# Technologies de Script

Mr Mohamed SIDIR & Mr Gérard-Michel Cochard

**Université Virtuelle de Tunis**

2007

# Notions de JavaScript

**Gérard-Michel Cochard**

**Sommaire :**


Qu'est-ce que c'est ?
Bases du langage JavaScript
Pseudo programmation orientée objet
Programmation événementielle
Exercices
Solution des exercices

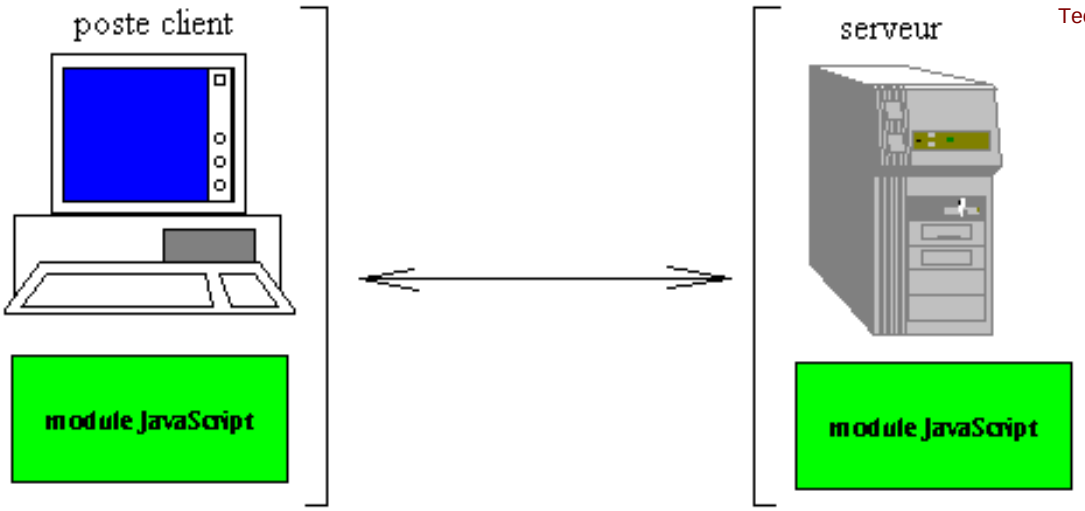

## Qu'est-ce que c'est ?

**HTML** (HyperText Markup Language) est le **langage de description** d'un document "visualisable" sur un poste client, doté d'un logiciel navigateur et connecté à un serveur WWW. HTML **n'est pas un langage de programmation** permettant la réalisation d'applications dynamiques.

**Java**, d'origine Sun, est un **langage de programmation** orienté objet ; il nécessite un compilateur (javac) et un interpréteur. Il peut s'utiliser comme langage de programmation ordinaire, indépendamment du Web. Il est toutefois surtout utilisé pour la confection d'**applets**, petits programmes incorporés à un document HTML et qui, s'exécutant sur le poste client, provoquent par exemple des effets d'animation.

**JavaScript**, d'origine Netscape (LiveScript) est également un **langage de programmation** (pseudo)orienté objet mais ce langage est seulement interprété et ne s'utilise que incorporé à un document HTML ; l'exécution du script, comme dans le cas des applets Java, s'effectue sur le poste client.

Bien entendu, le serveur comme le poste client doit posséder un module JavaScript pour que la communication et la compréhension soient possibles :




ce qui signifie que, pour utiliser JavaScript, il faut que le serveur comme le poste client "comprenne" ce langage ! De nos jours, la plupart des serveurs standards intègrent JavaScript ; du côté poste client, les deux poids lourds que constituent Netscape Communicator et Internet Explorer 4 intègrent également JavaScript (mais pour les autres navigateurs cela est moins sûr). Au moment où ces lignes sont écrites, JavaScript en est à sa version 1.2 (Netscape) ; de son côté Microsoft a créé JScript (version de JavaScript à la sauce Microsoft) ; mais il s'agit grosso modo du même langage.

Souvent, Java et JavaScript sont comparés comme si ils étaient équivalents. Ce n'est évidemment pas le cas. Le tableau ci-dessous montre les différences entre Java et JavaScript.

| JavaScript | Java |
|---|---|
| langage interprété par le client | langage compilé chargé sur le poste client et exécuté par celui-ci |
| "basé" objet | "orienté" objet |
| code intégré dans HTML | Applets distincts de HTML |
| types de données non déclarés | types de données déclarés obligatoirement |
| liaison dynamique : références aux objets vérifiées lors de l'exécution | liaison statique : références aux objets existant lors de la compilation |

L'expression "basé" objet est équivalente à "pseudo-orienté" objet. En effet, comme pour Java, tout objet dans JavaScript possède des propriétés et des méthodes, mais la similitude s'arrête pratiquement là.

Examinons le document HTML ci-dessous qui nous permettra un premier contact avec JavaScript.

Exemple 1

```
<HTML>
<HEAD>
<TITLE>Exemple 1</TITLE>
</HEAD>
<BODY>
Voici un premier exemple JavaScript
<P>
<SCRIPT LANGUAGE="JavaScript">
document.write("Bonjour les petits
enfants ! "); </SCRIPT>
</P>
```

L'application écrite en JavaScript se situe entre la balise <SCRIPT> et la balise </SCRIPT>. La balise <SCRIPT> possède un attribut LANGUAGE qui permet d'indiquer au navigateur en quel langage est écrit le script (on peut, en effet, utiliser d'autres langages de script, Visual Basic en est un exemple courant).

Dans l'exemple 1, l'application consiste en une seule instruction qui fait appel à un objet "document" et à la méthode

```
</BODY>
</HTML>
```

write() attachée à cet objet. Technologie de Script

On notera le signe ";" qui indique la fin de l'instruction JavaScript.

exemple 1

En fait, il existe trois façons d'introduire du code JavaScript dans HTML :

- 1ère façon : celle que l'on vient de voir dans l'exemple 1. On utilise la balise <SCRIPT> :
  <SCRIPT LANGUAGE="JavaScript">
  code JavaScript
  </SCRIPT>
- 2ème façon : insertion d'un module de traitement d'événement. Par exemple, on peut introduire un bouton avec une balise <INPUT> :
  <INPUT TYPE="Text" value="Cliquez ici" onClick="document.write('coucou');">. Tel est le cas de l'exemple 2 qui utilise un formulaire avec un bouton pour afficher un message d'alerte.
- 3ème façon : en référençant un fichier spécial (.js) dans un document HTML. L'exemple 3 en donne une illustration.

Les avantages de JavaScript sont évidents :

- simplicité d'écriture et de mise en oeuvre
- interprétation effectuée par le navigateur (ce peut être aussi un inconvénient !)
- compatibilité assez large (fonctionne sur PC et sur Mac)

Les inconvénients de JavaScript sont hélas plus nombreux que les avantages :

- l'utilisateur a accès au code source et par suite la confidentialité de ce code n'est pas assurée d'où l'obligation éventuelle de mentionner un copyright
- quelquefois assez lent
- problèmes de sécurité : intrusion manifeste sur le poste client : JavaScript peut connaître l'activité du poste client et en faire part au serveur ; ces problèmes sont relatifs aux premières versions des navigateurs (Netscape) ; ils sont, parait-il, résolus maintenant.
- Pour des scripts importants (c'est à dire longs), la taille du document WWW (HTML + code JavaScript) ne peut excéder 32 Ko. Pour sortir de cette impasse, on peut charger à part le fichier contenant le code JavaScript (enregistré avec l'extension .js) avec l'attribut "SRC".
- Il existe des navigateurs qui ne reconnaissent pas JavaScript. On utilise alors dans ce cas la balise <NOSCRIPT>....</NOSCRIPT>.

## Quelques références

Pour ceux qui veulent en savoir plus, on peut trouver sur Internet un certain nombre de documents sur JavaScript :

http://www.imaginet.fr/ime/javascri.htm
http://www.yahoo.com/Computers_and_Internet/Programming_Languages/JavaScript/
http://www.essex1.com/people/timothy/js-index.htm
http://developer.netscape.com/docs/manuals/communicator/dynhtml/index.htm
http://www.geocities.com/Athens/8281/

Un petit guide pour débutants :

http://www.javascriptguide.com/index.html



Des scripts à la pelle :
http://www.worldwidemart.com/script/
http://javascript.internet.com/

# Bases du langage JavaScript

Evidemment, JavaScript est très inspiré de Java. Un programme JavaScript est composé des éléments suivants :

- déclaration de variables
- blocs d'instructions
- fonctions
- modules de traitement

## Variables

Les variables peuvent être locales ou globales.
Une variable est **locale** si elle n'a de portée que dans la fonction où elle figure ; dans ce cas elle doit être déclarée avec le mot-clé **var**.
Une variable qui n'est pas déclarée avec le mot-clé var est **globale** : sa portée est celle du programme.
Les variables désignent par des noms des données de types définis. Les types de données de JavaScript sont les suivants :

- **entier** et **décimal** :
  64 3.14 6.023E+23
- **booléen** : true et false
- **chaîne de caractères** : "Bonjour", 'Bonsoir'
- **null** : valeur d'une variable indéfinie

Notons que JavaScript utilise des caractères spéciaux (comme Java) :

- \n : nouvelle ligne
- \f : saut de page
- \r : retour chariot
- \t : tabulation horizontale
- \b : retour arrière

## Fonctions

Elles appartiennent à deux types :



- les fonctions intégrées, c'est à dire prédéfinies par le langage

| | |
|---|---|
| parseInt() | conversion en entier : parseInt("400 coups") donne 400 |
| parseFloat() | conversion en décimal : parseFloat("3.14 est la valeur de pi") donne 3.14 |
| eval() | cette fonction évalue toute expression et restitue une réponse valide<br>exemple : eval("20+7+6") donne la valeur numérique 33<br>eval("Alfred a 5 ans") donne la chaîne "Alfred a 5 ans" |

- les fonctions définies par le programmeur

exemple :

```
function Somme(a,b) {
var resultat=a+b ;
return resultat ;}
```

Il faut distinguer la définition de la fonction et l'appel de la fonction. La définition de la fonction consiste à la décrire et à préciser ce qu'elle fait quand on l'appelle. L'appel d'une fonction, au contraire, provoque son exécution.

Dans la pratique, les fonctions définies par le programmeur sont placées dans la section <HEAD>...</HEAD> du document HTML ce qui permet leur utilisation dès que l'appel de ces fonctions a été chargé par le navigateur. Dans le cas où une fonction est définie dans la section <BODY>, il faut attendre que sa définition soient chargée pour qu'elle puisse être appelée et exécutée.

Une fonction peut retourner un résultat (avec le mot-clé return). L'exemple 23 donne une illustration du calcul d'une factorielle (ce qui montre en particulier qu'une fonction peut être récursive).

Une fonction comporte éventuellement des arguments. Ces arguments sont gérés dans un tableau à une dimension (array) et pour une fonction donnée, on peut utiliser séparément chacun de ses arguments en donnant le numéro d'ordre (le premier argument a le numéro 0) dans le tableau. L'exemple 24 illustre cette propriété. Des chaînes de caractère sont considérées comme arguments d'une fonction adherent(nom, prenom, ville, telephone). On peut afficher séparément les arguments nom, prenom, ville, telephone en précisant seulement adherent.arguments[0],adherent.arguments[1],adherent.arguments[2],adherent.arguments[3].

## Opérateurs

Les opérateurs de JavaScript sont assez courants. Ce sont ceux que l'on retrouve dans la plupart des langages :

- opérateur d'**affectation**

```
fable='Le corbeau et le renard';        x=x+1;
```

- opérateurs **arithmétiques**

| | |
|---|---|
| + - * / | les 4 opérations |
| % | modulo |
| ++ | incrémentation |
| -- | décrémentation |

- opérateur sur **chaîne de caractères** : + concaténation
- opérateurs **logiques**

| | |
|---|---|
| && | et |
| \|\| | ou |
| ! | non |

- opérateurs **binaires**

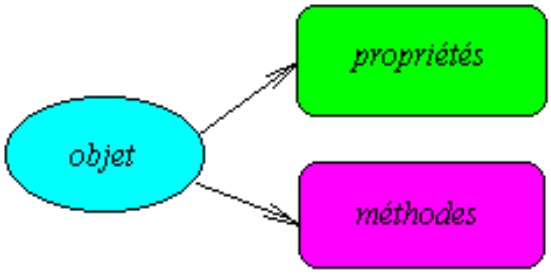
| | |
|---|---|
| & | et |
| \| | ou |
| ^ | ou exclusif |
| << | décalage à gauche |
| >> | décalage à droite |

- opérateurs de **comparaison**

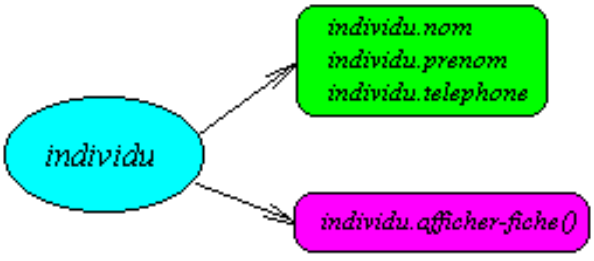
| | |
|---|---|
| == | égal à |
| < | inférieur à |
| <= | inférieur ou égal à |
| > | supérieur à |
| >= | supérieur ou égal à |

## Objets, propriétés, méthodes

Dans JavaScript, un objet désigne un type d'entité qui peut posséder des caractéristiques appelées **propriétés** et des procédures appelées **méthodes**.




