Chapitre 12

# Les sites dynamiques

En ayant appris à développer un site HTML qui utilise JavaScript et des feuilles CSS, vous connaissez toutes les techniques de développement web utilisées jusqu'en l'an 2000. Depuis, le développement web ne s'est pas arrêté, il s'est simplement déplacé. Jusque-là, tout était fait du côté de l'ordinateur du client. Le schéma était alors le suivant :

1. Un client se connecte à un site web via une adresse du type **www.mapageweb.com**.
2. L'ordinateur qui héberge le site envoie à l'ordinateur client l'objet de sa requête, c'est-à-dire la page demandée.
3. Si la page contient du contenu JavaScript, le script est exécuté sur l'ordinateur client.

Cela convenait à tout le monde, ou presque.

L'un des principaux désavantages de cette méthode est que toute la page doit être envoyée au client, et ce à chaque requête. Cela implique que toute la page est disponible du côté du client. Que se passe-t-il si la page contient un mot de passe ? Il est en clair sur la page et tous ceux qui peuvent la lire le voient. D'un point de vue concurrentiel, cela signifie également que l'on peut sauvegarder toutes les pages sur un disque dur, modifier le site puis le remettre en ligne en s'appropriant le travail.

En bref, ce schéma de développement présentait quelques désavantages.

# 12.1. Le schéma client-serveur

Pour remédier à ces problèmes, les langages de développement côté serveur sont apparus. La principale différence est que la page web n'est plus stockée, telle qu'elle est affichée, sur l'ordinateur distant qui fait la requête au serveur. On utilise un langage qui, une fois interprété par le serveur, donne la page finale au client. Les étapes sont les suivantes :

1. Un ordinateur se connecte à un site web.
2. Le serveur qui héberge le site interprète la page demandée.
3. La page voulue est envoyée au client dans une forme compréhensible par son navigateur web.

Le principal avantage de cette méthode est la possibilité d'un véritable échange entre le client et le serveur.

De cette manière et en utilisant un langage complexe, vous pourrez proposer à un visiteur de véritables programmes sur le Web, à travers vos sites. Tous les sites modernes utilisent des technologies serveur.

Deux principales technologies se partagent le marché : d'un côté, les solutions dites "open source", avec les outils PHP, Apache et MySQL, et de l'autre, les solutions propriétaires représentées par Microsoft, avec les technologies ASP et SQL Server.

# 12.2. PHP

PHP est l'acronyme de "PHP Hypertext Preprocesseur", ce qui signifie en français "préprocesseur PHP". Il s'agit de générer une page web au format HTML pour la rendre lisible sur un navigateur.

## Installer une solution PHP

Pour pouvoir développer des pages web en utilisant PHP, il faut installer un serveur sur votre machine. Il existe des packs préconfigurés. Leur utilisation est à la portée de tous. En outre, les utilisateurs forment une grande communauté et sont prêts à vous apporter leur aide pour peu que vous les sollicitiez gentiment.

La solution PHP tout en un la plus connue est certainement EasyPHP. En l'installant, vous disposez de tous les outils : un serveur web directement sur votre machine pour pouvoir tester vos pages, un moteur de base de données que vous pourrez utiliser avec vos pages et les outils d'administration graphiques.

Pour télécharger EasyPHP, rendez-vous à l'adresse **www.easyphp.org**. Téléchargez le programme d'installation, lancez-le et suivez les instructions.

Maintenant que votre machine est configurée pour exécuter des pages en PHP, lancez votre éditeur de texte et recopiez la page suivante :

```
<?php
Echo ("Ma première page Php");
?>
```

Enregistrez le fichier dans le dossier *www* du répertoire d'installation d'EasyPHP, en lui donnant l'extension *.php*.

Lancez maintenant votre navigateur Internet et rendre-vous à l'adresse **http://localhost/nomdevotrepage/php**. **localhost** est simplement un alias stipulant que la page est stockée sur votre serveur.


