2 Structure et éléments du langage

PLAN

OBJECTIFS

- Connaître la structure d'un programme PHP et quelques éléments du langage.
- Savoir intégrer du code PHP dans une page HTML.

2.1 Introduction

Un fichier source « .php » contient des lignes de codes PHP et HTML. Afin de maîtriser ce mélange de syntaxes, et d'éviter les incohérences structurelles, nous présentons la structure d'un programme PHP et certains éléments du langage.

2.2 Structure d'un programme

Les balises <?php et ?>

Un programme PHP commence par **<?php** et se termine par **?>**, comme le montre le programme `hello.php`.

Listing 2.1 – Programme hello.php

```
<?php
 echo "Hello";
?>
```

Si le programme est conçu pour être exécuté en mode shell, il ne contient qu'une seule occurrence de ces deux balises, en début et en fin de fichier.

Pour une exécution *via* un navigateur, ce qui est le cas le plus courant, le programme PHP mélange les syntaxes HTML et PHP. Ces deux balises sont utilisées à chaque fois que le développeur insère du code PHP dans une page HTML.

L'intégration dans un code HTML

Les balises <html> et </html>

Un programme PHP a vocation à être intégré dans des pages HTML. Il est généralement entouré de balises comme `<html>` et `</html>`.

Le programme `hello2.php` affiche deux textes suivis de sauts de ligne HTML `<br/>`.

Listing 2.2 – Programme hello2.php

```
<html>
 <?php
  echo "Hello <br/>";
  echo "World <br/>";
 ?>
</html>
```

La figure 2.1 présente l'exécution de ce programme *via* le navigateur.

Figure 2.1 – Exécution de hello2.php via le navigateur.

L'exécution de ce programme en mode shell fait simplement apparaître les codes HTML qui n'ont de sens que pour le navigateur.

Listing 2.3 – Exécution de hello2.php en mode shell

```
$ php hello2.php
<html>
 Hello <br/>World <br/></html>
```

Le mélange de code HTML et de code PHP

D'une manière générale, le code PHP peut être inséré n'importe où dans la page HTML, dès lors qu'il est délimité par `<?php` et `?>`. Il est important de structurer le fichier source « .php » afin de dissocier au mieux la partie PHP de la partie HTML, même si cela n'est pas toujours évident.

Pour les développements plus complexes, il est préconisé d'organiser son site web en se basant sur l'architecture *MVC* (pour *Modèle-Vue-Contrôleur*), qui modélise la conception des programmes, en séparant :

- la gestion des données, contenant des lignes de programmation PHP accédant aux bases de données = *le Modèle* ;
- l'affichage contenant principalement des syntaxes HTML = *la Vue* ;
- l'organisation entre les pages implémentant la logique d'enchaînement des pages, contenant des programmes PHP = *le Contrôleur*.

Même si cette organisation peut s'avérer lourde dans le cadre de petits sites web, il est important de concevoir les programmes pour permettre leur évolution ultérieure vers ce type d'architecture.

L'extension du fichier .html ou .php

Si le fichier ne contient que des syntaxes HTML, l'extension doit être « .html ». Mais elle peut être également « .php », même si cela n'a aucune incidence. Si le fichier contient des syntaxes PHP (en plus d'éventuelles syntaxes HTML), son extension doit obligatoirement être « .php ». Dans le cas contraire, le code PHP ne serait pas interprété. Voici l'exemple du programme `mapage2.php` :

Listing 2.4 – Programme mapage2.php

```
<!DOCTYPE html>
<html>
    <head> <!-- Entête HTML -->
        <meta charset="utf-8" />
        <title>Ma Page HTML</title>
    </head>
    <body> <!-- Corps de la page HTML -->
     <h1> Ma premi&egrave;re page HTML</h1>
     <p>
           Bienvenue sur ma page<br/>
           Madame, Monsieur <?php echo 'Dupont';?>
     </p>
    </body>
</html>
```

La figure 2.2 présente son exécution *via* le navigateur.

Figure 2.2 –Exécution de mapage2.php via le navigateur.

L'instruction include

L'instruction `include` inclut le texte d'un fichier à un endroit précis du programme. Cette instruction évite la réécriture de syntaxes présentes dans plusieurs pages, par exemple l'entête « header » ou le pied de page « footer ». Prenons l'exemple de `site_sans_include.php` de la figure 2.3. Les zones encadrées représentent les lignes présentes sur toutes les pages du site.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Site Assistance</title>
  </head>

  <body>
   <header>
    <!-- Les autres sites -->
    <nav id="autres_sites">
      <div class="element_autres_sites">
      <h3>Sites de recherche</h3>
       <ul>
        <li><a href="http://www.yahoo.fr">Rechercher sur Yahoo</a></li>
        <li><a href="http://www.google.fr">Rechercher sur Google</a></li>
       </ul>
      </div>
    </nav>
   </header>
    <!-- Le contenu -->
    <div id="contenu">
      <h1>Assistance</h1>
      <p>
         Vous &ecirc;tes sur le site d'assistance !<br>
         .........<br>
      </p>
    </div>