Ainsi, un individu est un objet qui pourra posséder les propriétés suivantes : individu.nom, individu.prenom, individu.telephone et la méthode individu.afficher_fiche()




exemple21 :

Dans la section <HEAD>...</HEAD>, on peut définir les fonctions suivantes :

```
function afficher_fiche() {
        document.write("<B>Nom:</B>",this.nom,"<BR>");
        document.write("<B>Prénom:</B>",this.prenom,"<BR>");
        document.write("<B>Téléphone:</B>",this.telephone,"<BR>");
}
function individu(nom, prenom, telephone) {
        this.nom=nom;
        this.prenom=prenom;
        this.telephone=telephone;
        this.afficher_fiche=afficher_fiche ;
}
```

La seconde méthode est le constructeur qui permet la création d'un nouvel objet, la première est une procédure qui provoque une action (ici l'affichage de la fiche).

Dans la section <BODY>...</BODY>, on placera les instructions d'utilisation des méthodes définies ci-dessus :

```
jojo=new individu("Durand","Jojo","03 45 67 89 10");
jojo.afficher_fiche();
```

Attention ! en JavaScript, contrairement à Java, la Programmation Orientée Objet (POO) est limitée :

- Dans Java, on définit des classes qui possèdent les propriétés d'héritage, de polymorphisme, de sous-classes.
- Dans JavaScript, on définit des "pseudo-classes" qui ne possèdent pas la propriété d'héritage, qui ne possèdent pas la propriété de polymorphisme, qui ne possèdent pas de sous-classes systématiques. On reviendra sur ces aspects dans le chapitre suivant.

## Structures conditionnelles et itératives

Les syntaxes sont très inspirées de Java et de C :

- structure **conditionnelle simple** :

```
if [condition] {
       [traitement si Vrai]
}
```

- structure **conditionnelle équilibrée**

```
if [condition] {
        [traitement si Vrai]
}
else {
        [traitement si Faux]
}
```

NB : on peut aussi utiliser l'opérateur ternaire ? : comme dans l'expression

```
salut=heure<12?'bonjour':'bonsoir'
```

- boucle **définie**

```
for([initialisation];[condition d'itération];[incrémentation]) {
                [traitement]
}
```

- boucle **indéfinie**

```
while [condition]{
          [traitement si Vrai]
}
```

Les instructions break et continue permettent de modifier la séquence d'instruction d'une boucle :

- **break** permet de **sortir** immédiatement d'une boucle
- **continue** permet de passer immédiatement à l'itération **suivante**

L'exemple 22 donne un échantillon d'application des structures précédentes : On effectue une itération de 10 passages au cours desquels, on tire au sort un nombre entier compris entre 0 et 9. Si le nombre est inférieur à 5, on affiche "bonjour" ; si le nombre est supérieur à 5, on affiche "bonsoir" ; si le nombre est égal à 5, on sort de la boucle.

```
for(i=0;i<10;i++) {
    alea=Math.floor(10*Math.random());
    if (alea==5) {
        document.write('sortie');
        break ;
    }
    else {
       if (alea<5) {
          document.write('bonjour'+'<br>');
       }
       else {
          document.write('bonsoir'+'<br>');
       }
    }
}
```

Remarque : dans le script précédent, il est fait usage de l'objet Math, unique objet de la pseudo-classe Math (pas besoin de new) et des méthodes floor() qui renvoie le plus grand entier inférieur au nombre donné et random() qui renvoie un nombre aléatoire entre 0 et 1.

# Pseudo Programmation Orientée Objet

Comme il a été signalé dans les chapitres précédents, l'utilisateur peut créer des pseudo-classes avec des propriétés et des méthodes. Nous employons à dessein le terme de pseudo-classes, car en JavaScript, il n'y

a pas de notion d'héritage, ni de polymorphisme. Attention ! dans beaucoup d'ouvrages, une pseudo-classe est souvent appelée "objet". Il existe cependant des pseudo-classes pré-définies qui appartiennent à deux catégories :



- les pseudo-classes usuelles
- les pseudo-classes de fenêtrage

Dans les deux cas, la manière de noter une propriété ou un objet est la manière standard. Ainsi

document.formulaire1.text2.value

désigne le contenu de la zone de texte "text2" du formulaire "formulaire1" du document courant.

La création d'un nouvel objet consiste

- à définir le type de l'objet à l'aide d'une fonction "créateur"
- à créer une instance de cet objet avec le mot clé new.

## Pseudo-classes usuelles

Il y en a quatre : String, Date, Math, Array

- pseudo-classe String : Cette pseudo-classe est l'équivalent du String du langage Java et correspond à l'objet chaîne de caractères. Elle ne possède qu'une seule propriété : length (longueur de la chaîne) et deux types de méthodes : les méthodes de formatage, qui correspondent à un style HTML, comme la méthode bold() qui est l'équivalent de la balise <B> : "Bonjour".bold() équivaut, en effet, à <B>Bonjour</B> ; les méthodes de traitement de chaînes données dans le tableau ci-dessous

| | | |
|---|---|---|
| substring(debut, fin) | extraction d'une sous-chaîne | debut est l'index du caractère de départ (le premier caractère a l'index 0) et fin est l'index du caractère suivant le dernier caractère à prendre en compte |
| undexOf("chaîne") | recherche dune chaîne dans une chaîne | renvoie l'index du premier caractère |
| charAt(index) | extraction d'un caractère | le caractère est défini par son index |

- pseudo-classe Date : Cette pseudo-classe permet de manipuler des dates ; elle ne possède que des méthodes (pas de propriété) :

| | | | | |
|---|---|---|---|---|
| getDate() | renvoie le jour du mois | | setDate() | fixe le jour du mois |
| getDay() | renvoie le jour de la semaine(0=dimanche) | | | |
| getHours() | renvoie l'heure | | setHours() | fixe l'heure |
| getMinutes() | renvoie les minutes | | setMinutes() | fixe les minutes |
| getMonth() | renvoie le mois (0=janvier) | | setMonth() | fixe le mois |
| getSeconds() | renvoie les secondes | | setSeconds() | fixe les secondes |

| | | | | |
|---|---|---|---|---|
| getTime() | renvoie le nombre de millisecondes écoulées depuis le 1er janvier 1970 | | setTime (millisecondes) | fixe une date d'après le nombre de millisecondes écoulées depuis le 1er janvier 1970 |
| getYear() | renvoie le nombre d'années depuis 1900 | | setYear(années) | fixe une année selon le nombre d'années écoulées depuis 1900 |

- pseudo-classe Math : ne possède qu'un seul objet ; les propriétés définissent les constantes usuelles ;

| | |
|---|---|
| E | 2.71828... |
| LN10 | 2.30258... |
| LN2 | 0.693147... |
| PI | 3.14159... |
| SQRT1_2 | 0.707106... |
| SQRT2 | 1.41421... |

les méthodes expriment les fonctions usuelles

| | | | | |
|---|---|---|---|---|
| abs() | renvoie la valeur absolue | | round() | renvoie l'entier le plus proche du nombre |
| acos() | renvoie l'arc cosinus | | cos() | renvoie le cosinus de l'arc |
| asin() | renvoie l'arc sinus | | sin() | renvoie le sinus de l'arc |
| atan() | renvoie l'arc tangente | | tan() | renvoie la tangente de l'arc |
| ceil() | renvoie le plus petit entier égal ou supérieur au nombre | | floor() | renvoie le plus grand entier égal ou inférieur au nombre |
| exp() | renvoie l'exponentielle | | log() | renvoie le logarithme du nombre |
| max(a,b) | renvoie le plus grand des deux nombres a et b | | min(a,b) | renvoie le plus petit des deux nombres a et b |
| pow(b,e) | renvoie à la puissance e | | sqrt() | renvoie la racine carrée du nombre |
| random() | renvoie un nombre au hasard entre 0 et 1 | | | |

- pseudo-classe Array : elle permet de définir des tableaux à 1 indice

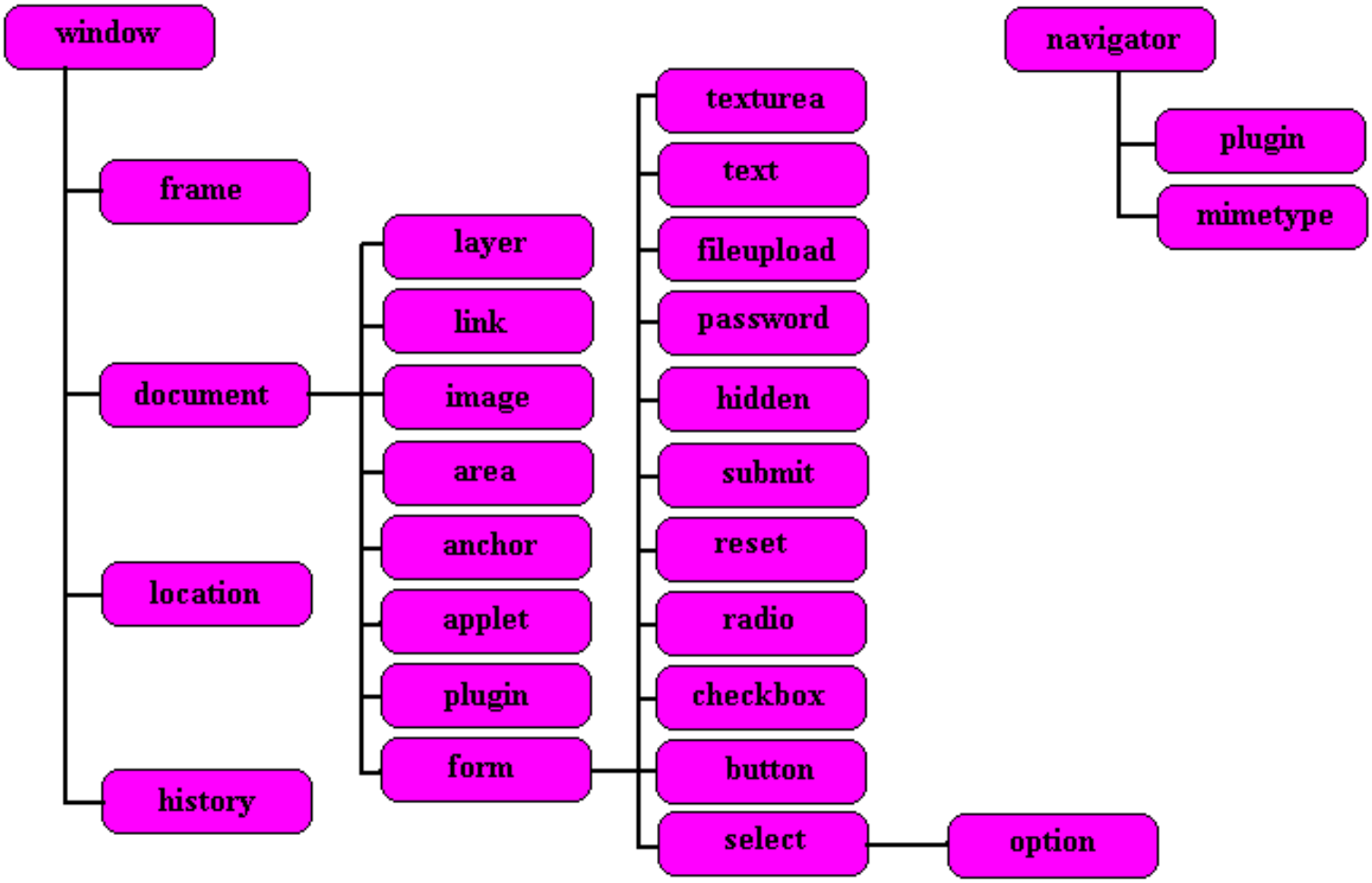
```
matrice=new Array(20) //définit 20 éléments numérotés de 0 à 19
matrice[15]=4.56 //valeur d'un élément du tableau
```

Cette pseudo-classe ne possède qu'une seule propriété : length et trois méthodes :join() : concaténation de tous les éléments en une chaîne (séparateur à préciser, sinon virgule)
sort() : tri avec critère (facultatif) à préciser
reverse() : transposition

## Pseudo-classes de fenêtrage

Ces pseudo-classes sont relatives aux objets d'affichage manipulés par le navigateur :






- window correspond aux fenêtres ouvertes par le navigateur
- frame correspond aux cadres qui opèrent éventuellement une division de la fenêtre
- document est relatif aux pages Web
- history est relatif à la liste des sites visités par le navigateur
- location est relatif à l'adresse URL du site visité
- layer est relatif aux couches du document
- links porte sur les liens hypertexte d'un document
- anchors porte sur les ancres (signets) d'un document
- images porte sur les images d'un document
- forms porte sur les formulaires d'un document
- etc...

Passons en revue les principales classes.

La pseudo-classe **window** concerne une zône d'affichage

| propriétés | |
|---|---|
| defaultStatus | message par défaut de la barre d'état |
| frames | tableau des divisions de la fenêtre |
| length | nombre de divisions |
| name | nom de la fenêtre |
| parent | fenêtre parent |
| self | fenêtre active |
| status | message de la barre d'état |
| top | fenêtre de premier plan |

| méthodes | |
|---|---|
| alert() | création d'une fenêtre de dialogue |
| open() | ouverture d'une fenêtre |
| close() | fermeture d'une fenêtre |
| clear() | effacement du contenu d'une fenêtre |
| prompt() | affichage d'une boite de dialogue avec un texte |
| setTimeout() | évaluation d'une expression ou appel d'une fonction après l'écoulement d'un délai fixé |

La pseudo-classe **document** porte sur le ou les contenus d'une fenêtre :

| propriétés | |
|---|---|
| anchors | accès aux ancres |
| forms | accès aux formulaires |
| lastModified | date de dernière modification |
| links | accès aux liens hypertextes |
| location | URL du document |
| title | titre du document en cours |

| méthodes | |
|---|---|
| clear() | effacement du contenu du document |
| open() | ouverture du buffer du document |
| close() | fermeture du buffer du document |
| write() | écriture dans le document |
| writeln() | idem avec retour à la ligne |

La pseudo-classe **history** mémorise la succession des sites visités par leur URL :

| propriété | |
|---|---|
| length | nombre d'URL chargées |

| méthodes | |
|---|---|
| back() | précédent |
| forward() | suivant |
| go() | aller à |

La pseudo-classe **location** ne possède que deux propriétés :

| propriétés |
|---|
| |

| | |
|---|---|
| host | URL du document |
| href | URL complet |



L'exemple 31 indique comment ces diverses pseudo-classes sont utilisées. Il porte sur l'analyse des liens contenus dans une page Web.

