PHP

Présentation

Parmi les nombreux langages qui permettent de construire des pages web dynamiquement en fonction d'un contexte et des demandes de l'utilisateur, le langage PHP a bénéficié de son association avec le principal serveur Web (Apache) et un système de gestion de base de données qui, en quelques années, a évolué jusqu'à devenir un outil professionnel relativement complet et performant (MySQL). Il ne s'agit pas de présenter ici toutes les options du langage, des sites et livres le font de manière exhaustive, mais de décrire les grands principes de ce langage et les utilisations les plus courantes pour gérer :

- L'intégration de PHP avec le HTML
- La récupération de données depuis un formulaire
- La relation avec une base de données
- La gestion des informations tout au long de la navigation de l'utilisateur grâce aux variables de session
- La réutilisation de code à plusieurs endroits (plusieurs pages ou plusieurs endroits dans une page)

I Quelques principes

Exécution du PHP

Comme tout langage de script, la partie PHP est traitée uniquement du côté serveur, et l'utilisateur final ne reçoit que le résultat au format HTML du traitement de ce code.

Syntaxe

PHP est un langage assez permissif : les variables n'ont pas besoin d'être déclarées, on peut les utiliser à tout moment, changer leur type en cours de code, etc. Cela peut être un souci car une erreur de frappe (par exemple une variable $text que l'on tape accidentellement $texte) ne créera pas d'erreur d'exécution, mais peut empêcher le fonctionnement du programme. Il est donc impératif d'être extrêmement rigoureux.

Erreurs

Les messages d'erreur générés par le serveur d'exécution de PHP sont relativement explicites. Cependant, il peut parfois pointer sur une ligne qui n'est pas la bonne du fait de l'oubli d'un ; ou une mauvaise fermeture d'accolade ou de guillemet. L'utilisation d'un éditeur à colorisation syntaxique permet de limiter ce genre d'oubli.

Débogage

Quelques environnements d'exécution permettent de tester et de dépanner du code PHP grâce à une exécution pas à pas, la pose d'espions (affichage du contenu d'une variable en cours d'exécution) ou de points d'arrêt (l'exécution s'arrête sur une ligne indiquée et permet le suivi pas à pas) :

ZendStudio est un exemple d'outil offrant ces possibilités, de même que les fonction de débogage ou d'exploration des navigateurs.

Pour les producteurs de code « à la main », le débogage (recherche de la cause d'une exécution erronée) passe par des affichages (commande echo pour un champ, var_dump pour une objet structuré) du contenu des informations ou de messages pour vérifier le passage à l'intérieur d'une boucle par exemple.

Version

Les versions de PHP sont mises à jour régulièrement. Elles rendent obsolètes certaines commandes (qui restent utilisables mais non conseillées : par exemple, les commandes ***mysql_*** sont remplacées par ***mysqli_*** depuis la version PHP5, encore supportées par PHP5 mais plus par PHP7) ou en suppriment certaines et, évidemment, en ajoutent de nouvelles. Le terme pour une fonction obsolète est ***deprecated***.

On peut connaître la version de PHP installée sur une machine avec la commande :

```
php -version
```

II L'intégration de PHP avec le HTML

Il est possible de créer des pages utilisant uniquement du PHP (pour vérifier la validité d'informations, pour réaliser des insertions dans une base, etc), mais on aura souvent recours simultanément à ce langage de script et à du HTML pour produire une page à destination de l'utilisateur.

Dès lors, le PHP va déterminer quelle partie du HTML sera renvoyée à l'utilisateur final, et permettra de renseigner certains éléments en puisant depuis une base de données ou en réutilisant les données fournies plusieurs pages auparavant grâce aux variables de session.

Il y a deux approches pour réaliser l'intégration des deux langages : par le PHP seul ou en combinant HTML et PHP au moment où on en a besoin.

PHP seul

La page PHP sera intégralement constituée par du code compris entre <?php et ?>.

Tout élément qui doit être renvoyé vers l'utilisateur (donc au format HTML) devra être affiché par le biais de la commande echo.

L'avantage de cette solution est de ne pas avoir à gérer les ouverture et fermeture de <?php et ?>. L'inconvénient est la difficulté à distinguer ce qui est véritablement du code à exécuter côté serveur des éléments renvoyés vers l'utilisateur qui se trouvent noyés dans les parenthèses.

