de les afficher.

Créez un nouveau fichier PHP, appellez-le `bonjour.php` et placez-y le code suivant

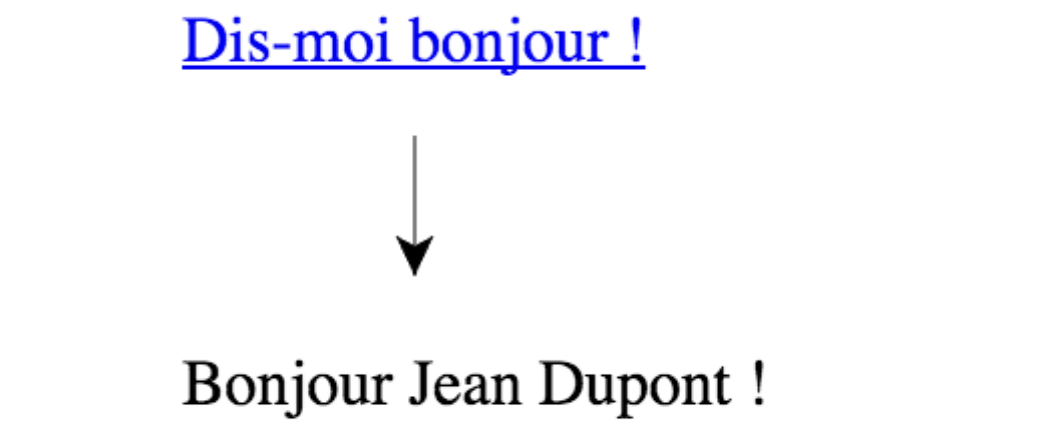
Bien sûr, pour une vraie page web il faudrait écrire toutes les informations d'en-tête nécessaires en HTML : le doctype, la balise `<head>`, etc. Ici, pour faire simple, je ne mets pas tout ce code. La page ne sera pas très belle (ni valide W3C, d'ailleurs) mais ce n'est pas encore notre problème : pour l'instant, nous faisons juste des tests.

php
```
1 <p>Bonjour <?php echo $_GET['prenom']; ?> !</p>
```

Ici, nous affichons le prénom qui a été passé dans l'URL. Bien entendu, nous avons aussi accès au nom. Nous pouvons afficher le nom et le prénom sans problème :

php
```
1 <p>Bonjour <?php echo $_GET['prenom'] . ' ' . $_GET['nom']; ?> !</p>
```

Comme vous le voyez, j'en ai profité pour faire une petite concaténation, comme nous avons appris à le faire. Ce code que vous voyez là n'est pas bien complexe : nous affichons le contenu de deux cases de l'array `$_GET`. C'est vraiment très simple et il n'y a rien de nouveau. Tout ce qu'il faut savoir, c'est qu'on peut retrouver les paramètres envoyés dans l'URL grâce à un array nommé `$_GET`.



Récupérer les paramètres "Jean" et "Dupont" en PHP

Vous devriez aussi faire le test chez vous pour vous assurer que vous comprenez bien le fonctionnement ! Si besoin est, vous pouvez télécharger mes fichiers d'exemple `index.php` et `bonjour.php` :

Télécharger les fichiers d'exemple

Ne faites jamais confiance aux données reçues !

Il est impossible de terminer ce chapitre sans que je vous mette en garde contre les **dangers** qui guettent les apprentis webmasters que vous êtes. Nous allons en effet découvrir qu'il ne faut JAMAIS faire confiance aux variables qui transitent de page en page, comme `$_GET` que nous étudions ici.