# Programmation événementielle

### Evénements

Un clic souris, une saisie de données, un chargement de page, le passage de la souris sur une zone,... constituent des événements. Ces événements peuvent provoquer des actions à l'aide de scripts JavaScript.

Les événements reconnus par JavaScript sont définis dans le tableau ci-dessous:

| description | événement |
|---|---|
| désélection de la cible | onBlur |
| click gauche de la souris | onClick |
| modification du contenu de la cible | onChange |
| activation de la cible | onFocus |
| chargement d'une page HTML | onLoad |
| passage de la souris sur la cible | onMouseOver |
| sélection de la cible | onSelect |
| soumission de formulaire | onSubmit |
| sortie de page HTML | onUnload |
| annulation du chargement d'une image | onAbort |
| glissement d'un objet vers la fenêtre du navigateur | onDragDrop |
| chargement causant une erreur | onError |
| frappe d'une touche clavier | onKeyDown |
| appui sur une touche clavier | onKeyPress |
| relâchement d'une touche clavier | onKeyUp |
| appui sur le bouton souris | onMouseDown |
| déplacement du curseur de la souris | onMouseMove |
| sortie du curseur de la souris de la zone sensible | onMouseOut |
| relâchement du bouton de la souris | onMouseUp |
| déplacement d'une fenêtre | onMove |
| reset d'un formulaire | onReset |
| changement de taille d'une fenêtre | onResize |

Pour créer un gestionnaire d'événement, il suffit d'utiliser une balise et d'ajouter le mot clé de l'événement avec un code JavaScript indiquant l'action à entreprendre si l'événement se produit

<BALISE onQuelquechose="code javascript">

exemple : <INPUT type="button" value="calcul" onClick="calculer(this.form);">

exemple41 : commentaires d'une carte ; lorsque la souris passe sur une zone, une fenêtre d'explication s'affiche

## Cookies

Le protocole HTTP ne reconnaît pas deux chargements consécutifs de la même page : ces deux chargements sont considérés comme totalement indépendants.Si l'on veut garder entre deux chargements consécutifs (ou plus) des valeurs de variables, on peut cependant utiliser la technique des cookies.

Les cookies sont des informations qui sont écrites sur le poste client, sur le disque dur et donc mémorisées en vue d'une utilisation ultérieure lors d'une connexion future. Le fichier qui contient les cookies est usuellement appelé cookies.txt (c'est un fichier texte) qui se trouve, soit dans le répertoire Netscape, soit dans le sous-répertoire cookies du répertoire Windows (il porte un autre nom de la forme nom_utilisateur@javascript.txt. Attention ! chaque navigateur a son emplacement pour le fichier relatif aux cookies ; si vous changez de navigateur, vous risquez des incohérences relativement aux cookies stockés. L'écriture et la lecture dans ce fichier peut être effectuée notamment par un petit programme JavaScript.

La structure d'un cookie est la suivante :

nom=valeur [; expires=date][; domain=nom_domaine][; path=chemin][;secure]

nom est une variable et valeur est sa valeur instantanée
expires=date exprime la date de validité du cookie ; par défaut, le cookie expire à la fin de la session.
domaine est le nom du domaine de validité du cookie (par défaut nom du serveur)
path définit le chemin du cookie
secure définit si le cookie est transmis de façon sécurisée.

- écriture d'un cookie : elle peut se faire à l'aide de la fonction setcookie(nom,valeur,expiration)

```
function setcookie(nom, valeur, expiration) {
                document.cookie=nom+"="+escape(valeur)+
                ((expiration==null)?"":(";expires="+expiration.toGMTString()))
            }
```

- lecture d'un cookie : elle peut s'effectuer à l'aide de la fonction getcookie(nom)

```
 function getcookie(nom){
                var recherche=nom+"=";
                if (document.cookie.length > 0) {
                      offset=document.cookie.indexOf(recherche);
                      if(offset != -1) {
                           offset=+=recherche.length;
                           fin=document.cookie.indexOf(";",offset)
                           if (fin== -1) fin=document.cookie.length;
                           return unescape(document.cookie.substring(offset,
  fin));
                                                              }
```

```
}
```



voir exemple 42 : enregistrement d'un abonné

# Exercices en JavaScript

**ex1) Changement de la couleur de l'arrière plan**

Ecrire un formulaire du type menu déroulant permettant de choisir la couleur de l'arrière plan à volonté.

**Solution**

---

**ex2) Horloge digitale**

Réaliser une horloge digitale donnant le jour et l'heure

**Solution**

---

**ex3) Ouverture de fenêtres** :

Donner la possibilité d'ouvrir de nouvelles fenêtres sans accessoires, avec barre d'état, avec menu simple, avec menu complet.

**Solution**

---

**ex4) Manipulation d'images**

En utilisant les images gif suivantes (arret.gif et marche.gif) et deux boutons, afficher successivement l'une des deux images.

**Solution**

---

**ex5) Affichage d'un message**

Afficher un message dans la barre d'état quand le curseur de la souris n'est pas sur une zone sensible et un message dans une boite de message quand on clique sur la zone sensible.

**Solution**

---

**ex6) Echange de contenus**

Echanger les contenus de deux menus

**Solution**

---

**ex7)Changement de langue**

Réaliser un menu déroulant présentant les jours de la semaine et un autre menu déroulant proposant deux langues : Français et Englis. Suivant le choix, les jours devront s'afficher dans la langue choisie.

**Solution**

# Solution des exercices

**ex1) Changement de la couleur de l'arrière plan**

***Code :***

```
<FORM>
<SELECT onChange="document.bgColor=this.options[this.selectedIndex].value">
<OPTION VALUE="40E0D0"> Turquoise
<OPTION VALUE="2E8B57"> Vert comme la mer
<OPTION VALUE="87CEEB"> Bleu comme le ciel
<OPTION VALUE="F4A460"> Brun comme le sable
<OPTION VALUE="FFF0F5"> Bleu lavande
<OPTION VALUE="FF1493"> Rose
<OPTION VALUE="FFFFFF" SELECTED> Blanc
</SELECT>
</FORM>
```

**Notes :**

la balise SELECT indique que l'on utilise un menu déroulant ; les balises OPTION indiquent les options possibles.

L'événement onChange permet de changer la sélection.

document.bgColor est la couleur de fond de l'objet document

value est la valeur de l'option (ici c'est une couleur codée).



options[this..selectedIndex] est l'option sélectionnée par l'utilisateur

On notera qu'au départ c'est l'option "Blanc" qui est sélectionnée par défaut.

**Réalisation**

**Retour à l'énoncé**

---

**ex2) Horloge digitale**

Code :

```
<TABLE BORDER=4 BGCOLOR="cyan">
<TR><TD>
<FORM NAME="clock_form">
<INPUT TYPE=TEXT NAME="cadran" SIZE=26>
</FORM>



<SCRIPT LANGUAGE="JavaScript">
function TicTac(){
Date_courante = new Date();
document.clock_form.cadran.value = " "+Date_courante;
document.clock_form.cadran.blur();
setTimeout("TicTac()", 1000);
}
TicTac();
// fin de l'horloge -->
</SCRIPT>
</TD></TR>
</TABLE>
```

**Notes :**

On utilise ici un formulaire qui présente une zone d'affichage de texte, dont le nom est "cadran".

Dans le script, la fonction TicTac() est définie. Elle crée un objet de type Date : Date_courante.

document.clock_form.cadran.value est la valeur affichée dans la zone cadran du formulaire clock_form du document web. La zone cadran est ensuite désactivée avec la fonction blur().

SetTimeOut est un minuteur qui déclanche la fonction TicTac() au bout de 1000 ms. On notera la récursivité puisque la fonction TicTac() s'appelle elle-même.

**Réalisation** :

**Retour à l'énoncé**

---

**ex3) Ouverture de fenêtres** :

**Code :**

```
<P>page simple :
<A HREF="#" onClick="window.open('exjs1.htm', 'fen2', 'width=500,
height=400')">Cliquez ici</A>
</P>
<P>page avec barre d'état :
<A HREF="#" onClick="window.open('exjs1.htm', 'fen1', 'width=370,height=240,
status=1')">Cliquez ici</A>
</P>
<P>page avec menu simple :
<A HREF="#" onClick="window.open('exjs1.htm', 'fen3', 'width=500,height=400,
menubar=yes,location=yes,scrollbars=yes')">Cliquez ici</A>
</P>
<P>page avec menu complet :
<A HREF="#" onClick="window.open('exjs1.htm', 'fen4', 'width=500,height=400,
toolbar=1,status=1,scrollbars=1,resizable=1')">
Cliquez ici</A>
</P>
```

**Notes :**

Paramètres d'une fenetre :

- Les deux premiers paramètres sont l'URL du document à charger dans la fenêtre et le deuxième paramètre est le nom de la fenêtre.
- Les autres paramètres sont des attributs de la fenêtre :

height : Hauteur de la fenêtre en pixels (ex: height=100)

width : Largeur de la fenêtre en pixels (ex: width=200)

directories : Spécifie s'il faut afficher les boutons de répertoire du navigateur comme "What's Cool" etc. (ex: directories=yes)

hotkeys : Autorise ou inhibe les touches de fonction (ex: hotkeys=no)

location : Spécifie si la boîte "Location" doit être affichée (ex: location=yes)

menubar : Specifie si la fenêtre doit avoir une barre de menu (ex: menubar=yes)

resizable : Specifie si l'utilisateur peut redimensionner la fenêtre (ex: resizable=no)

scrollbars : Specifie si la fenêtre peut être muni d'une barre de défilement (ascenseur) (ex: scrollbars=yes)

toolbar : Specifie si la fenêtre doit avoir une barre d'outils (ex: toolbar=yes)

**Réalisation**

**Retour à l'énoncé**

---

**ex4) Manipulation d'images**

**Code :**

```
<FORM>
<IMG NAME="ventilateur" SRC="arret.gif" WIDTH=61 HEIGHT=72>
<BR><BR>
<INPUT TYPE=BUTTON VALUE=" Arrêt " onClick="ventilateur.src = 'arret.gif'">
<INPUT TYPE=BUTTON VALUE=" Marche " onClick="ventilateur.src = 'marche.gif'">
</FORM>
```

**Notes :**

Une image est créée avec la balise IMG ; son nom est "ventilateur" ; sa source est donnée par l'adresse URL "arret.gif".

Deux boutons sont créés avec une action déclanchée par les événements onClick ; cette action consiste simplement à changer la source, c'est à dire l'adresse, de l'image affichée.

**Réalisation**

**Retour à l'énoncé**

---

**ex5) Texte défilant dans la barre d'état**

**Code :**

```
<a href="exjs5.htm" onMouseOver = "window.status='Cliquez ici pour en savoir
plus...';" onMouseOut = "window.status='';" onClick="alert('petit curieux ! ')" ;
>Cliquez ici</a>
```

**Notes :**

Les balises <A> et </A> indiquent un lien sur la phrase "Cliquez ici". Ce lien est, en fait inactif puisqu'on réaffiche la même page (supposée être exjs5.htm) ; on pourrait aussi écrire href="#" ;

L'événement onMouseOver provoque l'écriture dans la barre d'état d'un texte quand la souris passe sur le lien.

L'événement onMouseOut provoque l'effacement du message de la barre d'état quand la souris est en dehors du lien.

L'événement onClick provoque l'affichage d'un message d'alerte.

**Réalisation**

**Retour à l'énoncé**

---

**ex6) Echange d'informations**

**Code :**

```
<html><head><title> transfert </title>
<script language="JavaScript">

function SversD() {
  indexS=document.trans.source.options.selectedIndex;
  if (indexS < 0) return;
  valeur=document.trans.source.options[indexS].text;
  document.trans.source.options[indexS]=null;
  a = new Option(valeur);
  indexD=document.trans.destination.options.length;
  document.trans.destination.options[indexD]=a;
}

function DversS(){
  indexD=document.trans.destination.options.selectedIndex;
  if (indexD < 0)return;
```

```
  valeur=document.trans.destination.options[indexD].text;
  document.trans.destination.options[indexD]=null;
  a = new Option(valeur);
  indexS=document.trans.source.options.length;
  document.trans.source.options[indexS]=a;
}

</script>
</head>
<body>
<p>
<form name=trans>
<table>
<tr>
<td>
  <select size=7 width=100 name=source>
    <option>Citrons
    <option>Oranges
    <option>Pamplemousses
    <option>Clementine
    <option>Kiwis
    <option>Bergamotes
    <option>Grenades
  </select>
<td>
  <input type=button value=">>" onClick=SversD()><br>
  <input type=button value="<<" onClick=DversS()>
<td>
  <select size=7 width=150 name=destination>
  </select>
</tr>
</table>
</form>
</body>
</html>
```

**Notes :**

Le formulaire comporte un premier menu (rempli initialement et appelé source), deux boutons, un second menu (vide initialement et appelé destination).

Les fonctions SversD() et DversS()effectuent les transferts ; elles sont évidemment symétriques. Il suffit d'étudier l'une d'elles, par exemple SversD().

Dans la fonction SversD, on définit d'abord indexS qui est l'index sélectionné dans la liste des options du menu source du formulaire trans du document (ouf !). Valeur est le contenu pointé par cet index. On fait disparaître ce contenu (valeur null). On crée une nouvelle option contenant la valeur sélectionnée. On créé dans le second menu Destination un index supplémentaire (en utilisant la longueur actuelle de la chaîne d'index)et on affecte à cet index la valeur sélectionnée.