Illustration

```
<?php
echo("<html><head><title>Accueil</title></head>") ;    //entête HTML de la
page résultat
```

```
echo("<body>") ;
echo("Bienvenue sur le site") ;              //affichage de texte
echo("<hr><b><i>Aujourd'hui nous sommes le : </i>") ;    //affichage de HTML
et de texte
echo(date("d/m/Y")."</b><br>Merci de votre visite<br>");//affichage d'un
résultat php suivi de HTML et texte
    echo("</body></html>") ;
?>
```

HTML et PHP

On peut aussi choisir de ne mettre que les éléments PHP entre <?php et ?> et laisser le reste directement en HTML, évitant au serveur d'exécuter les lignes correspondantes. La difficulté de cette solution est de distinguer exactement les éléments imbriqués, de déterminer les éléments nécessitant un traitement et ceux pouvant se contenter d'un simple affichage (en particulier dans le cadre des tableaux, listes et autres objets complexes).

Illustration

```
<html><head><title>Accueil</title></head>
<body>Bienvenue sur le site
<hr><b><i>Aujourd'hui nous sommes le : </i><?php echo(date("d/m/Y") ;?>
</b><br>Merci de votre visite<br>
</body></html>
```

III La récupération de données depuis un formulaire

Un des intérêts principaux des **langages de script côté serveur** est de pouvoir récupérer des informations fournies par un utilisateur dans un formulaire et d'effectuer un traitement en fonction de la demande (choix d'information ou d'action dans une liste déroulante par exemple) ou sur les données fournies (stockage dans une base apr exemple).

Principe

Une page consultée par un utilisateur lui présente un certain nombre de zones de saisies (voir HTML), réunies au sein d'un formulaire (balise <form>) contenant un bouton de type submit (celui qui réalise l'envoie effectif des données).

Lorsque l'utilisateur clique sur le bouton ***submit***, il fait alors appel à la page précisée dans l'option ***action*** de la balise ***form***. Celle-ci récupèrera les informations transmises selon le mode de transmission précisé dans l'option ***method*** de la balise form :

- **POST** : les données sont transmises au serveur sans s'afficher dans la barre de navigation, ce qui permet une certaine confidentialité, une meilleure lisibilité de l'adresse apparaissant dans la barre de navigation

Dans les deux cas, les informations du formulaire envoyées sont transmises **en**

- **GET** : les données transmises sont lisibles dans la barre de navigation après l'adresse de la page. Exemple : http://localhost/tissusplus/execPanier.php?agir=Enregistrer. Un des avantages de cette technique est la possibilité de construire le lien en fonction du contexte (faire varier un paramètre) et d'expédier des informations déterminées par le code et non par la saisie de l'utilisateur.

clair sur la connexion réseau et sont donc interceptables.

Il ne restera plus qu'à effectuer le traitement attendu.

HTML : Formulaire

Un formulaire HTML est un ensemble de zones comprises à l'intérieur de la balise <form> ... </form>. Chaque zone a un type (checkbox, text, submit, select, etc). Elle possède un nom (option name) qui est l'information par laquelle on pourra l'appeler dans la page PHP et une valeur (information saisie ou option value). Elle peut aussi posséder un id qui permet de la traiter au sein de la page elle-même par un script côté client. Le bouton de type submit est celui qui réalise l'opération contenue dans l'option action.

Illustration

```
<form name="formCoord" action="post" method="trait.php">
    Votre nom <input type="text" name="txtNom"> </input>
    Votre prénom <input type="text" name="txtPrenom"> </input>
    Votre date de naissance <input type="text" name="txtNaiss"> </input>
    <select name="lstSexe">
        <option value="f">Femme</option>
        <option value="m">Homme</option>
    </select>
    <input type="cancel" >Annuler</input>
    <input type="submit">Enregistrer</input>
</form>
```

PHP Version simple : données connues

Si l'on souhaite récupérer les informations d'un formulaire dont on connaît exactement le nombre de données qu'il renvoie et leur nom, il suffit d'interroger les données transférées. La page trait.php pourrait alors être la suivante.