**Les instructions**

En PHP, les instructions se terminent par un point-virgule (`;`) en fin de ligne. Ne l'omettez pas, sous peine de rendre vos pages inutilisables.

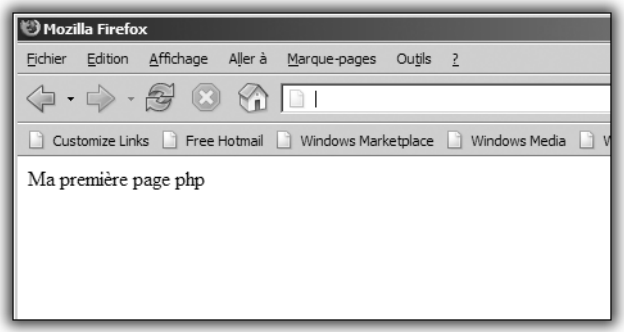


***Figure 12.1 :*** *Votre première page*

Remarquez que le code PHP est placé à l'intérieur des balises `<?php` et `?>`.

L'utilité des technologies serveur est la réception et l'envoi de données. Vous allez voir comment procéder en recourant aux formulaires web.

## Utiliser les formulaires web

Il est possible de déclarer des formulaires dans une page web en utilisant les balises `<form>` et `</form>`. Entre ces deux balises, vous pouvez placer des balises `input`, qui constitueront les éléments de votre formulaire.

**1** Recopiez la page suivante dans un éditeur de texte :

```
<html>
<head>
    <title>Ma page de formulaire</title>
    <link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<form>
Nom:<input type=text name="nom"/>
Prénom:<input type=text name="prenom"/>
Âge:<input type=text name="age"/>
<input type=submit />
</form>
</body>
</html>
```

**2** Sauvegardez la page au format HTML puis affichez-la dans votre navigateur.

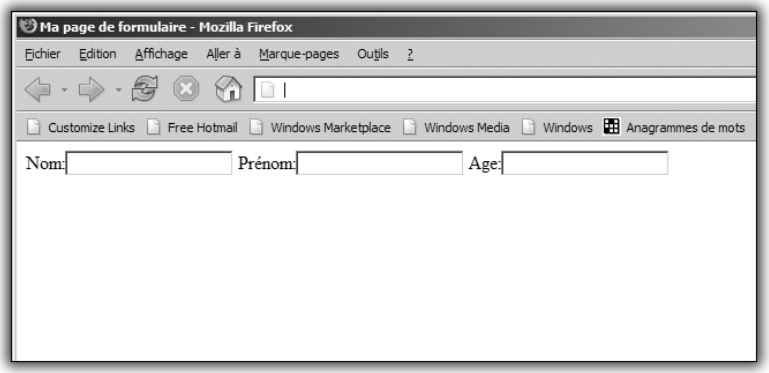


***Figure 12.2 :*** *Le formulaire*

Vous disposez maintenant d'une page qui affiche trois zones de saisie de texte : *Nom*, *Prénom* et *Âge*.

Le problème est que, pour l'instant, vous ne pouvez rien en faire. Vous allez changer cela tout de suite.

**3** Ouvrez un nouveau document texte et copiez le contenu suivant :

```
<?Php
$nom = $_POST["nom"];
```

```
$prenom= $_POST["prenom"];
$age=$_POST["age"]
echo "Vous vous appelez".$prenom." ".$nom." et vous
✂ avez ".$age." ans";
?>
```

Il s'agit de la page qui sera destinée au traitement des données du formulaire. Pour récupérer ces données, il convient de les stocker en utilisant des variables. Il faut pour cela les déclarer de cette façon : `$nomdelavariable=valeur`.