**Réalisation**

**Retour à l'énoncé**

---

**ex7) Conversion de langue**

**Code :**

```
<html>
<head><title>test select</title>
<script language="JavaScript">
function chLangue(forme) {
  if (forme.langue.options[1].selected == true) {
    forme.jours.options[0].text = "Monday"
    forme.jours.options[1].text = "Tuesday"
    forme.jours.options[2].text = "Wenesday"
    forme.jours.options[3].text = "Thursday"
    forme.jours.options[4].text = "Friday"
    forme.jours.options[5].text = "Saturday"
    forme.jours.options[6].text = "Sunday"
    }
  else {
    forme.jours.options[0].text = "Lundi"
    forme.jours.options[1].text = "Mardi"
    forme.jours.options[2].text = "Mercredi"
    forme.jours.options[3].text = "Jeudi"
    forme.jours.options[4].text = "Vendredi"
    forme.jours.options[5].text = "Samedi"
    forme.jours.options[6].text = "Dimanche"
    }
  }
</script>
</head>
<body>
<form><font size=5>
  <select name="jours">
    <option>Lundi
    <option>Mardi
    <option>Mercredi
    <option>Jeudi
    <option>Vendredi
    <option>Samedi
    <option>Dimanche
  </select>
  <p>
  <select name="langue" onChange="chLangue(this.form)">
    <option selected>Francais
    <option>English
  </select>
</form>
</body>
</html>
```

**Notes :**

Le formulaire comprend deux menus déroulant, l'un pour afficher les jours, l'autre pour afficher les langues. Suivant le choix de langue, la fonction chLangue() remplit les options du menu déroulant des jours.

***Réalisation***

***Retour à l'énoncé***

# Introduction aux
# Active Server Pages

## Introduction

Les **ASP** (Active Server Pages) sont un standard Microsoft permettant de développer des applications Web interactives. Ils permettent d'exécuter des scripts côté serveur. Une page web ASP (dont l'extension est .asp) aura donc un contenu dynamique, pouvant être différent selon certains paramètres (des informations stockées dans une base de données, les préférences de l'utilisateur,...) alors que la page web "classique" (dont l'extension est .htm ou .html) affichera continuellement la même information.

Il s'agit en réalité d'un langage de script, interprété, et exécuté du côté du serveur et non du côté client (les scripts écrits en Javascript ou les applets Java s'exécutent dans le navigateur du poste client). Le serveur retourne la réponse en format HTML.

A l'origine, les ASP ont été conçues pour fonctionner sur le serveur Web de Microsoft intitulé **Microsoft IIS** (Internet Information Server). Ce serveur web, mis au point par Microsoft en 1996 fonctionne sous Microsoft Windows NT.

Aujourd'hui, il parait que cette technologie est disponible sur d'autres serveurs web que celui de Microsoft. il est supporté par le serveur Netscape FastTrack utilisant le module Chili!Software, puis sur d'autres serveurs dont Apache, avec le module **Apache::ASP**, je n'ai pas encore eu l'occasion de le vérifier. En tous cas, c'est cela qui rendra possible la création de sites Web utilisant la technologie des ASP sur de nombreuses plate-formes (Unix,Linux,PowerPC,...).

Les ASP sont intégrables au sein d'une page Web en HTML, à l'aide de balises spéciales **<%.....%>**.
Le code compris à l'intérieur de ces balises est interprété par le serveur, le résultat (en code HTML) est renvoyé au navigateur du client.

**Tous les exemples de ce cours peuvent être copiés sur wordPad ou bloc note, enregistrés en format .asp, type "texte seulement " et exécutés sur le serveur personnel PWS**

**Exemple 1**

```
<%@language="VBscript"%>

<HTML>
<head><title>exemple d'une page asp</title></
head>

<body>

<% for x=1 to 8 %>

<HR width=20% size=<%=x%> color=red
align=left>
la taille de cette ligne est <%=x%>
pixels<br><br>

<%next%>

</body>
</html>
```

**Résultat**

- **<%@language="VBscript"%>** permet d'indiquer au serveur dans quel langage sont écrits les scripts asp contenus dans la page. On pourrait utiliser JavaScript, Jscript, Perl, Java, ...

- L'ASP utilise les délimiteurs **<%** et **%>** qui permettent au serveur de repérer les codes et les exécuter avant de restituer un code HTML au navigateur demandeur.
- le langage VBscript possède tout un jeu d'instructions, issues de Visual Basic dont **for et next** qui permettent de répéter un groupe d'instructions sur un certain nombre de fois données
- la commande **<%=x%>** est utilisée pour changer la taille de la ligne et afficher à l'écran la valeur de cette variable x.

Utiliser uniquement les instructions d'un langage comme VBscript c'est un peu limité...

Nous aborderons par la suite les objets ASP qui permettent d'envoyer des informations vers d'autres pages ou accéder à une base de données en utilisant technologie **ADO** (ActiveX Data Object). Celle-ci fournit les éléments nécessaires à la connexion au système de gestion de bases de données, et à la manipulation des données en utilisant un langage comme SQL.

## I - Les objets de base des Active Server Pages

Dans sa version 3.0, l'ASP est architecturé autour de 6 objets internes, qui comprennent des méthodes et des propriétés, permettant d'effectuer les principaux traitements sur les données en constituant le moteur des scripts.

Active Server Pages 3.0 dispose de six objets intégrés :

* **Request** qui permet de traiter les informations en provenance du client par l'intermédiaire de formulaires. Il permet de récupérer les valeurs des champs de requête issus du formulaire du

navigateur.

* **Response** qui représente le résultat à afficher sur le navigateur.

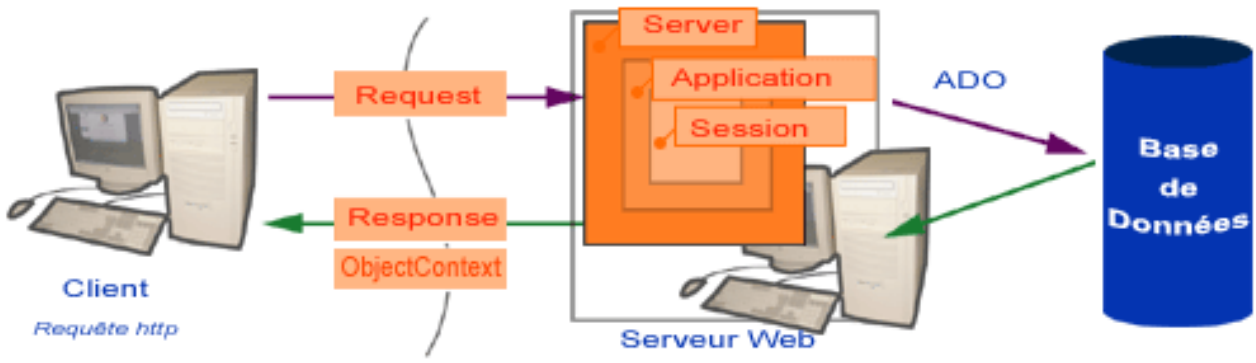
* L'objet **Server** contient les informations concernant l'état du serveur et fournissant des méthodes pouvant être utilisées dans les scripts.

* **Session** qui représente l'utilisateur. Il permet de conserver les données relatives à l'utilisateur d'une page d'un site à un autre

*L'objet **Application** sert à stocker les informations utilisées lors de l'exécution des scripts.

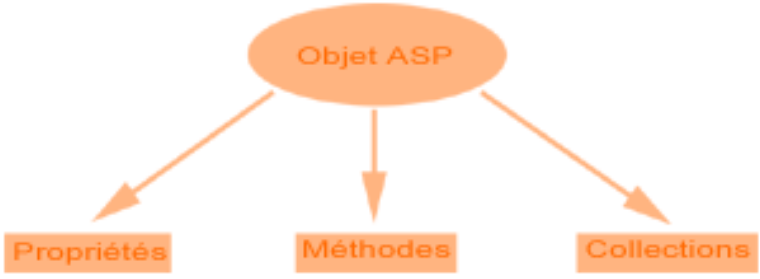
* **ObjectContext** est un objet plus spécifique, utilisé lors de transactions gérées par le logiciel MTS (Microsoft Transaction Server).

Ces objets possèdent des éléments de méthodes, évènements et propriétés typiquement orientés objets.




## II. La structure d'un objet ASP

Les objets ASP constituent l'essentiel du moteur des scripts ASP. Ce sont les principaux éléments regroupant les propriétés (valeurs) et les méthodes (traitements) utilisables dans les scripts.




En réalité un objet est composé de trois types d'entités:

· **Les propriétés**: ce sont des valeurs spécifiques directement accessibles. On accède à une propriété d'un objet ASP par la syntaxe:

**objet.propriete**

· **Les collections**: groupe de propriétés. Ce sont des structures de données (une sorte de tableau) contenant un ensemble de valeurs repérées par une clé. Chaque objet peut donc contenir plusieurs collections de variables. Une valeur d'une collection d'un objet est accessible par la syntaxe suivante:

**objet.colllection("clé")**

· **Les méthodes**: ce sont des fonctions standards associées à un objet, permettant de manipuler des valeurs passées en argument. La syntaxe d'une méthode ressemble donc à ceci:

**objet.methode(arguments)**

## II. 1 Présentation de l'objet Request

Le rôle de l'objet **Request** est de permettre de récupérer la réponse HTTP envoyée par le navigateur au serveur web, il possède une seule méthode et propriété et de nombreuses collections :

| Objet Request | | |
|---|---|---|
| **Collections** | **Methode** | **Propriété** |
| QueryString | | |
| Form | | |
| Cookies | | |
| ServerVariables | | |
| ClientCertificates | TotalBytes | BinaryRead |

La majorité des propriétés et des méthodes de l'objet **Request** correspondent à des fonctions ou propriétés permettant de manipuler les champs de la requête HTTP.
Manipulons ces exemples pour comprendre ces collections.

### II. 1 a) utilisation de la collection QueryString

La collection **QueryString** permet de récupérer la valeur associée à un champ par la syntaxe suivante :

**<% Request.QueryString("Champ") %>**

Ce champ peut être lié à un lien ou un champ d'un formulaire

**Exemple 2 :**

Transmission des variables à une autre page web par un lien hypertexte

En cliquant sur ce lien : ceci est un exemple de lien dynamique, on transmet à la page asp exemple2.

asp quatre variables

- la variable nom avec pour contenu **Machin**
- la variable prenom avec pour contenu **bidule**
- la variable age avec pour contenu **25**
- la variable adresse avec pour contenu **Amiens**

**code1 : le passage de variables dans le lien hypertexte : page nommée, par exemple, exemple2a.html**

```
<HTML>
<head><title>exemple 2a</title></head>

<body>

<A href="exemple2B.asp?
nom=Machin&prenom=Bidule&age=25&
adresse=Amiens> ceci est un exemple de lien dynamique</
A>

</body>
</html>
```

la balise **<A href="..".>** permet de créer un lien hypertexte

**code2 : cette page doit être nommée *exemple2B.asp*, elle contient le script de récupération des variables**

```
<%@language="VBscript"%>

<HTML>
<head><title>exemple2B</title></head>

<body>
<br><br><br><br>

<p>La variable <b>"nom"</b> contient <%=Request.QueryString
("nom")%> </p>
<p>La variable <b>"prenom"</b> contient <%=Request.QueryString
("prenom") %> </p>
<p>La variable <b>"age"</b> contient <%=Request.QueryString("age")
%> </p>
<p>La variable <b>"adresse"</b> contient <%=Request.QueryString
("adresse") %> </p>

</body>
</html>
```

**Exécuter ces deux pages sur le serveur PWS.**

## II. 1 b) utilisation de la collection Form

### La réception de données

Pour comprendre comment utiliser l'objet **Request**, il est nécessaire de connaître la manière par laquelle les données sont envoyées au navigateur. Pour cela, il faut se référer au code HTML lié à un "formulaire".

Les formulaires HTML se créent à l'aide de la balise **<FORM>** contenant des boutons, des champs, des listes, des cases à cocher,...ainsi qu'un bouton de soumission du formulaire qui envoie l'ensemble des informations au script indiqué en tant qu'attribut **Action** de la balise **FORM** selon la méthode **GET** ou **POST**. Chaque élément du formulaire doit posséder un nom unique et une valeur (le contenu du champ).

L'envoi des données se fera différemment suivant la méthode utilisée pour l'envoi du formulaire : **GET** ou **POST**.

· **la méthode GET** permet d'envoyer les éléments au travers de l'URL du script, en ajoutant l'ensemble des paires nom/valeur à cette URL, séparé de celui-ci par un point d'interrogation comme dans l'exemple 2. ou l'exemple suivant :

http://nom_du_serveur/Dfoad/Formation.asp?champ1=valeur1&champ2=valeur2

Toutefois, la longueur de la chaîne URL étant limitée à 255 caractères, les informations situées au-delà de cette limite seront perdues. De plus, cela crée une URL surchargée dans la barre d'adresse d'un navigateur et peut dévoiler des informations sensibles comme un mot de passe, un login, numéro de CB, ...

· **la méthode POST** est une bonne alternative à la méthode GET. Cette méthode code les informations de la même façon que la méthode GET (encodage URL et paires nom/valeur) mais elle envoie les données à la suite des en-têtes HTTP, dans un champ appelé corps de la requête. De cette façon la quantité de données envoyées n'est pus limitée, et est connue du serveur grâce à l'en-tête permettant de connaître la taille du corps de la requête.

***Exemple 3 :***
**utilisation d'un formulaire pour transmettre des variables vers une autre page :**

La collection **QueryString** permet de récupérer de façon simple les données envoyées au script ASP par l'intermédiaire de l'URL (c'est-à-dire par la méthode GET) voir exemple2.