Illustration

```
<?php
    $leNom=$_POST['txtNom'] ;        //récupération de la valeur de la zone
txtNom dans une variable
    $lePrenom=$_POST['txtPrenom'] ;        //récupération de la valeur de la
zone txtPrenom
    $dateNaiss=$_POST['txtNaiss'] ;        //récupération de la valeur de la
```

```
zone txtNaiss
    $leSexe=$_POST['lst'] ;            //récupération de la valeur de la
zone lstSexe
    echo("Bonjour, ") ;
    if ($leSexe=="f")
        {echo("Madame ") ; }
    else
        {echo("Monsieur ") ; }
    echo($lePrenom." ".$leNom."<br>") ;
?>
```

PHP Version complexe : nombre de données inconnu

Dans le cas où le nombre d'informations transmises ou que les noms des zones du formulaire ne sont pas connus a priori, on va devoir récupérer les informations en parcourant le tableau des informations renvoyées. La page pourrait alors être la suivante

Illustration

```
<?php
    foreach( $_POST as $Nom => $Valeur) {
        echo("Pour le champ ".$Nom.", vous avez saisi la valeur ".$Valeur) ;
    }
?>
```

IV La relation avec une base de données

En tant que langage côté serveur, PHP autorise l'interaction avec une base de données. PHP a la particularité d'être étroitement lié avec le SGBD MySQL et de proposer directement les outils de dialogue avec cette base particulière. Toutefois, pour interagir avec tout type de base, on pourra aussi passer par une source ODBC (Open DataBase Connectivity) dont la mise en œuvre est décrite en fin de document.

Principes

Dans les deux cas, on procèdera selon les trois étapes suivantes pour dialoguer avec le SGBD : 1. Etablissement de la connexion 2. Actions sur la base 3. Fermeture de la connexion

On gèrera les erreurs en ajoutant l'instruction or die("message d'erreur adapté au contexte") à la suite de toute opération.

Connexion

La connexion à une base renvoie un pointeur qui servira par la suite à indiquer sur quelle connexion

s'effectue l'échange. La particularité de MySQL est de nécessiter d'indiquer le nom de la base au sein du SGBD (ce qui est fait par la source ODBC dans l'autre cas).

MySQL

```
$cnx = mysqli_connect("nomMachine","nomUtil","MotPasse") or die("erreur de connexion à la base");
mysqli_select_db("touttissus",$cnx);     //choix de la base
```

ODBC

```
$cnx = odbc_connect("dsnTissus","nomUtil","MotPasse") or die("erreur de connexion à la base");
```

Interrogation

Pour interroger une base de données, on doit lui fournir la requête à exécuter (un SELECT), qui retourne un curseur ou resultset qu'il faudra ensuite parcourir.

En PHP, on procèdera à l'interrogation grâce à :

- Le texte de la requête (qu'il est préférable de renseigner dans une variable chaîne pour la lisibilité)
- Le pointeur résultant de l'exécution de la requête
- Le curseur de parcours du résultat.

MySQL

```
echo("<select name='lstProd'>") ;
// chaine de caractère pour requête SQL
$reqProduits="Select PRO_CODE, PRO_NOM from PRODUIT ";
//exécution de la requete pour les produits ou affichage d'un message d'erreur
$resLesProd=mysqli_query($reqProduits) or die ("erreur sur la requete");
//variable curseur pour le parcours de la table des produits. Tant qu'il est possible de progresser dans ce curseur
while ($cursLesProd=mysqli_fetch_array($resLesProd)) {
    echo("<option value='".$cursLesProd['PRO_CODE']."'>".$cursLesProd['PRO_NOM']."</option>");
}
```

ODBC

```
echo("<select name='lstProd'>") ;
// chaine de caractère pour requête SQL
$reqProduits="Select PRO_CODE, PRO_NOM from PRODUIT ";
//récupération d'un pointeur mémoire vers le résultat
$resLesProd=odbc_exec($cnx,$reqProduits) or die ("erreur sur la requete");
//variable curseur pour le parcours de la table des produits. Il n'y a pas de variable résultat
```

```
while ($cursLesProd=odbc_fetch_array($resLesProd)) {
    echo("<option
value='".$cursLesProd['PRO_CODE']."'>".$cursLesProd['PRO_NOM']."</option>");
}
```

Requête exécution (insertion, suppression, modification)

A la différence de la manipulation précédente, les requêtes exécution ne retournent pas de résultat à parcourir. L'écriture est donc légèrement simplifiée.