Le but de cette section est, je l'avoue, de vous faire peur, car trop peu de développeurs PHP ont conscience des dangers et des failles de sécurité qu'ils peuvent rencontrer, ce qui risque pourtant de mettre en péril leur site web ! Oui je suis alarmiste, oui je veux que vous ayez — un peu — peur ! Mais rassurez-vous : je vais vous expliquer tout ce qu'il faut savoir pour que ça se passe bien et que vous ne risquiez rien. ;-)

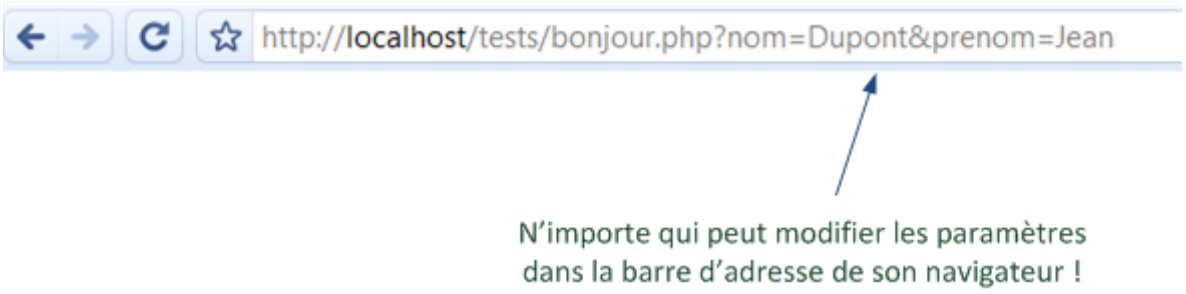
Ce qui compte, c'est que vous ayez conscience très tôt (dès maintenant) des problèmes que vous pourrez rencontrer lorsque vous recevrez des paramètres de l'utilisateur.

Tous les visiteurs peuvent trafiquer les URL

Si vous faites les tests des codes précédents chez vous, vous devriez tomber sur une URL de la forme :

```
http://localhost/tests/bonjour.php?nom=Dupont&prenom=Jean
```

On vous dit bien « Bonjour Jean Dupont ! ». Mais si vous êtes un peu bricoleurs, vous pouvez vous amuser à changer les paramètres directement dans la barre d'adresse, comme dans la figure suivante.



On peut facilement modifier les paramètres dans le navigateur

Essayez par exemple de modifier l'adresse pour :

```
http://localhost/tests/bonjour.php?nom=Dupont&prenom=Marc
```

Comme vous le voyez, ça marche ! N'importe qui peut facilement modifier les URL et y mettre ce qu'il veut : il suffit simplement de changer l'URL dans la barre d'adresse de votre navigateur.

Et alors, quel est le problème ? C'est pas plus mal, si le visiteur veut adapter l'URL pour qu'on lui dise bonjour avec son vrai nom et son vrai prénom ?

Peut-être, mais cela montre une chose : **on ne peut pas avoir confiance dans les données qu'on reçoit**. N'importe quel visiteur peut les changer. Dans la page `index.php` j'ai fait un lien qui dit bonjour à Jean Dupont, mais la page `bonjour.php` ne va pas forcément afficher « Bonjour Jean Dupont ! » puisque n'importe quel visiteur peut s'amuser à modifier l'URL.

Je vous propose de faire des modifications encore plus vicieuses et de voir ce qu'on peut faire pour éviter les problèmes qui en découlent.

Tester la présence d'un paramètre

Allons plus loin. Qu'est-ce qui empêche le visiteur de supprimer tous les paramètres de l'URL ? Par exemple, il peut très bien tenter d'accéder à :

```
http://localhost/tests/bonjour.php
```

Que va afficher la page `bonjour.php` ? Faites le test ! Elle va afficher quelque chose comme :

```
Bonjour
Notice: Undefined index: prenom in C:\wamp\www\tests\bonjour.php on line 9

Notice: Undefined index: nom in C:\wamp\www\tests\bonjour.php on line 9
!
```

Que s'est-il passé ? On a essayé d'afficher la valeur de `$_GET['prenom']` et celle de `$_GET['nom']`… Mais comme on vient de les supprimer de l'URL, ces variables n'ont pas été créées et donc elles n'existent pas ! PHP nous avertit qu'on essaie d'utiliser des variables qui n'existent pas, d'où les « `Undefined index` ».