La collection **Form** permet de manipuler les données envoyées par un formulaire en utilisant la

méthode **POST**. La récupération de la valeur associée à un champ s'effectue par la syntaxe suivante :

```
<% Request.Form("Champ") %>
```

**Passer des variables à une autre page web par un formulaire**

**a) Saisie d'un formulaire dans une page html : formulaire.htm**

```
<HTML>

<HEAD><TITLE>Exemple 3 d'une page asp</TITLE></HEAD>
<BODY>

<br><Br><br><Br>

<FORM METHOD="POST" Action = "exemple3.asp">

<P> Nom : <INPUT TYPE ="TEXT" NAME="name"></P>

<P> Prénom : <INPUT TYPE ="TEXT" NAME="surname"></P>

<P> Age : <INPUT TYPE ="TEXT" NAME="age"></P>

<P> Adresse : <INPUT TYPE ="TEXT" NAME="address"></P>

<P Align="CENTER"> <INPUT value="Envoyer" TYPE ="SUBMIT"></P>

</FORM></BODY></HTML>
```

Listing de la page asp nommée **exemple3.asp**, permettant d'afficher les données saisies dans le formulaire précédent

```
<%@ LANGUAGE="VBScript" %>

<HTML>
<HEAD><TITLE>Script de page exemple3.asp</TITLE></HEAD>

<BODY>

<P>Votre nom est : <%=Request.form("name" )%></P>

<P>Votre prénom est : <%=Request.form("surname" )%> </P>

<P>Votre age est : <%=Request.form("age" )%> </P>

<P>Votre adresse est : <%=Request.form("address" )%> </P>

</BODY></HTML>
```

**Essayons cet exemple**

## La collection ServerVariables

La collection **ServerVariables** de l'objet Request contient les en-têtes de la requête http. Elle permet de récupérer des informations plus ou moins utiles sur les visiteurs ou le navigateur du client.

La syntaxe : **Request.ServerVariables("NOM-EN-TETE")**

les principaux en-têtes utiles:

| Nom de l'en-tête | Description |
|---|---|
| CONTENT_TYPE | Type de contenu du corps de la requête (par exemple text/html). |
| METHOD | Type de méthode utilisée par le client (ie POST ou GET) |
| REFERER | URL du lien à partir duquel la requête a été effectuée |
| REMOTE_ADDR | Adresse IP du client |
| HTTP_ACCEPT_LANGUAGE | Langage attendu par le browser (anglais par défaut) |
| HTTP_USER_AGENT | Chaîne donnant des informations sur le client, comme le nom et la version du navigateur, du système d'exploitation |

**Exemple 4a : page ASP permettant de connaître le type de navigateur utilisé**

```
<%@ LANGUAGE="VBScript" %>

<HTML>
<HEAD><TITLE>Script de page exemple4a.asp</TITLE></
HEAD>

<BODY>

 <%variable_navig=Request.ServerVariables
("HTTP_USER_AGENT" )%>

<br><br><br><br>

<P>vous utilisez le navigateur :  <%=variable_navig%> </P>

</BODY></HTML>
```

Résultat

**Exemple 4b** : Récupération de la langue du navigateur

```
<%@ LANGUAGE="VBScript" %>

<HTML>
<HEAD><TITLE>Script de page exemple4a.asp</TITLE></HEAD>

<BODY>

 <%variable_lang=left(Request.ServerVariables
("HTTP_ACCEPT_LANGUAGE"),2)%>

<br><br><br><br>

<P>Votre navigateur utilise la langue : <%=variable_lang%> </P>

</BODY>
</HTML>
```

**Résultat**

La fonction **left** permet d'extraire seulement les deux premiers caractères du code langue, "fr" pour le français et "es" pour l'espagnole, ...

**Exemple 4c : Adresse IP**

Le code permettant de stocker dans une variable l'adresse IP d'un visiteur, et l'afficher:

```
<%@ LANGUAGE="VBScript" %>
<HTML>
<HEAD><TITLE>Script de page exemple4c.asp</
TITLE></HEAD>

<BODY>

  <%variable_IP=Request.ServerVariables
("REMOTE_ADDR")%>

<br><br><br><br>

<P>Votre adresse IP : <%=variable_IP%> </P>

</BODY></HTML>
```

**Résultat**

Pour compter les visiteurs d'un site par exemple, il peut être intéressant de stocker l'adresse IP de ces derniers et de compter le nombre d'adresses IP stockées et différentes chaque jour !.

## II. 2 Présentation de l'objet Response

L''objet **Response** permet de créer la réponse HTTP et de manipuler l'ensemble des informations à destination du navigateur du client.

### a) L'envoi de données au navigateur

Pour envoyer du texte au navigateur dans un code ASP, il suffit d'utiliser la propriété **write** de l'objet **Response**,

Nous avons en fait utiliser déjà cette propriété d'une façon contractée. En effet, lorsque nous avons écrit dans les scripts d'exemple 4 : **<%=variable_navig%>, <%=variable_lang%>** ou**<%=variable_IP%>** pour afficher le contenu de ces variables sur le poste client, le signe **=** est une abréviation de la méthode write de l'objet Response. on aurait peu écrire :

| | |
|---|---|
| **<%=variable_navig%>** | **<%Response.write "variable_navig"%>** |
| **<%=variable_lang%>** | **<%Response.write "variable_lang"%>** |
| **<%=variable_IP%>** | **<%Response.write "variable_IP"%>** |

### b) Les constituants de l'objet Response

L'objet Response possède une seule collection, mais aussi de nombreuses propriétés et méthodes:

| **Objet Response** | | |
|---|---|---|
| **Collections** | **Méthodes** | **Propriétés** |
| Cookies | Buffer | AddHeader |
| | CacheControl | AppendToLog |
| | CharSet | BinaryWrite |
| | ContentType | Clear |
| | Expires | End |
| | ExpiresAbsolute | Flush |
| | isClientConnected | Redirect |
| | Status | Write |
| | PICS | |

La majorité des propriétés et des méthodes de l'objet Response correspondent à des champs de la réponse HTTP.

### c) La collection Cookies

#### i) L'envoi de cookies au navigateur

Le protocole HTTP ne reconnaît pas deux chargements consécutifs de la même page : ces deux chargements sont considérés comme totalement indépendants. Si l'on veut garder entre deux chargements consécutifs (ou plus) des valeurs de variables, on peut cependant utiliser la technique des cookies.

Les cookies sont des informations qui sont écrites sur le disque dur du poste client et donc mémorisées en vue d'une utilisation ultérieure lors d'une connexion future vers le serveur. Ces

informations ne peuvent être lues que par le site web qui les a disposées. Le fichier qui contient les cookies est usuellement appelé cookies.txt (c'est un fichier texte) qui se trouve, soit dans le répertoire Netscape, soit dans le sous-répertoire cookies du répertoire Windows (il porte un autre nom de la forme nom_utilisateur@domaine.txt.

En dehors de la possibilité d'envoyer une page Web au navigateur du client, l'objet **Response**, permet aussi de lui envoyer des **cookies**.

Le protocole HTTP permet de spécifier les valeurs de cookies dans les en-têtes http. L'envoi de cookies au navigateur dans la page ASP doit donc se faire avant tout envoi dans le corps de la réponse.

Pour stocker une valeur dans un cookie appelé nom_Cookie, il suffit d'utiliser la commande suivante:

```
Response.cookies("nom_Cookie") = "Valeur"
```

Ce cookie s'effacera dès que l'internaute aura fermé son navigateur. Si on souhaite garder ces informations pour les futures visites, il faut employer une propriété de cookie permettant de définir sa durée de vie, on écrit donc :

```
Response.cookies("nom_Cookie").expires = Mois/Jour/Année heure:minute:seconde
```

Ce cookie s'effacera le Mois/Jour/Année (format anglais) heure:minute:seconde défini par le serveur (plutôt le développeur de l'application).

Pour être sûr que le cookie ne sera lu que par le serveur qui l'a créé, on peut utiliser la propriété suivante :

```
Response.cookies("nom_Cookie").domain = "/www.mosite.com/"
```

Nous pouvons aussi écrire :

```
Response.cookies("nom_Cookie").secure = True
```

ce qui permet de lire le cookie avec une connexion SSl. cette connexion est codée et sécurisée afin de protéger les informations contenues dans ce cookie. Cette technique est utilisée par les entreprises, site de commerce électronique et les banques.

Pour stocker plusieurs valeurs associées à des index dans un cookie appelé **nom_Cookie**, il suffit d'utiliser la commande suivante:

```
Response.cookies("nom_Cookie")("Index1") = "Valeur1"

Response.cookies("nom_Cookie")("Index2") = "Valeur2"

<Response.cookies("nom_Cookie")("Index3") = "Valeur3"
```

**ii) Lecture de cookies**

L'accès aux données d'un cookie fait appel à l'objet asp Request qui se manifeste de la manière suivante:

**Variable = Request.Cookies("Nom_Cookie")**

Pour lire les différentes variables associées à des index, on utilise les commandes :

**varaible1=Request.cookies("nom_Cookie")("Index1")**

**varaible2=Request.cookies("nom_Cookie")("Index2")**

**varaible3=Request.cookies("nom_Cookie")("Index3")**

Dans le prochain chapitre, nous étudierons un exemple pratique d'utilisation des cookies.

## d) Collection Redirect : Redirection vers une autre page

La fonction Redirect est très utilisée pour renvoyer automatiquement le visiteur d'une page vers une autre selon les critères prédéfinis.

exemple :

```
<%@ LANGUAGE="VBScript" %>

<% If left(Request.ServerVariables("HTTP_ACCEPT_LANGUAGE"),2)
="fr" Then
Response.Redirect("bienvenue.htm")

End IF

If left(Request.ServerVariables("HTTP_ACCEPT_LANGUAGE"),2)
="es" Then
Response.Redirect("bien_venido.htm")

End IF

If left(Request.ServerVariables("HTTP_ACCEPT_LANGUAGE"),2)
="en" Then
Response.Redirect("welcome.htm")

End IF %>

<HTML>
<HEAD><TITLE>Exemple : objet application</TITLE></HEAD><BODY>!!!!</
BODY></HTML>
```

**Résultat**

Dans cet exemple, le docuement HTML affiché dépend de la langue utilisé par le navigateur

## II. 3 Objet Session

Cet objet permet de stoker des informations mais contrairement au cookie, la durée de vie d'une session est limitée à une visite de site web et son stockage se fait sur le serveur et non sur la machine client. Les développeurs commencent à utiliser de plus en plus cette objet pour remplacer l'objet cookie puisque un certains nombre d'internautes configurent leur navigateur à refuser les variables cookies.

Une instance de l'objet session est souvent nommé Session . Une session spécifique étant attribuée à chaque utilisateur et identifiée de façon unique. Un numéro d'identification est nommé **SessionID** . Pour obtenir **SessionID** de la Session d'un utilisateur, on utilise la syntaxe : **Session. SessionID** ce qui donne une valeur numérique unique à chaque connexion au serveur.

Pour écrire une valeur dans une variable session, on utilise la syntaxe suivante : **Session (nom_variable_session)=valeur**

**Exemple** : **<% Session("nom") = "Michel" %>**

une variable de session peut être lue avec la syntaxe suivante : **<%=Session (nom_variable_session) %>**

**Exemple** : **Bienvenue <%= Session("nom") %>**

Une session se termine automatiquement si l'utilisateur n'a pas demandé ou actualisé la page d'une application au bout d'une durée déterminée. Cette valeur est de 20 minutes par défaut. on peut modifier la valeur par défaut d'une application en définissant la propriété **Session.Timeout**

**Exemple** : le script ci-dessous définit un délai d'expiration de 5 minutes.

**<% Session.Timeout = 5 %>**

## II. 4 Présentation de l'objet Application

L'objet Application permet de partager des informations entre plusieurs internautes.

### a) Collections

Il possède deux collections : **Contents** (propriété par défaut) et **StaticObjects**.

**Contents** permet de stocker en mémoire du serveur des variables globales à l'ensemble de l'application. Cela signifie que tous les internautes verront la même informations dans ces variables.

**exemple** : **Application.Contents("DataBaseName") = "DSN=mabase;"**
ou
**Application("DataBaseName") = "DSN=mabase;"**

(Pour avoir plus d'informations sur le DSN voir configuration de l'ODBC.)

Maintenant, dans toutes les autres pages de votre site et pour tous les internautes, vous pouvez utiliser cette variable :

**Response.Write Application("DataBaseName")**

Cela permet, par exemple, de stocker dans une variable globale le texte de connexion à une base de donnée et de ne pas à avoir à la réécrire dès que vous voulez accéder à celle-ci.

Toutefois, pour pouvoir modifier la valeur d'une variable globale, comme plusieurs internautes peuvent accéder et modifier cette information, il faut "locker" et "délocker" l'application avant de faire tous changements :

```
Application.Lock
Application("DataBaseName") = "DSN=mabase;"
Application.Unlock
```

L'objet Application possède deux évènements déclarées dans le fichier **Global.asa**, placé dans le répertoire racine du site Web :

**Application_OnStart** qui se déclenche dès que l'application démarre.
**Application_OnEnd** qui se déclenche dès que l'application s'arrête.

un autre exemple

```
<%@ LANGUAGE="VBScript" %>
<HTML>
<HEAD><TITLE>Exemple : objet application</TITLE></
HEAD>

<BODY>

<br><br><br><br>

<b>votre dernière connexion à cette page était le  <%
=Application("date_heure")%>

<%
Application.Lock
Application("date_heure") = now
Application.Unlock
%>

</BODY></HTML>
```

**Résultat**

## II. 5 Présentation de l'objet Server

L'objet Server vous permet d'en savoir plus sur votre serveur et ses propriétés. Ce qui permet de régler quelques paramètres relatifs à l'exécution des scripts.