MySQL

```
// chaîne de caractère pour requête SQL
$reqProduits="Insert into compte values
('".$_POST['txtUtil']."','".$_POST['txtMP']."','".$_POST['txtRole']."')";
//exécution de la requête pour les produits ou affichage d'un message
d'erreur
$execLesProd=mysqli_query($reqProduits) or die ("erreur sur la requête");
```

ODBC

```
// chaîne de caractère pour requête SQL
$reqProduits="Insert into compte values
('".$_POST['txtUtil']."','".$_POST['txtMP']."','".$_POST['txtRole']."')";
$execLesProd=odbc_exec($cnx,$reqProduits) or die ("erreur sur la requête");
```

Déconnexion

Dans les deux cas, la déconnexion de la base est une nécessité. En effet, l'ouverture d'une connexion monopolise des ressources sur le serveur (mémoire, nombre d'accès simultané à la base), et il faut les libérer pour ne pas le saturer ou empêcher d'autres interrogations.

La syntaxe est identique dans les deux cas :

```
$cnx.close ;     //$cnx est la variable définie lors de l'utilisation de la
méthode connect
```

Comparaison MySQL et ODBC

On voit que la syntaxe diffère très peu entre les deux manipulations.

- L'avantage de MySQL est son lien étroit et direct avec le langage PHP (les deux ont été conçus en parallèle). On peut donc supposer que l'interpréteur PHP est optimisé pour interroger une base PHP. D'ailleurs, l'ensemble Apache/MySQL/PHP est devenu un standard en version Linux (LAMP), Windows (WAMP) ou MACOs (MAMP). Cependant, la migration vers une autre base, pour des raisons de performances ou de capacité technique (procédures stockées, réplication, etc) nécessitera la réécriture de toutes les pages. Il sera intéressant à ce titre de proposer une page

de fonctions pour toute interaction avec la base, et seule cette page sera à modifier.
- ODBC (ou d'autres technologies équivalentes) offre un avantage de compatibilité et d'universalité (y compris vers une base MySQL). Cependant, en ajoutant un intermédiaire, on augmente le temps de traitement (peu, mais sur un nombre de requêtes conséquent, l'impact peut être non négligeable). En outre, certaines manipulations standard en SQL créent des erreurs lorsqu'elles passent par ODBC (par exemple, avec Access, la clause LIKE du langage SQL ne fonctionne pas).

V La gestion des informations tout au long de la navigation : variables de session

Avec les formulaires HTML, il est possible de récupérer côté serveur un ensemble d'information fournies du côté du poste client.

Pour pouvoir conserver des informations au delà de cette page de récupération, il faut que le serveur stocke en mémoire les données qu'on lui aura indiqué (ou d'autres), et qu'il puisse les restituer à la demande. Il n'est pas possible de réaliser cette conservation côté client car c'est le code PHP qui est sensé réutiliser cette information. On utilise alors la technique des variables de session qui sont initialisées dans le code PHP et conservées tant que le même utilisateur continue d'interagir avec le serveur (tant que cet utilisateur garde sa session ouverte).

Démarche

Avant toute manipulation d'une variable de session dans une page (en lecture comme en écriture), on devra utiliser la commande

```
session_start() ;
```

Les variables de session sont définies dans l'objet $_SESSION

```
$_SESSION['nomVar']="valeur" ;        #donne une valeur à la variable nommée nomVar
$_SESSION['nomVar']="" ;           #vide la variable de session, elle est détruite
```

Pour savoir si une variable de session existe, on la testera ainsi :

```
if (isset($_SESSION['nomVar'])) {…}
```

Illustration

```
Formulaire                  Page qui récupère les données et les met en session
```

VI La réutilisation de code : fonctions et include

Dans tout programme, on cherche à optimiser le code en factorisant les éléments récurrents, ceci dans un objectif d'écrire un code compact, de faciliter la mise à jour (on n'a pas besoin de modifier la portion réutilisée alors que le code écrit à plusieurs reprises doit être modifié autant de fois qu'il apparaît) et de créer des outils réutilisables.