Pour résoudre ce problème, on peut faire appel à une fonction un peu spéciale : `isset()`. Cette fonction teste si une variable existe. Nous allons nous en servir pour afficher un message spécifique si le nom ou le prénom sont absents.

php
```php
<?php
if (isset($_GET['prenom']) AND isset($_GET['nom'])) // On a le nom et le prénom
{
    echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !';
}
else // Il manque des paramètres, on avertit le visiteur
{
    echo 'Il faut renseigner un nom et un prénom !';
}
?>
```

Que fait ce code ? Il teste si les variables `$_GET['prenom']` et `$_GET['nom']` existent. Si elles existent, on dit bonjour au visiteur. S'il nous manque une des variables (ou les deux), on affiche un message d'erreur : « Il faut renseigner un nom et un prénom ! ».

Essayez maintenant d'accéder à la page `bonjour.php` sans les paramètres, vous allez voir qu'on gère bien le cas où le visiteur aurait retiré les paramètres de l'URL.

Est-ce que c'est vraiment la peine de gérer ce cas ? C'est vrai quoi, moi je ne m'amuse jamais à modifier mes URL et mes visiteurs non plus, je pense !

Oui, oui, trois fois oui : il faut que vous pensiez à gérer le cas où des paramètres que vous attendiez viendraient à manquer.

Vous vous souvenez de ce que je vous ai dit ? Il ne faut jamais faire confiance à l'utilisateur. Tôt ou tard vous tomberez sur un utilisateur malintentionné qui essaiera de trafiquer l'URL pour mettre n'importe quoi dans les paramètres. Il faut que votre site soit prêt à gérer le cas.

Dans notre exemple, si on ne gérait pas le cas, ça ne faisait rien de bien grave (ça affichait juste des messages d'erreur). Mais lorsque votre site web deviendra plus complexe, cela pourrait avoir des conséquences plus ennuyeuses.

Un exemple ? Sur le Site du Zéro lui-même, nous avions une fois oublié de gérer le cas où un paramètre viendrait à manquer. Un utilisateur avait essayé de l'enlever « pour voir » : notre page PHP était faite de telle sorte que si le paramètre manquait, ça supprimait toutes les préférences de tous les membres inscrits du site ! Vous n'en êtes pas encore là, mais je tiens à vous sensibiliser aux problèmes potentiels que cela peut provoquer. Soyez donc vigilants dès maintenant et ne supposez jamais que vous allez recevoir tous les paramètres que vous attendiez.

Contrôler la valeur des paramètres

Allons plus loin dans notre tripatouillage vicieux de l'URL. On va modifier notre code source pour gérer un nouveau paramètre : le nombre de fois que le message doit être répété. Les paramètres vont donc être :

`bonjour.php?nom=Dupont&prenom=Jean&repeter=8`

Le paramètre `repeter` définit le nombre de fois que l'on dit « bonjour » sur la page. Sauriez-vous adapter notre code pour qu'il dise bonjour le nombre de fois indiqué ? Voici la solution que je vous propose :

```php
<?php
if (isset($_GET['prenom']) AND isset($_GET['nom']) AND isset($_GET['repeter']))
{
    for ($i = 0 ; $i < $_GET['repeter'] ; $i++)
    {
        echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !<br />';
    }
}
else
{
   echo 'Il faut renseigner un nom, un prénom et un nombre de répétitions !';
}
?>
```

On teste si le paramètre `repeter` existe lui aussi (avec `isset($_GET['repeter'])`). Si tous les paramètres sont bien là, on fait une boucle (j'ai choisi de faire un `for`, mais on aurait aussi pu faire un `while`). La boucle incrémente une petite variable `$i` pour répéter le message de bienvenue le nombre de fois indiqué.