Propriétés

**Server.ScriptTimeOut** définie le temps au bout duquel un script est abandonné. Ainsi au bout de 90 seconde, si l'exécution du script est trop longue, est boguée ou encore si un script tourne en boucle, le déroulement du script est arrêté et un message d'erreur est renvoyé au navigateur du visiteur.

Dans le cas d'un site web ayant un gros trafic et pour "économiser" les ressources du serveur, on affecte une valeur à l propriété **Server.ScriptTimeOut = 15** (par exemple). Ainsi l'exécution du script s'arrêtera au bout de 15 secondes en cas de problème.

Méthodes

**CreateObject** crée une instance d'un objet en lui passant son identificateur . Il faut bien entendu que l'objet, le composant soit installé sur le serveur.

exemple conn = CreateObject.

Nous étudierons cette exemple dans dans le prochain chapitre

**Execute** permet d'exécuter une autre page ASP ou une réquete SQL comme si l'on appelait une fonction

exemple : Server.Execute "recherche.asp" ou Server.Execute ( )

**GetLastError** retourne un objet ASPErrorObject qui contient la description de la dernière erreur générée.
**HTMLEncode** permet d'encode un texte en html.
**MapPath** retourne le chemin réel sur le serveur d'un chemin virtuel.
**Tranfert** permet de passer d'une page asp à une autre.

## II. 6 Présentation de l'objet ObjectContext

L'objet ObjectContext permet de manipuler les transactions avec un composant installé sur le serveur. Il possède seulement deux méthodes :

**SetAbort** indique qu'il faut arrêter la transaction.
**SetComplete** indique que la transaction peut être effectuée.

**Exemple :**

cet exemple a été récupéré de site http://www.c2i.fr

Imaginez un site de commande de marchandises. Vous allez dans votre page créer une instance de votre composant qui devra mettre à jour les stocks.

```
Dim objStock
Set objStock = Server.CreateObject("MonComposant.Stock")
```

Puis vous allez vérifier que la demande en quantité du client n'est pas supérieure au stock :

```
<%@ Transaction = Required%>
If objStock.MonStock("Patates")<Request.Form("DemandeClient") Then
    objStock.SetAbort
    Response.Write "Pas assez de stock"
Else
    objStock.SetComplete
    Response.Write "Vous serez bientôt livré "
End If
```

Dans cette même page, vous pouvez gérer les deux évènements liés que sont **OnTransactionAbort** et **OnTransactionCommit**. Le premier est déclenché quand SetAbort est appelé. Le second, quand la transaction a été effectuée. Pour récupérer ces évènements, il suffit de les déclarer dans votre page :

```
Sub OnTransactionAbort
...
End Sub

Sub OnTransactionCommit
...
End Sub
```

## Conclusion

Ce chapitre vous a présenté les objets ASP avec des exemples d'applications. Certains objets seront fréquemment utilisés dans le prochain chapitre.

# Technologies de Script

## La Programmation Web avec PHP

---

Objectifs

1. Qu'est-ce que php ?
2. Les scripts PHP
3. Installation de PHP
4. Configuration d'un serveur IIS

---

Mr Mohamed SIDIR

# 1 - Qu'est-ce que php ?

---

**PHP** est un langage de script HTML, compatible avec tous les systèmes d'exploitation ; Unix, linux, windows , Macintosh. L'essentiel de sa syntaxe est emprunté aux langages C, Java et Perl, mais y ajoute plusieurs fonctionnalités uniques. Le but de ce langage est de permettre aux développeurs web de concevoir des sites aux pages dynamiques.

La programmation avec php consiste à insérer des lignes de programmation dans la source d'un document html. Ces lignes représentent une succession de commandes qui doivent être interprétées par la machine qui exécute le script.

PHP est un langage de script qui fonctionne côté serveur, le client ne reçoit que le résultat du script, sans aucun moyen d'avoir accès au code qui a produit ce résultat.

**Exemple 0**

```
<html>

<head><title>Exemple 0</title></head>

<body>

<?php

echo "Bonjour, je ne suis qu'un simple exemple d'introduction!";

?>

</body>

</html>
```

**résultat_**

Comme le montre cet exemple, le code PHP est inclus à l'intérieur d'une page HTML, il permet de réaliser une action précise (dans ce cas là, la commande **echo** affiche la chaîne de caractère

entre guillemets : Bonjour, je ne suis qu'un simple exemple d'introduction!).

la règle du traitement par le serveur est la suivante :

Le code PHP est inclus entre une balise de début et une balise de fin, comme c'est le cas pour d'autres langage de scripts comme le javascript, ces deux balises permettent au navigateur de passer en "mode PHP". Chaque instruction PHP se termine par un ";" .

- Lorsque le programme d'analyse détecte la balise de début (**<?php** par exemple), il considère que les instructions qui suivent sont des scripts PHP et les traite comme telles, et ce, jusqu'à ce que la balise de fin soit rencontrée.
- Lorsque le programme d'analyse rencontre la balise de fin (**?>** par exemple), il considère que les instructions qui suivent ne sont plus des scripts PHP et les raites comme un code HTML, il se contente donc de les envoyer au navigateur client pour les afficher.

---

# 2 - Comment introduire les scripts PHP dans une page web ?

---

les scripts PHP sont généralement intégrés dans le code d'un document HTML même s'ils peuvent bien être complètement indépendants. Depuis la version 3 de PHP, plusieurs balises sont acceptées, le tableau suivant résume les différents types :

| | |
|---|---|
| **<?**<br>...<br>Script à exécuter<br>...<br>**?>** | style XML, il faut configuré le serveur pour accepter cette écriture. |
| **<SCRIPT LANGAGE="php">**<br>...<br>Script à exécuter<br>...<br>**</SCRIPT>** | si vous travaillez avec un éditeur HTML qui a tendance à essayer de remplacer, voire de supprimer les instructions qu'il ne connaît pas, sachez que le marquage de ce type est long à écrire mais conseillé puisque il indique à votre éditeur que le code contenu entre balises ne le regarde pas et qu'il ne doit pas y toucher.<br><br>Cadrez cette remarque en rouge si vous étés fan de FrontPage ! |
| **<%**<br>...<br>Script à exécuter style Asp<br>...<br>**%>** | pour les nostalgiques du code ASP, ce marquage est utilisé mais n'est pas reconnu par tous les serveurs web. |
| **<?php**<br>...<br>Script à exécuter<br>...<br>**?>** | Pour des raisons de commodité, il est conseillé d'utiliser cette solution qui est plus condensée et reste acceptée par l'ensemble des serveurs. |

Le même fichier peut comporter plusieurs blocs de code PHP exactement où on le désire, du moment que les balises de début et de fin sont bien utilisées pour chaque bloc. De plus, le code PHP peut intervenir n'importe où, même au beau milieu d'une ligne de code HTML, pour afficher une variable PHP par exemple.

**Exemple 1**

```
<html>

<head>

<title>Exemple 1</title></head>

<body>

<?php

$nom="Machin";

$prenom="bidule";

?>

nom : <b>  <?php  echo $nom ?> </b> <br>

Prénom : <b> <?php  echo $prenom ?></b>

 </body>

</html>
```

**Résultat_**

**Les fichiers externes :**

Lorsque du code doit être réutilisé de manière régulière dans plusieurs scripts différents, il vaut mieux écrire ce code dans un fichier externe et l'appeler dans le script principal lorsqu'on en a besoin, plutôt que de le réécrire dans chacun d'eux. Pour cela on utilise le mot clef « include » de la manière suivante :

Include " nom_du_fichier.inc "

## Exemple 2

```
<html>
<head>
<title>Exemple 1</title></head>
<body>
<?php
$var= " bonjour à ";
include "bonjour.inc";
?>
</body>
</html>
```

**Fichier inclus : bonjour.inc**

```
<?php
$chaine= $var.  " tout le monde ";
echo $chaine ;
?>
```

**Résultat**

Le langage PHP possède les même fonctionnalités que les autres langages permettant d'écrire des scripts CGI, comme collecter des données, générer dynamiquement des pages web ou bien envoyer et recevoir des cookies.

La plus grande qualité et le plus important avantage du langage PHP est le support d'un grand nombre de bases de données qui permet d'interfacer.

Les bases de données suivantes sont supportées par PHP

dBase, dbm, FilePro, Hyperware, Informix, Interbase, Sybase, mSQL, MySQL, Orale, PostgresSQL, Solid, Access, SqlServer, ...

## Commenter le code

Il est toujours intéressant de commenter un script. Si l'on veut insérer un commentaire au sein d'un programme PHP on utilise les deux écritures suivantes :

- Commentaire sur une ligne: **//** ou **#**
- Commentaire sur plusieurs lignes: **/* ... */**

exemple :
```
nom : <b>  <?php  echo $nom ?> </b> // affiche le nom de la personne en gras
prénom : <b>  <?php  echo $prenom ?> </b> /* affiche le prénom de la personne en gras */
```

---

# 3 - Installation de PHP

Installation de php peut se faire de différentes manières.

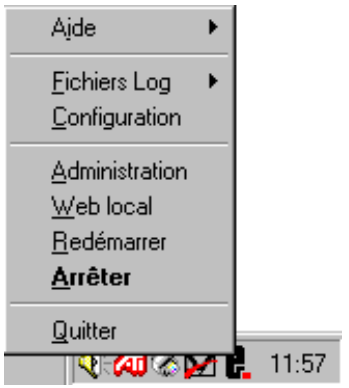
## 1- Installation de easyphp

La plus simple est l'utilisation de package **easyphp** disponible sur le site www.easyphp.org . Ce package installe le moteur de script php, le Système de Gestion de Base de Données MySQL, le serveur apache et l'interface PhpMyAdmin qui permet de gérerle SGBD Mysql (voir plus loin). ce type d'installation est conseillée lorsque on veut utiliser l'envirenement PHP/MySQL.

Pour installer **easyphp**, il suffit de lancer le fichier d'installation et de se laisser guider.

Le serveur apache installé par cette méthode est consideré comme étant le serveur par défaut, par rapport aux serveurs Internet Information Server (IIS) ou Personnel Web Server (PWS) de windows s'ils sont déja installés. Par conséquent, Pour utiliser des fichiers Active Server Page (ASP), il faudra arrêter le serveur apache, et relancé le serveur PWS ou IIS.


Après l'installation, Easyphp ajoute une icône en bas à droite de la barre des tâches




L'administration, la configuration d'easyphp se font par un clic droit sur cette icône

Cette méthode est simple, mais le package est un peu lourd à télécharger.

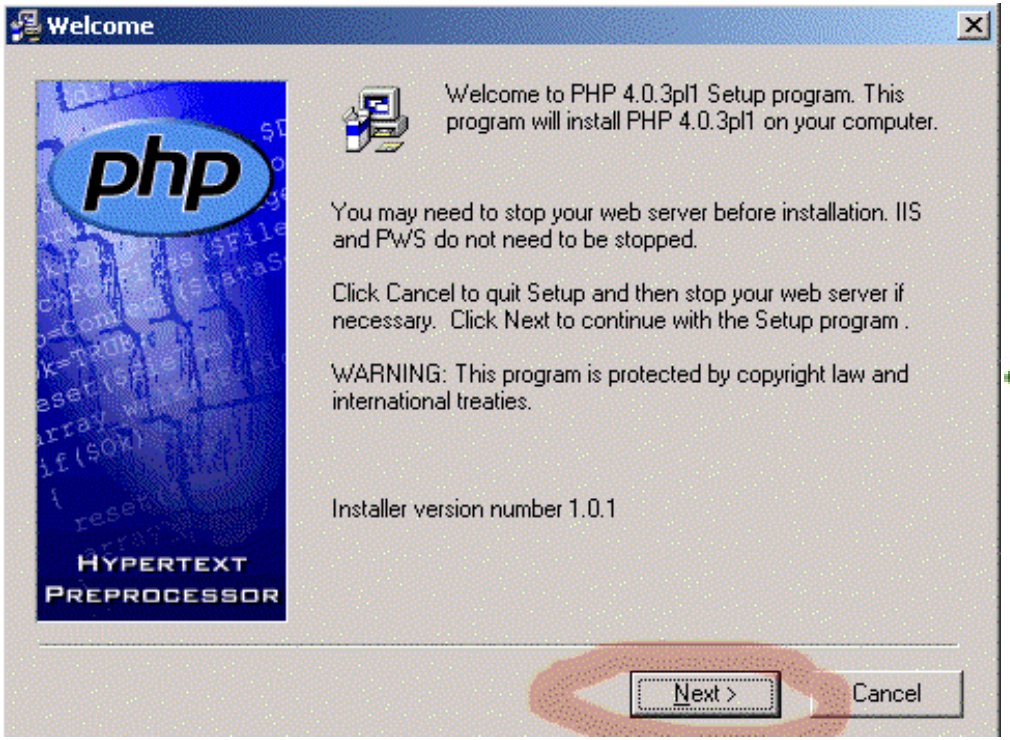
## 2- Installation de moteur de script php

Pour faire fonctionner **que** le php sur un serveur, il suffit de télécharger sur www.php.org ou www. php.net ou d'autres sites, le fichier **phpinstaller.exe**, qui fait moins de 1 Mo.