Fonctions

En programmation, on a recours aux procédures (qui exécutent une série d'actions) et fonctions (qui retournent un résultat final) qui permettent de créer une portion de programme autonome qui peut être appelée à plusieurs endroits. En PHP, seule la dénomination de fonction existe. Une procédure sera alors une fonction qui ne retourne pas de valeur.

Illustration

```
//fonction qui retourne le plus grand des deux paramètres
function plusGrand($p1, $p2) {
    if ($p1>$p2)      { $result=$p1 ;)
    else             {$result=$p2 ;}
    return $result ;    //retourne l'information
}
//procédure qui affiche le contenu d'un tableau dont on connaît le nombre de
case sous forme d'une liste à puce
function affTabloListe($ptablo ,$pNbCases) {
    echo("<ul>") ;
    for ($cpt=1 ;$cpt<=$pNbCases ;$cpt++) {
        echo("<li>".$ptablo[$cpt]) ;
    }
    echo("</ul>") ;
}
//…….Autre code
$tabUtil[1]="Lundi" ;
$tabUtil[2]="Mardi" ;
$tabUtil[3]="Mercredi" ;
$tabUtil[4]="Jeudi" ;
$tabUtil[5]="Vendredi" ;
//appel de la procédure
affTabloListe($tabUtil,5) ;
//*******************************Autre code
$valeur=$_POST['NombreSaisi'] ;   //récupère une info saisie par
l'utilisateur dans un formulaire
$nombre=rand(1,15) ;         //génère un nombre aléatoire entre 1 et 15
//utilisation de la fonction définie plus haut
if (plusGrand($valeur,$nombre)) {
    echo("Bravo, vous êtes le plus fort") ;
}
```

Les fonctions sont donc un moyen pratique de factoriser le code pour pouvoir le réutiliser en lui fournissant en paramètre effectif les informations propres au contexte d'exécution.

Include

Pour éviter de réécrire une portion de code dans plusieurs pages, on peut l'extraire dans une page indépendante à laquelle on fera appel par le biais de la commande include. Le serveur prendra alors la page appelante, y ajoutera le contenu de la page incluse, puis exécutera l'ensemble comme s'il s'était agi d'une page unique.

Cette technique permet notamment de constituer une bibliothèque de fonctions (par exemple la gestion des échanges avec une base de données, depuis la connexion jusqu'à la fermeture en passant par l'interrogation) à laquelle on fera appel à tout instant.

Illustration

Fichier de fonction (biblioFct.php)

```
<?php
   // retourne le plus grand des deux paramètres
   function plusGrand($p1, $p2) {
    if ($p1>$p2)      { $result=$p1 ;)
    else             {$result=$p2 ;}
    return $result ;    //retourne l'information
}
   // affiche le contenu d'un tableau en liste à puce
   function affTabloListe($ptablo ,$pNbCases) {
    echo("<ul>") ;
    for ($cpt=1 ;$cpt<=$pNbCases ;$cpt++) {
        echo("<li>".$ptablo[$cpt]) ;
    }
    echo("</ul>") ;
}
?>
```

Fichier utilisateur de la bibliothèque

```
<?php
  //demande l'incorporation de la bibliothèque
  include("biblioFct.php") ;
//autre code
$tabUtil[1]="Lundi" ;
$tabUtil[2]="Mardi" ;
$tabUtil[3]="Mercredi" ;
$tabUtil[4]="Jeudi" ;
$tabUtil[5]="Vendredi" ;
//appel de la procédure
affTabloListe($tabUtil,5) ;
?>
```

VII Connexion à un annuaire

PHP autorise la connexion avec un annuaire via la bibliothèque ***php-ldap***.