Le résultat ? Si on va sur…

```
http://localhost/tests/bonjour.php?nom=Dupont&prenom=Jean&repeter=8
```

… alors le message de bienvenue s'affichera huit fois :

```
Bonjour Jean Dupont !
Bonjour Jean Dupont !
Bonjour Jean Dupont !
Bonjour Jean Dupont !
Bonjour Jean Dupont !
Bonjour Jean Dupont !
Bonjour Jean Dupont !
Bonjour Jean Dupont !
```

Nous avons amélioré notre page pour qu'elle puisse dire « bonjour » plusieurs fois, et ce nombre de fois est personnalisable dans l'URL. Très bien.

Maintenant, mettez-vous dans la peau du **méchant**. Soyez méchants. Soyez vicieux. Que pourrions-nous faire et que nous n'aurions pas prévu, juste en modifiant l'URL ?

J'ai une idée ! On peut mettre un nombre de répétitions très grand, par exemple 1984986545665156. La page PHP va boucler un très grand nombre de fois et votre PC ne pourra probablement pas supporter une telle charge de travail.

```
bonjour.php?nom=Dupont&prenom=Jean&repeter=1984986545665156
```

Si vous avez peur d'essayer, je vous rassure : votre PC ne va pas prendre feu en faisant ça. Par contre, il va se mettre à consommer énormément de mémoire pour stocker tous les messages « bonjour » et n'arrivera sûrement pas jusqu'au bout (PHP s'arrête au bout d'un certain temps d'exécution). Ou alors s'il y arrive, votre page va être extrêmement surchargée de messages « bonjour ».

Mon dieu, c'est horrible ! Comment peut-on empêcher ça ?

En étant plus malins et surtout plus prévoyants. Voici ce qu'il faut anticiper.

- Le cas où le nombre qu'on vous envoie **n'est pas une valeur raisonnable**.
  - Par exemple on peut dire que si on dépasse 100 fois, c'est trop, et il faut refuser d'exécuter la page.
  - De même, que se passe-t-il si je demande de répéter « -4 fois » ? Ça ne devrait pas être autorisé. Il faut que le nombre soit au moins égal à 1.
- Le cas où **la valeur n'est pas logique**, où elle n'est pas du tout ce qu'on attendait. Rien n'empêche le visiteur de remplacer la valeur du paramètre `repeter` par du texte !
  - Que se passe-t-il si on essaie d'accéder à

    `bonjour.php?nom=Dupont&prenom=Jean&repeter=grenouille` ?

    Cela ne devrait pas être autorisé. Le mot « grenouille » n'a pas de sens, on veut un nombre entier positif.
  - Que se passe-t-il si on essaie d'accéder à

    `bonjour.php?nom=Dupont&prenom=Jean&repeter=true` ?

    Cela ne devrait pas être autorisé non plus, on attendait un nombre entier positif, pas un booléen.

Pour résoudre ces problèmes, on doit :

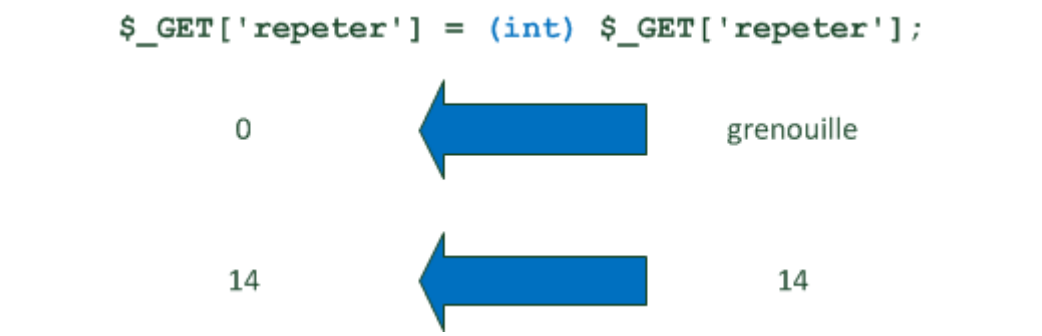
1. vérifier que `repeter` contient bien un nombre ;
2. vérifier ensuite que ce nombre est compris dans un intervalle raisonnable (entre 1 et 100, par exemple).