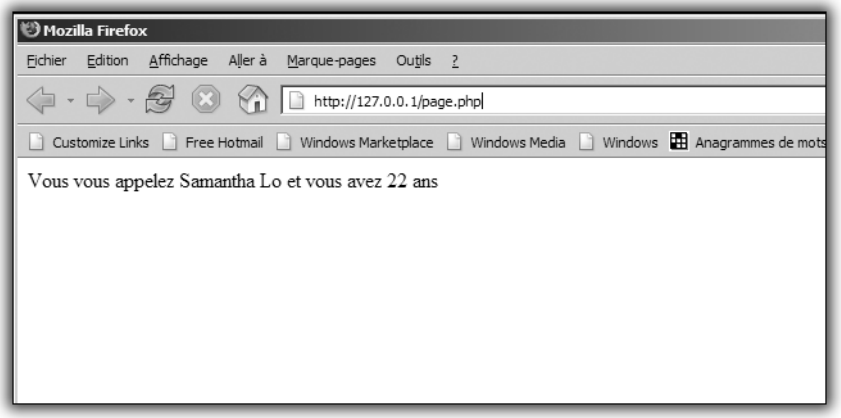


***Figure 12.3 :*** *Le questionnaire traité*

Tout comme JavaScript et Visual Basic.Net, PHP est un langage faiblement typé. Il n'est donc pas nécessaire de préciser le type des variables lorsque vous les déclarez. La valeur que vous leur affectez ici est celle qui est récupérée depuis le formulaire. En PHP, lorsqu'un formulaire est utilisé, ses variables sont stockées différemment selon la méthode utilisée pour déclarer le formulaire. Il existe deux méthodes, `POST` et `GET`, sur lesquelles nous reviendrons. Vous utilisez ici la méthode `POST`.

Quelle que soit la méthode employée, la manière de stocker les variables du formulaire ne change pas. Elles sont enregistrées dans un tableau, chacune dans une case qui porte le nom donné à la balise `input` du formulaire. Le nom du tableau est toujours du type `$_POST[""]` ou `$_GET[]` selon la méthode utilisée.

Vous appelez ensuite la fonction `echo` de PHP, qui permet d'afficher du texte sur une page. Il suffit de lui passer en paramètre les chaînes et les variables dont vous souhaitez afficher le contenu.

**4** Il faut maintenant faire le lien entre le formulaire et son traitement. Modifiez la page d'origine conformément au modèle suivant :

```
<html>
<head>
    <title>Ma page de formulaire</title>
    <link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<form method="POST" action="traitement.php">
Nom:<input type=text name="nom"/>
Prénom:<input type=text name="prenom"/>
Age:<input type=text name="age"/>
<input type=submit />
</form>
</body>
</html>
```

Il subsiste une ligne non encore décrite. Il s'agit de :

```
<input type=submit />
```

Elle a pour but d'ajouter à la page de formulaire un bouton de validation qui valide les données et les envoie à la page de traitement spécifiée dans l'attribut `action` de la balise ouvrante du formulaire. Copiez ces deux pages dans le dossier *www* du répertoire d'installation d'EasyPHP.

## Les deux méthodes

L'attribut `method` de la balise ouvrante du formulaire permet de spécifier si l'on souhaite utiliser la méthode `POST` ou `GET` pour transmettre les données. `GET` transmet les données dans l'adresse de la page. C'est le cas par exemple des données transmises à un moteur de recherche comme Google. Si vous allez sur le site et que vous faites une recherche, l'adresse de la page de résultat contiendra cette recherche.

En d'autres termes, une fois que vous cliquerez sur le bouton de validation du formulaire, la page de traitement sera chargée ; son adresse

contiendra le nom du paramètre et sa valeur, sous la forme **www.mapagedetraitement.com?nomduparamètre=valeur**. Vous pouvez prendre l'exemple suivant pour effectuer une recherche sur Google à partir de votre site :

```
<html>
<head runat="server">
    <title>Ma page de formulaire</title>
    <link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<form method=GET action=http://www.google.fr>
Recherche:<input type=text name=q/>
<input type=submit />
</form>
</body>
</html>
```

Dans la partie `body` de la page, vous déclarez un formulaire. La méthode est `GET` et le traitement est reporté à la page **www.google.fr**. Dans ce formulaire, vous ajoutez un champ de saisie de texte appelé `q`, puis un bouton de validation.