    <!-- Le pied de page -->
    <footer id="pied_de_page">
     <p>Contact : assistance@assist.fr</p>
    </footer>

  </body>
</html>
```

Figure 2.3 –Fichier site_sans_include.php.

La figure 2.4 présente son exécution *via* le navigateur. Les zones encadrées correspondent à l'en-tête et au pied de page.

Figure 2.4 –Exécution de site_sans_include.php via le navigateur.

Pour faciliter la maintenance de ces parties communes aux pages du site, il est préférable de les isoler dans des fichiers spécifiques et d'y faire référence *via* l'instruction `include`. Ainsi la modification du seul fichier d'en-tête provoquera la modification de toutes les pages y faisant référence. Le programme précédent est transformé en trois fichiers :

- `site_avec_include.php` contenant les informations du programme précédent avec deux syntaxes `include` à la place des lignes d'en-tête et de pied de page ;
- `site_include_header.php` contenant les lignes d'en-tête ;
- `site_include_footer.php` contenant les lignes de pied de page.

La figure 2.5 présente cette organisation.

Listing 2.5 – Programme site_avec_include.php

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Site Assistance</title>
 </head>
 <body>
   <?php include("site_include_header.php"); ?>
   <!-- Le contenu -->
   <div id="contenu">
    <h1>Assistance</h1>
    <p>
     Vous &ecirc;tes sur le site d'assistance !<br/>
     .........<br/>
    </p>
   </div>
   <?php include("site_include_footer.php"); ?>
 </body>
</html>
```

Listing 2.6 – Programme site_include_header.php

```
<header>
 <!-- Les autres sites -->
 <nav id="autres_sites">
  <div class="element_autres_sites">
  <h3>Sites de recherche</h3>
   <ul>
    <li><a href="http://www.yahoo.fr">Rechercher sur Yahoo</a></li>
    <li><a href="http://www.google.fr">Rechercher sur Google</a></li>
   </ul>
  </div>
 </nav>
</header>
```

Listing 2.7 – Programme site_include_footer.php

```
<!-- Le pied de page -->
<footer id="pied_de_page">
   <p>Contact : assistance@assit.fr</p>
</footer>
```

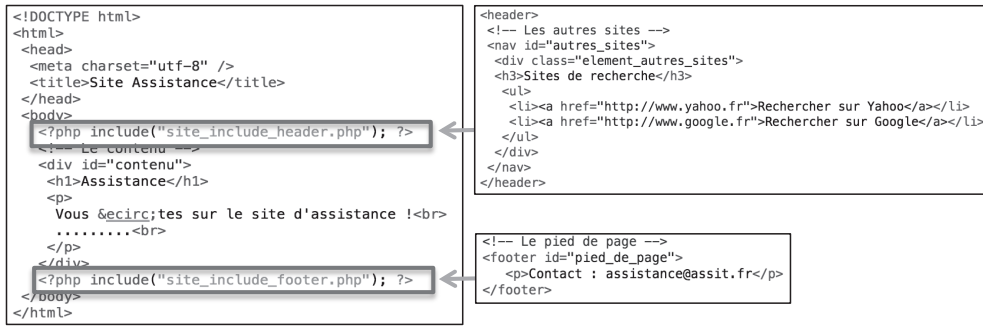

Figure 2.5 –Structure avec include.

2.3 Éléments du langage

Cette section présente quelques éléments structurant du langage PHP.

Les constantes

Principe et convention de nommage

Une constante est une valeur fixe (entière, réelle, chaîne de caractères…), non modifiable (contrairement aux variables). Il est possible d'affecter un *nom* (identifiant) à une constante, pour la rendre plus « lisible » dans le programme. Ce nom est sensible

à la casse (majuscule/minuscule), il commence par un caractère, ou le souligné, suivi par un nombre quelconque :

- de lettres (A-Z, a-z) ;
- de chiffres (0-9) ;
- du caractère souligné (_).

Voici quelques exemples de noms de constantes valides et non valides :

- `MESSAGE` : valide ;
- `2Message` : non valide ;
- `Message_2` : valide.

Par convention, le nom des constantes est toujours en majuscules.

Portée de la déclaration d'une constante

Une constante est accessible de manière *globale*. On la définit n'importe où dans le programme, et elle est accessible depuis n'importe quelle fonction. Il est cependant recommandé de déclarer les constantes au début du programme afin de les « retrouver » facilement.

L'instruction *define*

La définition de constantes utilise l'instruction `define`. Le programme `constante.php` déclare la constante `MESSAGE`, puis affiche cette valeur :

Listing 2.8 – Programme constante.php

```
<?php
 define("MESSAGE","Bienvenue sur ce site !");
 echo MESSAGE;
?>
```

Une constante n'est pas une variable, il n'y a aucun « $ » devant son nom !

Le préfixe *const* en programmation objet

La syntaxe précédente ne s'applique pas à la programmation objet, pour déclarer des constantes dans une classe. Le programme `constante2.php` déclare :

- les constantes `VALEUR_MIN` et `VALEUR_MAX` *via* l'instruction `define` en dehors d'une classe ;
- les constantes `PARAM_VALEUR_MAX` et `PARAM_VALEUR_MAX` *via* le préfixe `const` dans la classe `Parametres`.