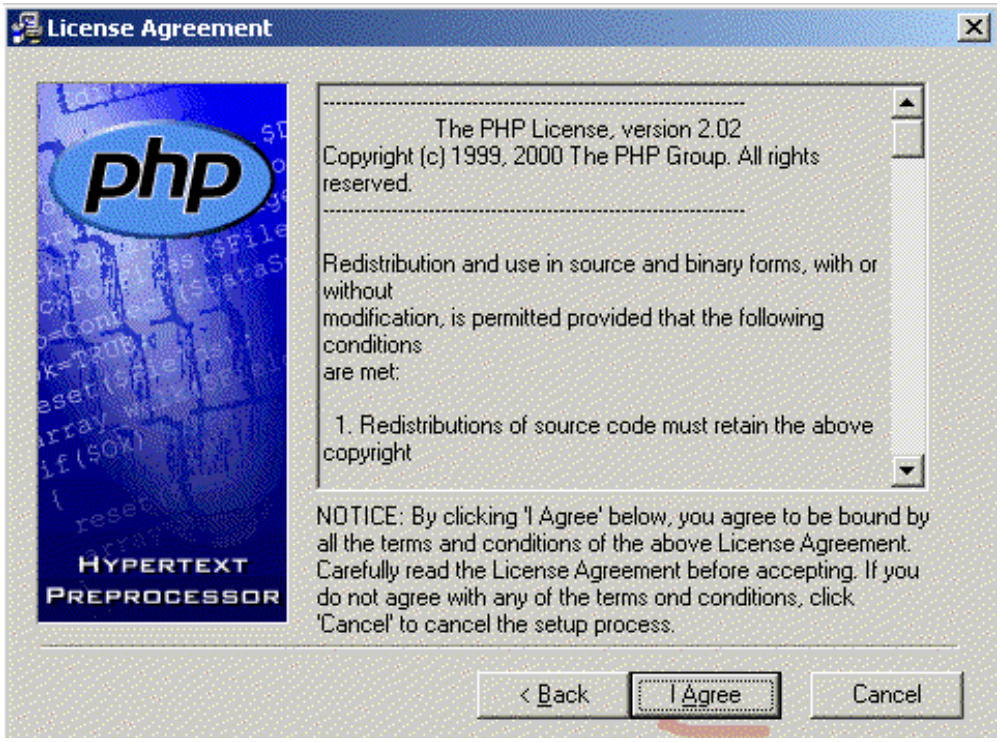
Attention : **phpinstaller.exe** n'installe que PHP. Il vous faudra un serveur web du type apache, IIS, PWS, , ... pour faire exécuter les scripts des pages php.

Les étapes d'installation se déroulent suivant les flèches vertes suivantes :

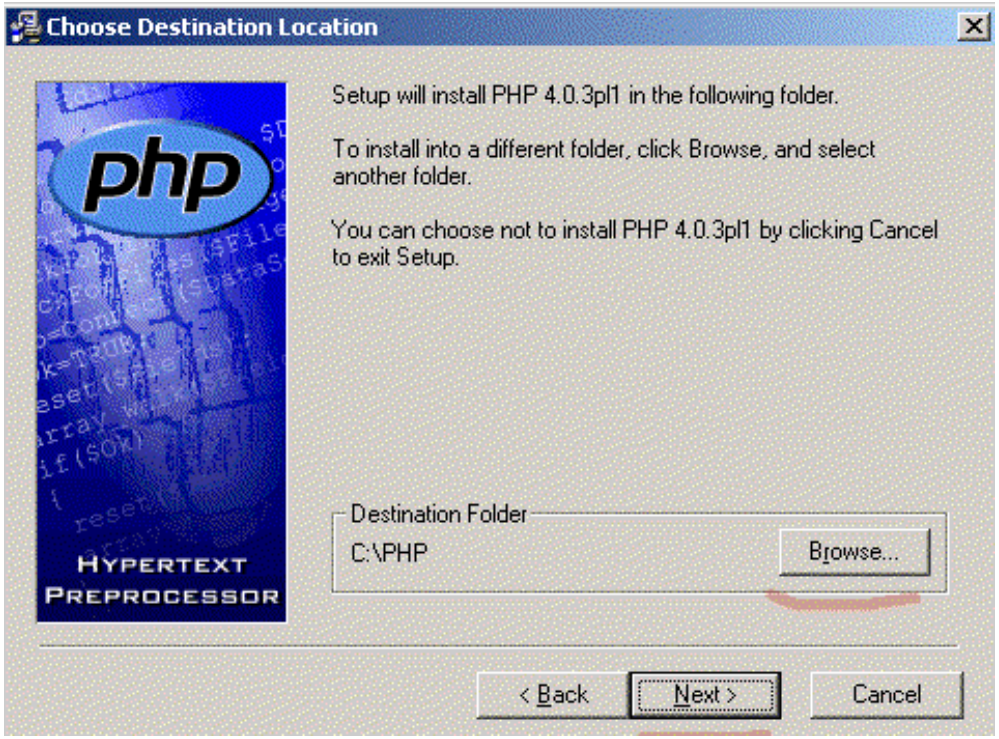
1. lancer phpinstaller.exe



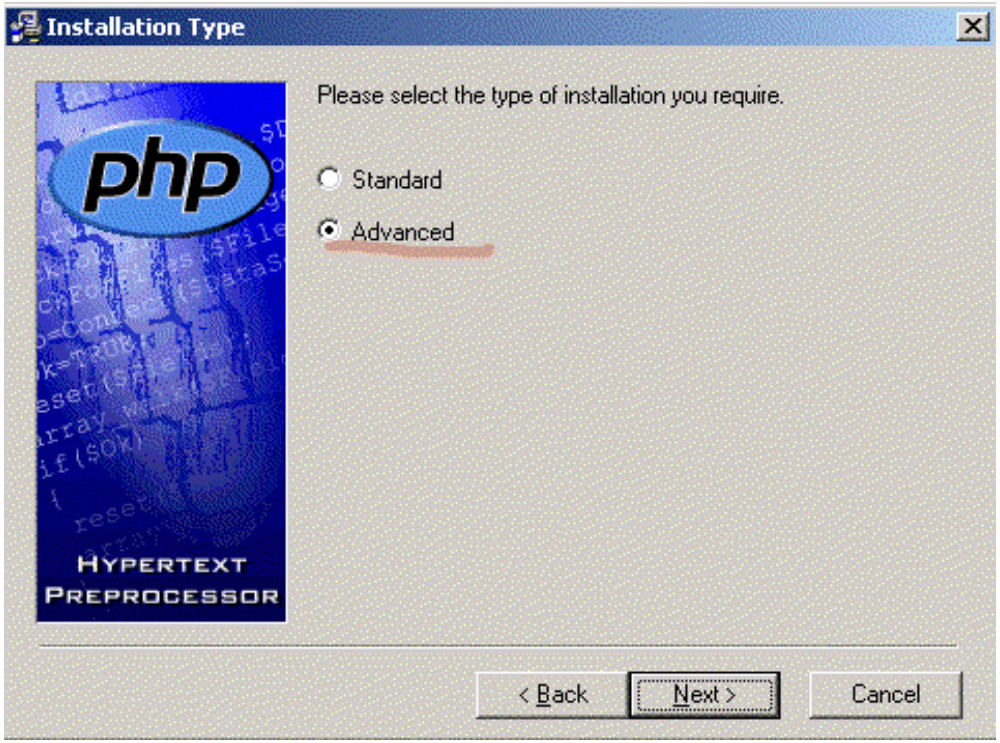

Cliquer sur "suivant"



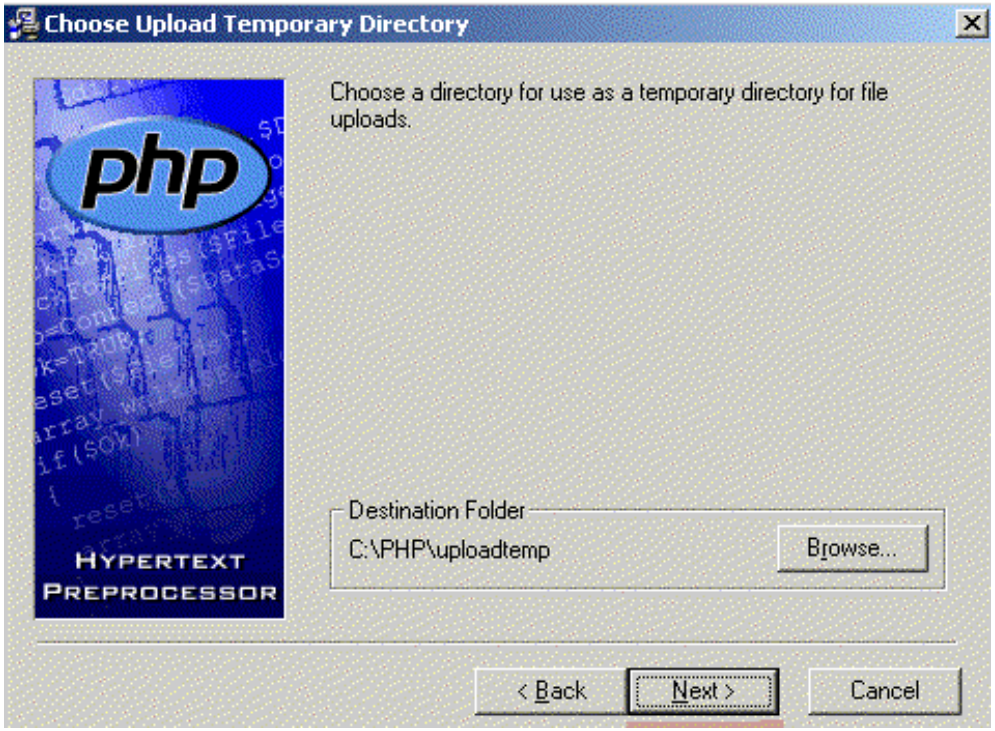

Cliquer sur "J'accèpte"



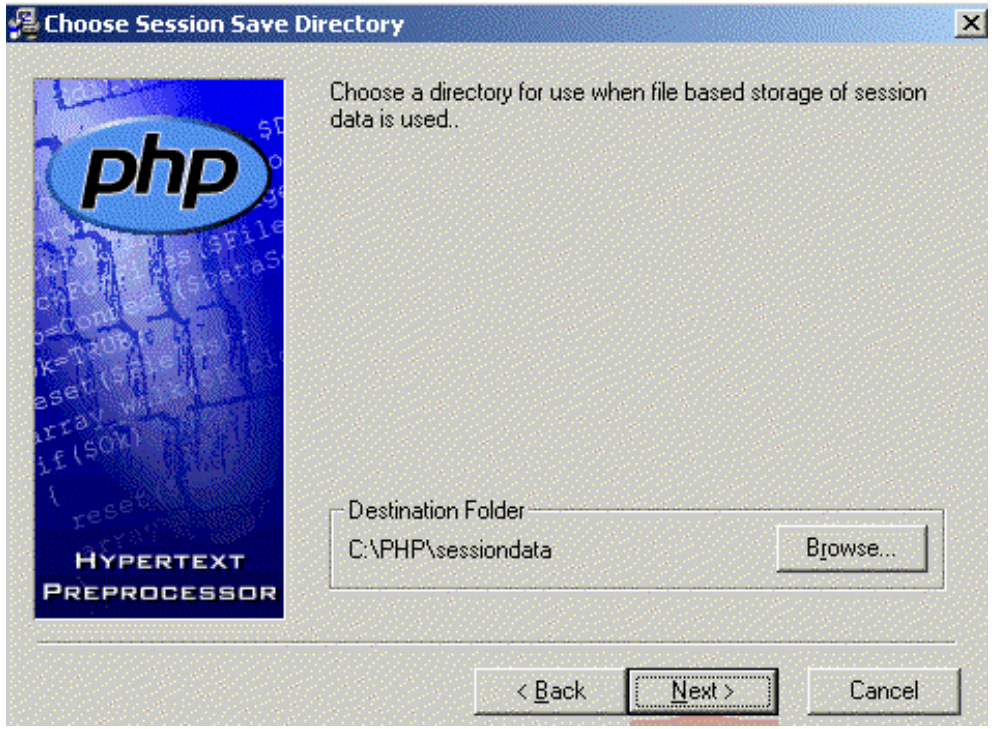

sélectionner le répertoire d'installation de php
(vous pouvez laisser le répertoire par défaut c:\php)