Chargez la page dans un navigateur, entrez un ou plusieurs mots dans le champ de texte et appuyez sur le bouton de validation. Votre navigateur va charger la page de traitement du formulaire, en ajoutant les données saisies dans le champ de recherche. L'adresse chargée devient alors **www.google.fr?q=mot+recherché**.


**Récupération des données**

Bien que les données de formulaire soient passées dans la barre d'adresse, il n'est pas possible de les récupérer pour les traiter autrement qu'en lisant le tableau de données `GET[]` associé au formulaire.

La méthode `POST` transmet les données dans l'en-tête de la page, c'est-à-dire de manière complètement transparente pour l'utilisateur. L'avantage principal est de ne pas exposer les données sensibles à n'importe qui. Si par exemple vous demandez une identification par mot de passe sur votre site en faisant appel à la méthode `GET`, le mot de passe apparaît en clair dans la barre d'adresse du navigateur, ce qui est à proscrire dans le cadre d'une application sécurisée.

# Les instructions de contrôle

PHP propose des méthodes de contrôle, au même titre que Visual Basic .NET. Toutes les boucles et structures classiques, comme `If… Then… Else` ou `Switch`, sont utilisables. Leur syntaxe en PHP, comparativement à Visual Basic .NET, est légèrement différente. Voici un récapitulatif de ces structures de contrôle en PHP.

## Si… Alors… Sinon

La boucle de type `If` se décrit de cette manière :

```
<html>
<head runat="server">
<title>Ma page de If</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<?php
$variable = 150
if ($variable<250){
 echo("La variable est plus petite que 250");
}
else{
 echo("La variable est plus grande que 250");
}
?>
</body>
</html>
```

Notez bien l'utilisation d'accolades pour délimiter les blocs d'instructions. Si vous avez plusieurs tests successifs, vous pouvez utilisez la structure `If… ElseIf… Else`, qui permet d'imbriquer plusieurs tests, de la manière suivante :

```
<html>
<head runat="server">
<title>Ma page de If</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<?php
$variable = 150
if ($variable<250){
 echo("La variable est plus petite que 250")
}
elseif($variable>250){
 echo("La variable est plus grande que 250")
}
```

```
else{
  echo("La valeur de la variable n'a pas été testée")
}
?>
</body>
</html>
```

Il existe une autre façon d'écrire le contrôle `If`, sans les accolades :

```
<html>
<head runat="server">
<title>Ma page de If</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<?php
if(votre test):
  Instructions;
else:
  Instructions;
?>
</body>
</html>
```

Attention car cette écriture n'est pas compatible avec toutes les versions de PHP.

## Les boucles

Les boucles itératives permettent de répéter une ou plusieurs instructions.

### *While*

La boucle `While`, également proposée par Visual Basic .NET, a la syntaxe suivante en PHP :

```
While(votrecondition){
//Vos instructions
}
```


**Commentaires**

Les doubles barres obliques correspondent à la mise en commentaire d'une ligne.

La boucle `While` permet de répéter les instructions tant que le test conditionnel est vrai. À cause de cela, si vous ne pensez pas à mettre un cas d'arrêt, c'est-à-dire un modificateur de la condition de test, la boucle ne s'arrêtera jamais. Heureusement, la plupart des navigateurs récents arrêtent le programme automatiquement au bout de 30 secondes.

### For

La boucle `For` permet, contrairement à la boucle `While`, de préciser le nombre d'itérations à partir d'une condition initiale et l'arrêt de la boucle avec une condition d'arrêt.

Elle est utile lorsque vous raisonnez en termes de nombre d'itérations plutôt qu'en termes de condition.

La syntaxe de la boucle `For` en PHP est la suivante :

```
$i;
$j=2;
For($i=0;$i<10;$i++){
   Echo $i*$j;
}
```

Vous affichez ici la table de multiplication par 2. Sur les deux premières lignes, vous initialisez deux variables `i` et `j`. La première sert de compteur, la seconde de variable pour la table de multiplication.