Installation

Pour cela, il faut installer le module correspondant :

```
apt install php-ldap
```

On active ensuite le module pour Apache que l'on redémarre ensuite :

```
a2enmod ldap
systemctl restart apache2
```

Code PHP pour LDAP

La connexion passe par deux étapes :

1. connexion à l'annuaire avec ***ldap_connect()***
2. interrogation de la base d'annuaire avec un compte grâce à ***ldap_bind***

Cela donne par exemple :

```
<?php
   // Fichier de configuration pour l'interface PHP
   //  de notre annuaire LDAP
   $server = "172.20.168.100";
   $port = "389";
   $racine = "dc=tts,dc=intra";
   $rootdn = "user@tts.intra";
   $rootpw = "@mpU5R";
   echo "Connexion...<br>";
   //connexion au serveur
   $ds=ldap_connect($server);
   //si la connexion a réussi
   if ($ds)
      {        // on s'authentifie en tant qu'utilisateur pour vérifier que
les informations sont correctes
       $r=@ldap_bind($ds,$rootdn,$rootpw);
       if ($r){ //si les informations sont correctes, $r vaut 'true'
                // Ici les opérations à effectuer
              }
   echo "Déconnexion...<br>";
   ldap_close($ds);
  }
else {
```

```
 echo  "Impossible de se connecter au serveur LDAP";
  }
?>
```

Éléments du langage PHP

Voici quelques éléments du langage. On trouvera de nombreux sites de référence présentant tous les détails du langage, des scripts prêts à l'emploi, des tutoriaux complets. Notamment :

- http://www.phpsources.org/ (multiples sources et scripts, éléments du langage)
- http://www.php.net/manual/fr/langref.php (manuel complet du langage)
- http://www.atelierphp.net/intro/index.php (cours, exercices et QCM)
- http://www.commentcamarche.net/contents/php/phpintro.php3 (présentation des éléments du langage très bien structurée)
- http://www.apprendre-php.com (cours et scripts, très clair)

Syntaxe de base

Pour écrire du PHP, il faut respecter quelques principes :

- Tout code PHP est compris entre <?php et ?>
- Toute ligne de code est terminée par un ;
- PHP est sensible à la casse (majuscule et minuscule)
- Les commentaires s'écrivent derrière deux '/' ou entre '/* ... */'
- Toute variable commence par un $
- Les variables ne doivent pas être déclarées, elles sont typées par leur utilisation
- Les tableaux sont notés avec des crochets : $tablo[] ou $tablo['indice']
- Les conditions (dans un si ou une boucle) s'écrivent entre parenthèses : if (condition) ...
- Une portion de code dépendante (qui concerne un si ou une boucle) s'écrit entre accolades : if (condition) {...}
- La concaténation de chaînes de caractères (mise bout à bout) se fait par le caractère « . »
- La comparaison d'égalité utilise = =
- La négation s'écrit ! (!= ... !(condition))
- L'affichage d'informations sur le fichier de sortie se fait par la commande echo

Astuces

- L'incrémentation (ajout de 1 à une valeur numérique) s'écrit ++ (exemple $i++)
- On peut combiner l'affectation (=) avec une autre opération (concaténation, ajouts et multiplications).

Exemple : plutôt que d'écrire $texte=$texte." Euros", on écrira $texte.=" Euros". De même, on simplifiera $num=$num+5 par l'écriture $num+=5, et $val=$val*10 par $val*=10 ;

Conditions

Les conditions utilisent les éléments de comparaison et de logique suivant :

- = = : retourne vrai en cas d'égalité entre deux éléments
- != : retroune vrai si les deux éléments sont différents
- >, >= : comparaison de supériorité
- <, ⇐ : comparaison d'infériorité
- AND ou && : conjonction de conditions (condition1 && condition2)
- OR ou || : disjonction de conditions (condition1 OR condition2)
- !(condition) : test l'inverse de la condition (! est le caractère de négation)

Structures de contrôle

Les principales opérations de contrôle (alternatives, boucles, choix multiples) de PHP sont présentées ci-dessous.

	Syntaxe	Exemple

Alternative	if (condition)
	{code_du_alors}
	[elseif (condtion2) {code_deuxième_condition ;}]
	[else {code_du_sinon)]
		`if ($age<=16)`
		`{$trancheAge="junior" ;}`
		`elseif ($age<65)`
		`{$trancheAge="adulte" ;}`
		`else {$trancheAge="senior" ;}`

Pour	for (initialisation ; test_continuation ; pas évolution) { code_de_boucle } $tabloMois=array("Janvier","Février","Mars",...) ;	`for ($indice=0 ; $indice<12 ;$indice++) {` `echo( ($indice+1)."-` `".$tabloMois[$indice]." ") ;` `}`