Pour vérifier que `repeter` contient bien un nombre, une bonne solution pourrait être de forcer la conversion de la variable en type entier (`int`). On peut utiliser pour cela ce qu'on appelle le **transtypage**, une notion qu'on n'a pas encore vue jusqu'ici mais qui est très simple à comprendre. Regardez ce code :

php

```
<?php
$_GET['repeter'] = (int) $_GET['repeter'];
?>
```

Il faut lire cette instruction de droite à gauche. Le `(int)` entre parenthèses est comme un tuyau de conversion. Tout ce qui transite à travers ressort forcément en une valeur de type `int` (entier). Voyez la figure suivante pour vous en convaincre.



Transtypage

Si la valeur est bien un entier (par exemple 14) alors elle n'est pas modifiée. En revanche, si la variable contient autre chose qu'un entier (par exemple « grenouille »), elle est transformée en entier. Ici, comme PHP ne peut pas associer de valeur à *grenouille*, il lui donne 0.

Après avoir exécuté cette instruction, la variable `$_GET['repeter']` contient maintenant forcément un nombre entier. Il ne reste plus qu'à vérifier que ce nombre est bien compris entre 1 et 100 à l'aide d'une condition (ce que vous savez faire, normalement !).

Voici donc le code final sécurisé qui prévoit tous les cas pour éviter d'être pris au dépourvu par un utilisateur malintentionné :

php

```php
<?php
if (isset($_GET['prenom']) AND isset($_GET['nom']) AND isset($_GET['repeter']))
{
    // 1 : On force la conversion en nombre entier
    $_GET['repeter'] = (int) $_GET['repeter'];

    // 2 : Le nombre doit être compris entre 1 et 100
    if ($_GET['repeter'] >= 1 AND $_GET['repeter'] <= 100)
    {
        for ($i = 0 ; $i < $_GET['repeter'] ; $i++)
        {
            echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !<br />';
        }
    }
}
else
{
   echo 'Il faut renseigner un nom, un prénom et un nombre de répétitions !';
}
?>
```

Cela fait beaucoup de conditions pour quelque chose qui était à la base assez simple, mais c'était nécessaire. Vous devrez toujours faire très attention et prévoir tous les cas les plus tordus, comme nous venons de le faire :

- vérifier que tous les paramètres que vous attendiez sont bien là ;
- vérifier qu'ils contiennent bien des valeurs correctes comprises dans des intervalles raisonnables.

Si vous gardez cela en tête, vous n'aurez pas de problèmes. :-)

Nous n'avons pas encore vu tous les risques liés aux données envoyées par l'utilisateur. Cette première approche devrait déjà vous avoir sensibilisés au problème, mais nous irons plus loin dans le chapitre suivant pour finir de couvrir ce sujet. Tout ceci est un peu complexe, je suis d'accord, mais c'est réellement important !

En résumé

- Une URL représente l'adresse d'une page web (commençant généralement par `http://`).
- Lorsqu'on fait un lien vers une page, il est possible d'ajouter des paramètres sous la forme `bonjour.php?nom=Dupont&prenom=Jean` qui seront transmis à la page.
- La page `bonjour.php` dans l'exemple précédent recevra ces paramètres dans un array nommé `$_GET` :
    - `$_GET['nom']` aura pour valeur « Dupont » ;
    - `$_GET['prenom']` aura pour valeur « Jean ».

- Cette technique est très pratique pour transmettre des valeurs à une page, mais il faut prendre garde au fait que le visiteur peut les modifier très facilement. Il ne faut donc pas faire aveuglément confiance à ces informations, et tester prudemment leur valeur avant de les utiliser.
- La fonction `isset()` permet de vérifier si une variable est définie ou non.
- Le transtypage est une technique qui permet de convertir une variable dans le type de données souhaité. Cela permet de s'assurer par exemple qu'une variable est bien un `int` (nombre entier).

Que pensez-vous de ce cours ?

Quiz: Quiz 2

Transmettre des données avec les formulaires

The author

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Check out this course via

eBook


Livre papier

PDF

Vidéo