Dans la boucle `for`, vous précisez l'initialisation, ici la valeur de base de `i`, ensuite le cas d'arrêt, à savoir une sortie de la boucle d'exécution quand `i` est égal à 9 (il faut donc tester si la valeur de `i` est strictement inférieure à 10), et enfin le pas, ce qui représente la modification appliquée à la valeur de `i` à chaque itération de la boucle.

### Switch

La structure `Switch` permet de remplacer avantageusement un grand nombre de tests conditionnels :

```
<html>
<head runat="server">
<title>Ma page de If</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<?php
if ($i == 0) {
    print "i égale 0";
```

```
}
if ($i == 1) {
    print "i égale 1";
}
if ($i == 2) {
    print "i égale 2";
}
switch ($i) {
    case 0:
        print "i égale 0";
        break;
    case 1:
        print "i égale 1";
        break;
    case 2:
        print "i égale 2";
        break;
}
?>
</body>
</html>
```

Ces deux blocs de code sont équivalents. Le `Switch` permet de clarifier la syntaxe qui peut être brouillonne si vous multipliez les tests.


**Instruction Break**

L'instruction `break` des éléments `case` de la structure précédente permet d'arrêter le traitement des instructions. Si vous l'omettez, toutes les instructions de tous les `case` seront exécutées.

## Les fonctions

Il est également possible de déclarer des fonctions en PHP, avec tous les avantages que cela amène. Pour écrire des fonctions en PHP, il suffit d'utiliser le mot-clé `Function` de la manière suivante :

```
<html>
<head runat="server">
<title>Ma page de fonctions</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<?php
Function nomdelafonction($argument1,$argument2){
Echo $argument1+$argument2;
?>
```

```
</body>
</html>
```

Ici, la fonction va effectuer un affichage sur la page de deux arguments qu'elle reçoit en paramètre.


**Tests**

Ce genre de fonctions est utile pour effectuer des tests sur un projet afin de trouver d'où viennent les problèmes.

Vous pouvez maintenant structurer correctement les fichiers de votre site en organisant au mieux votre code. Pour bien réutiliser du code, il convient de le placer dans un fichier à part. Imaginez une fonction qui affiche le formulaire de recherche sur Google sur chacune des pages de votre site :

```
<html>
<head runat="server">
<title>Ma page de fonctions</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>

<?
Function QuestionGoogle(){
<form method=GET action=http://www.google.fr>
Recherche:<input type=text name=q/>
<input type=submit />
</form>
}
?>
</body>
</html>
```

Vous allez la placer dans un fichier à part : *fonctions1.php*. Pour réutiliser le contenu de *fonctions1.php* dans un autre fichier, il suffit d'ajouter la ligne suivante :

```
<?include "fonctions1.php"?>
```

En ajoutant cette ligne en haut du fichier courant, vous pourrez utiliser tout le code placé à l'intérieur de *fonctions1.php* comme si vous l'aviez copié-collé.

# 12.3. PHP côté serveur

Vous savez à présent développer des scripts PHP qui permettent un retour d'informations au client, c'est-à-dire à l'ordinateur qui se connecte au site web.

Il y a deux manières d'interagir avec le client :

- soit en utilisant des fichiers cookies, qui enregistrent des informations sur l'ordinateur du client ;
- soit en utilisant des sessions, qui stockent temporairement des informations dans la mémoire de l'ordinateur qui héberge le site web.

Le cookie est un fichier texte qui peut contenir la valeur que vous voulez. Vous pouvez faire en sorte que le fichier cookie expire lorsque l'utilisateur quitte votre site. Mais il est courant de l'utiliser pour enregistrer par exemple la date de la dernière visite de l'internaute. Si vous ne prévoyez pas un délai assez large entre deux visites, vous perdrez l'information.