Tant que	while (condition_continuation) {
	code_boucle
	}
		`while (...) {`
		`...`
		`}`

Répéter	do {
	code_boucle
	}
	while (condition_continuation)
		`do { \`
		`... }`
		`while (...)`

Pour chaque	#boucle qui parcourt le contenu d'un tableau
	foreach ($tableau as $valeur) {
	code_boucle
	}
		`$tMoisJour=array`
		`("Janvier"=>31,"Février"=>28,"Mars"=>31,…)`
		`;`
		`foreach ($tMoisouJr as $nom =>$nbJours) {`
		`echo("le mois de ".$nom." comporte`
		`".$nbJours." Jours") ;`
		`}`

Selon	switch ($variable) {
	case valeur1 : code_1 ; break ;
	case valeur2 : code_2 ; break ;
	...
	default : code_pour_autres_valeurs ;
	}
		`switch ($trancheAge) {`
		`case 'senior' :`
		`$reduction="Vermeille" ; break ;`
		`case 'adulte' : $reduction="Plein`
		`Tarif" ; break ;`
		`case 'junior' : $reduction="Enfant" ;`
		`break ;`
		`default : $reduction="Plein Tarif" ;`
		`break ;`
		`}`

Quelques commandes

Renvoi vers une autre page : une page PHP peut rediriger vers n'importe quelle page, sous réserve qu'elle n'ait pas déjà commencé à construire une page à retourner vers l'utilisateur (il ne doit pas y

avoir de code HTML ou d'utilisation de la commande echo). On utilise la syntaxe :

```
header("Location:nomPageRedirection") ;
```

ANNEXE 2 : MISE EN PLACE D'UNE SOURCE DE DONNEES ODBC

Middleware et ODBC Pour ne pas limiter l'accès au SGBD par les seuls outils propre à l'éditeur, un principe d'accès universel aux différents SGBD a été défini : cet intermédiaire prend des demandes au format standard SQL et les transforme dans la syntaxe propre au SGBD sollicité. On appelle ces outils des middlewares (ou intergiciels), dont l'exemple le plus connu est ODBC (Open DataBase Connectivity) promu par Microsoft.

Création d'une source

Dans le panneau de Configuration, on aura recours à l'outil ODBC 32 Bits

- Ajoute, supprime ou définit des sources de données avec des DSN utilisateur. Ces sources de données sont spécifiques à un ordinateur (locales) et ne peuvent être utilisées que par l'utilisateur en cours.
- Ajoute, supprime ou définit des sources de données avec des DSN fichier. Il s'agit de sources de données basées sur des fichiers qui peuvent être partagées par tous les utilisateurs disposant des mêmes pilotes installés et qui ont donc accès à la base de données. Il n'est pas nécessaire que ces sources de données soient associées (dédiées) à un utilisateur ou spécifiques à un ordinateur (locales).
- Ajoute, supprime ou définit des sources de données avec des DSN système. Ces sources de données sont spécifiques à un ordinateur (locales), et non associées (dédiées) à un utilisateur. Le système, ou tout utilisateur disposant d'autorisations d'accès, peut utiliser une source de données définie avec un DSN système.

En cliquant sur , on obtiendra l'écran ci-contre qui permet de choisir un pilote ODBC parmi ceux installés sur l'ordinateur.

Une fois le pilote choisi, on devra localiser la base correspondante et donner un nom à notre DSN. L'écran est sensiblement différent selon le SGBD auquel on accède, et le pilote qui est utilisés. Toutefois, en plus du nom de la source qui sera appelé lors de l'utilisation de cette connexion, on pourra être amené à fournir aussi le nom de la base (cas pour MySQL ou Oracle) ainsi que les coordonnées d'un utilisateur autorisé à accéder aux informations.

Nom générique de la source que l'on utilisera par la suite Descriptif textuel qui explicite le rôle de la source (quelle base, son type, le compte utilisé, etc)

Permet de sélectionner une base existante ou d'en créer une nouvelle

Il est dès lors possible d'accéder à la base correspondante à travers n'importe quel langage de programmation (notamment les langages Web) ou environnement (par exemple, exploiter une base MySQL par des interfaces Access).