La figure 2.6 présente son exécution.

Listing 2.9 – Programme constante2.php

```
<?php
 define('VALEUR_MIN', '0.0'); // Déclaration en dehors
 define('VALEUR_MAX', '1.0'); // d'une classe
 // -- Classe Paramètres --
 class Parametres
 {const PARAM_VALEUR_MIN = 0.0;  // Déclaration dans la classe
  const PARAM_VALEUR_MAX = 1.0;
  public static function RetourValMin()
  {return self::PARAM_VALEUR_MIN;
  }
  public static function RetourValMax()
  {return self::PARAM_VALEUR_MAX;
  }
 }
 echo "VALEUR_MIN : ".VALEUR_MIN."<br/>";
 echo "VALEUR_MAX : ".VALEUR_MAX."<br/>";
 $Mes_Parametres = new Parametres();
 echo "PARAM_VALEUR_MIN:".$Mes_Parametres->RetourValMin() ;
 echo "<br/>";
 echo "PARAM_VALEUR_MAX : ".$Mes_Parametres->RetourValMax() ;
 echo "<br/>";
?>
```

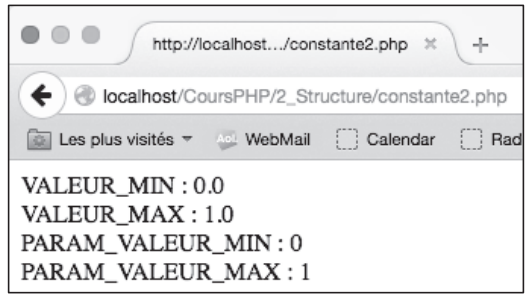

Figure 2.6 –Exécution de constante2.php.

Les constantes prédéfinies du langage

Le langage PHP propose un certain nombre de constantes prédéfinies comme ;

- `PHP_VERSION` : la version de PHP ;
- `PHP_MAXPATHLEN` : la longueur maximale d'un nom de fichier ;
- `PHP_EOL` : le caractère fin de ligne selon la plateforme.

La liste de ces constantes est accessible à l'URL :

`http://php.net/manual/fr/reserved.constants.php`

La sensibilité à la casse

En PHP, la sensibilité à la casse dépend de l'élément du langage :

- le nom des variables est sensible à la casse (majuscules/minuscules) :
- `$Nom` est différent de `$nom` ;
- les autres éléments du langage (fonction, instructions…) ne le sont pas :
- `include()` est identique à `INCLUDE()`;
- la sensibilité à la casse des noms de fichiers est dépendante du système d'exploitation. Pour en être indépendant, il est préférable de considérer que les noms des fichiers sont sensibles à la casse.

Les commentaires

Un commentaire est du texte que le développeur insère dans son programme afin d'expliquer ce qu'il fait. Il clarifie le programme en l'expliquant, aussi bien pour le développeur lui-même que pour les autres dans le cas d'un travail collaboratif.

Les commentaires sont indispensables ! Ils conditionnent la maintenabilité du programme et la possibilité de le faire évoluer facilement grâce aux explications qui aident à la compréhension des traitements implémentés.

Les commentaires sont ignorés à la génération de la page. La syntaxe des commentaires en PHP et en HTML est différente.

En HTML

Le texte mis en commentaire doit être encadré par les balises :

`<!--` et `-->`

En voici un exemple :

```
<!-- Entête HTML -->
```

Cette syntaxe ne doit pas se trouver dans la partie PHP, donc pas à l'intérieur d'un bloc délimité par les balises `<?php` et `?>`.

En PHP

Les commentaires reprennent les syntaxes du C et du C++. Il existe deux syntaxes :

- `//` devant une ligne de texte (une seule ligne) ;
- `/*` `*/` en encadrement du texte, éventuellement sur plusieurs lignes.

En voici un exemple :

```
<?php
  // --- On récupère la donnée ---
  $nom=$_POST['nom'];
  /* On traite la donnée
     en la convertissant en majuscules */
  $nom=strtoupper($nom);
  echo $nom ;
?>
```

Cette syntaxe doit être uniquement utilisée dans la partie PHP, donc à l'intérieur de la partie encadrée par les balises `<?php` et `?>`.

Les caractères spéciaux

Certains caractères ont un sens particulier dans le langage PHP. En voici quelques-unes (liste non exhaustive) :

Tableau 2.1 – Les caractères spéciaux

Opérateur	Signification	Exemple
{ }	Marquage de bloc	`if ($i < 10) {...}`
/* */ ou //	Commentaires	`/* début du programme */`
[]	Élément de tableau	`$i = $tab[10] ;`
;	Fin d'une instruction	`$i = 10 ;`
" ou '	Délimiteur de chaîne de caractères	`echo "bonjour\n";`
\	Spécialise/déspécialise un caractère	`echo "\n";`
\n	Nouvelle ligne (LF)	`echo "\n";`
\r	Retour à la ligne (CR)	`echo "\r";`
\t	Tabulation	`echo "\t";`