## Enregistrer un cookie

Pour enregistrer un cookie sur l'ordinateur du client, utilisez le code suivant :

```
<html>
<head runat="server">
<title>Ma page de fonctions</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>

<?php
$valeur = "Valeur de test";

setcookie("MonCookie", $valeur);
?>
</body>
</html>
```

La fonction `setcookie()` définit le fichier à écrire sur l'ordinateur client ou plutôt le contenu du fichier.


**Plusieurs valeurs pour un cookie**

Il n'est permis d'écrire qu'une seule valeur par fichier cookie. Aussi, si vous comptez stocker plusieurs informations, vous devez utiliser un séparateur de texte, par exemple la barre oblique ou un point-virgule.

Selon la manière dont l'utilisateur a configuré son navigateur Internet, le fichier est maintenant stocké quelque part sur le disque dur de l'ordinateur client. Pour en lire le contenu, il faudra utiliser un code ressemblant à celui-ci :

```
<html>
<head runat="server">
<title>Ma page de fonctions</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>

<?php
echo $_COOKIE["MonCookie"];
?>
</body>
</html>
```

La lecture des cookies se fait par un tableau les contenant tous. En inscrivant le nom de votre cookie (le premier paramètre de la fonction `setcookie`) en index du tableau, vous pourrez en récupérer la valeur et l'utiliser directement. Dans l'exemple précédent, il est question de l'afficher, mais vous pouvez le stocker dans une variable, comme suit :

```
<html>
<head runat="server">
<title>Ma page de fonctions</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>

<?php
$Variable = $_COOKIE["MonCookie"];
?>
</body>
</html>
```

Les cookies sont des systèmes de stockage d'informations utilisés plutôt sur le long terme. Pour le court terme, il est d'usage de recourir au mécanisme de sessions.

# Organiser des sessions

Les sessions sont un peu l'équivalent des variables globales de Visual Basic .NET. En effet, lorsque vous déclarez une variable en PHP, elle n'est visible qu'à l'intérieur de la page dans laquelle se trouve la définition. Si un utilisateur change de page, la variable est détruite. Pour éviter cela, il est possible de définir une session, dotée d'un numéro qui sera unique, vous permettant d'une part d'identifier les utilisateurs, d'autre part de garantir la sécurité des données leur appartenant.

Le mécanisme de sessions doit être démarré au début de chaque page à l'aide de la commande :

```
<?
Session_start();
?>
```

Cette ligne induit deux comportements différents :

- Soit l'utilisateur vient d'une page où les sessions existaient déjà, il dispose donc d'un numéro unique de session qui est réactivé sur cette page.
- L'utilisateur vient d'une page où les sessions n'étaient pas initialisées et un numéro unique est initialisé dans les informations de navigation de l'internaute sur le serveur.

## Activer une session

Maintenant que la session est activée, il suffit de s'en servir pour enregistrer des variables de session de la manière suivante :

```
<?php
Session_start();
$Variable = "valeur";
Session_register("Variable");
?>

<html>
<head runat="server">
<title>Ma page de sessions</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>

</body>
</html>
```

Dans cet exemple de page, vous déclarez une variable contenant la chaîne "valeur", qui sera ensuite intégrée à la collection des variables de session. Le paramètre de Session_register() est le nom d'une variable et n'est pas précédé du caractère $.

Attention : les initialisations de session doivent figurer en tout début de fichier. En effet, vous risquez d'avoir des erreurs dans l'exécution du script de votre page si le démarrage des sessions se fait au milieu de la page.

## Utiliser les variables de session

Maintenant qu'une variable de session est enregistrée, il faut l'utiliser. Il suffit pour cela d'avoir recours au tableau de valeurs $_SESSION[nom de la valeur]. Le code suivant permet d'afficher sur une deuxième page le contenu de la variable enregistrée précédemment :

```
<?php
Session_start();
?>

<html>
<head runat="server">
<title>Ma page de sessions</title>
<link rel=stylesheet href=StyleSheet.css />
</head>
<body>
<?
Echo $_SESSION[Variable]
?>
</body>
</html>
```

Lorsque vous chargez cette page, le contenu de la variable enregistrée précédemment s'affiche.