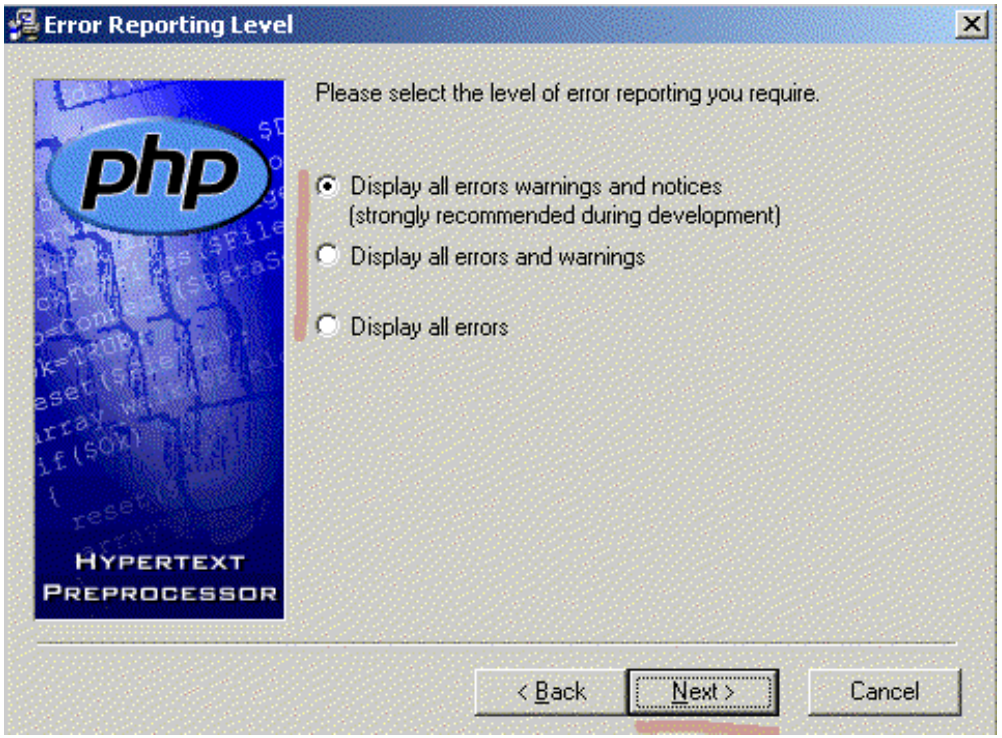

sélectionner le mode avancé pour avoir la main sur plus d'options




laisser le répertoire d'uploads de fichiers temporaires à sa valeur pas défaud

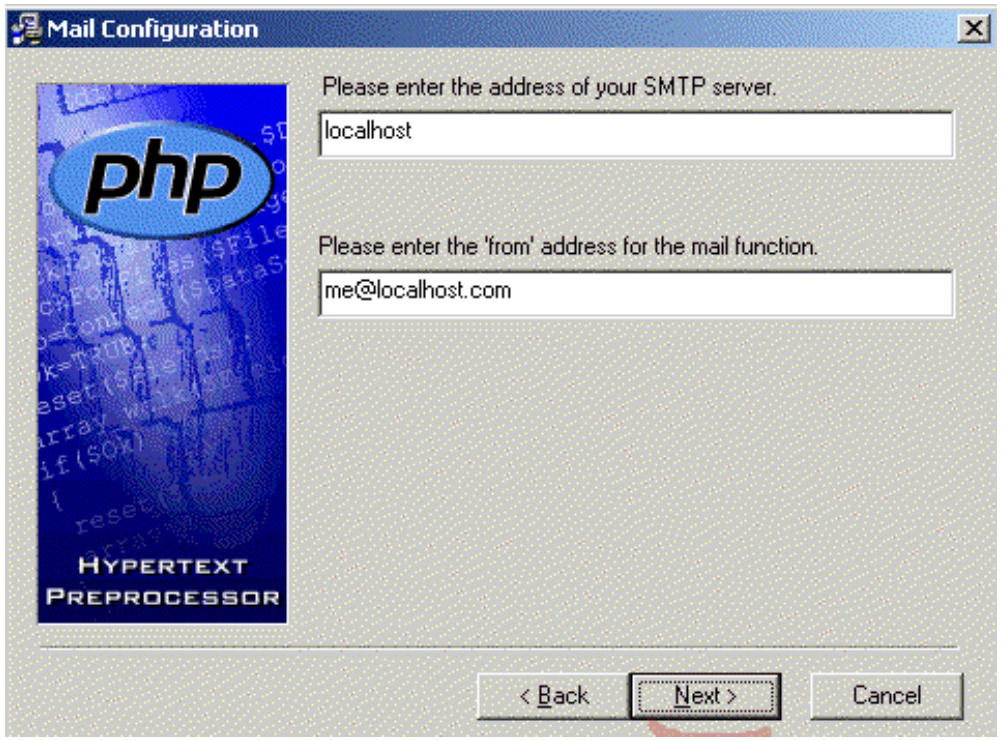



idem pour le répertoire des variables de session

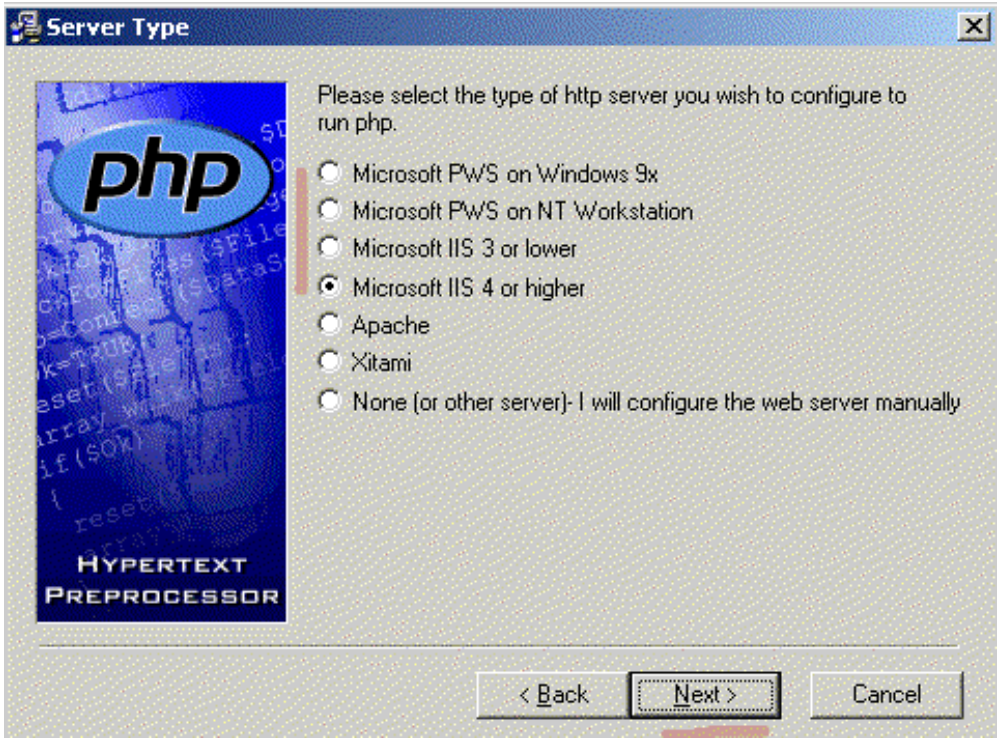



cet écran permet de définir le niveau de rapport d'erreurs de php.
Cette étape est importante

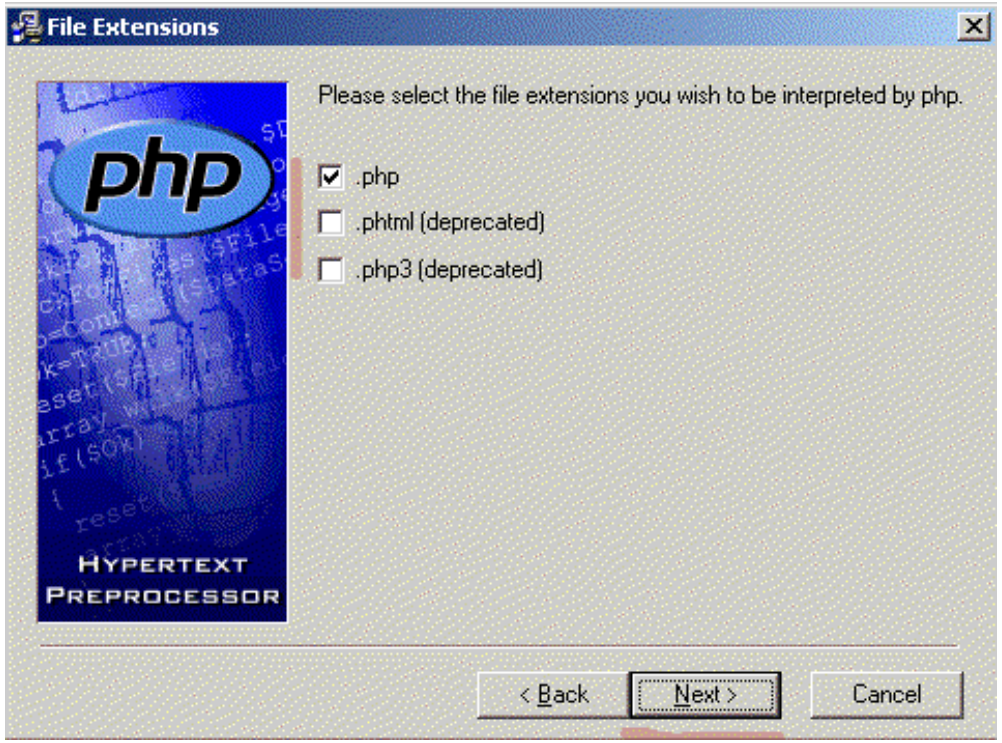
3 possibilités, dont la première et la seconde permettent à l'étape du développement de débogger vos fichiers php, la troisième possiblité est à utiliser sur un vrai serveur web afin de n'afficher que les erreurs de comportement du serveur, de la base de données, ou des fichiers.



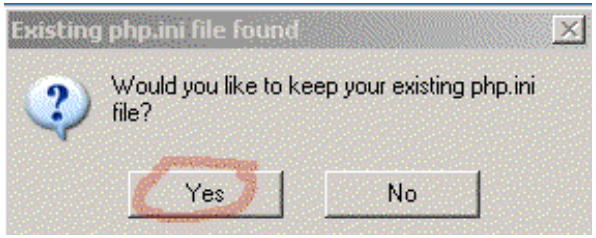

cet écran permet de définir l'adresse d'un serveur SMTP
(pas nécessaire sauf pour gérer des emails)



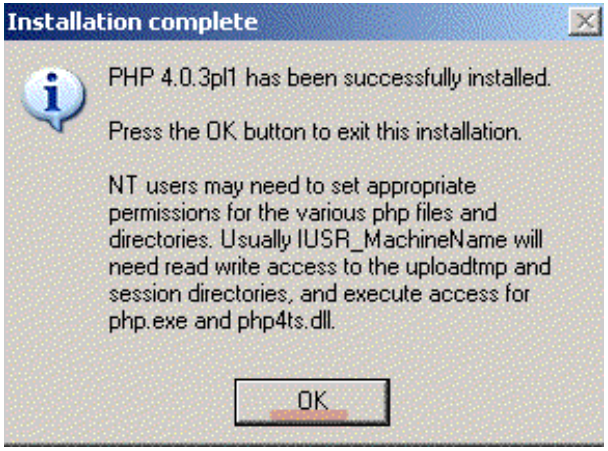

Sélectionnez le serveur installé sur votre ordinateur (sous windows 95/98 vous pouvez avoir installé PWS ou apache, sous windows 2000,NT,XP vous pouvez avoir installé IIS 4 ou 5, ou apache)

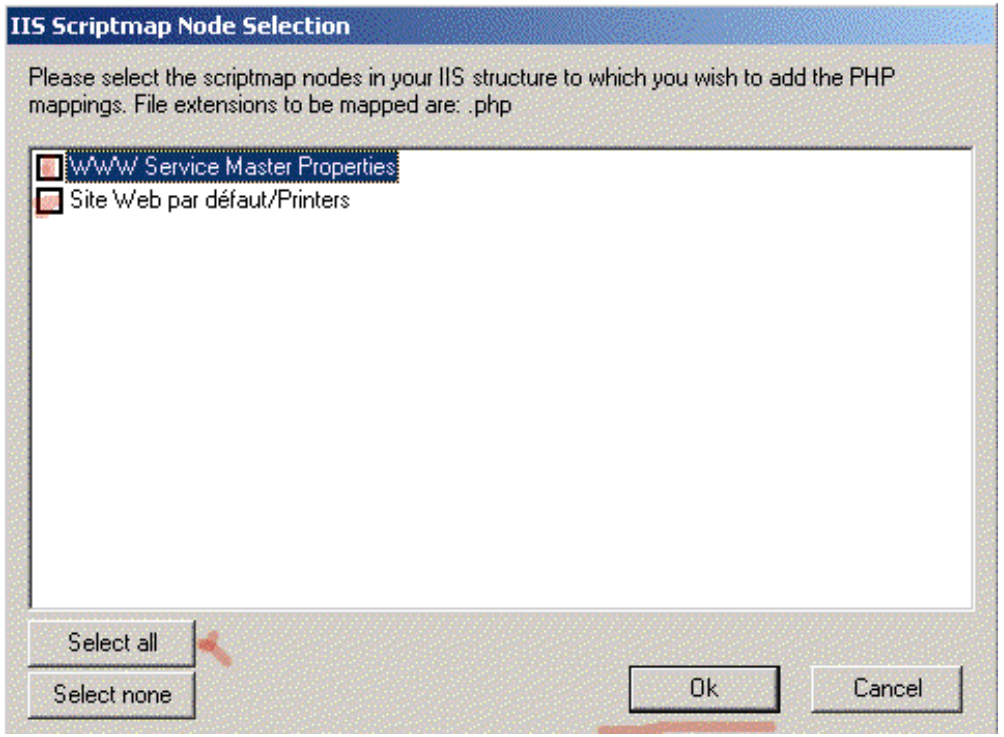



ici vous sélectionnez les types de fichiers interprétés par php. Il est conseillé de donner l'extension .php à vos fichiers pour qu'ils soient reconnus sur les serveurs récents.




Si vous avez déjà fait une installation de php, il vous est proposé de conserver (keep) ou écraser votre fichier de configuration php.ini situé dans le répertoire racine de windows.




si vous avez déja installé IIS, cochez toutes les cases pour que php s'exécute dans tous vos sites.

L'installation proprement dite est terminé

---

# 4 - Configurer Internet Information Serveur IIS d'un serveur Windows

---

On peut utliser des scripts PHP sous une configuration IIS. Dans ce cas, il faut modifier légèrement la configuration de ce dernier

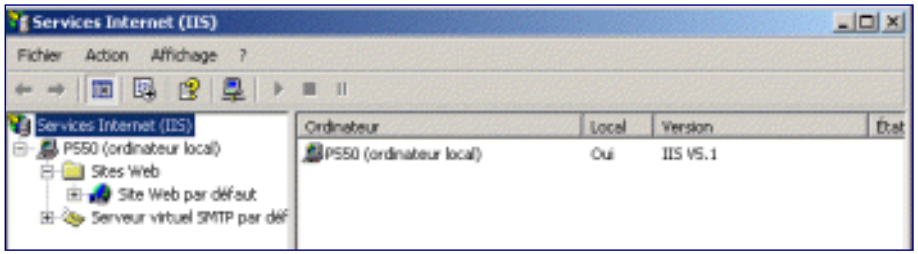
Attention : vous devrez installer le module PHP mais n'installez pas le easyphp puisque ce dernier installe par défaut le serveur apache qui risque de rentrer en conflit avec votre serveur IIS.

## Installation et configuration de IIS

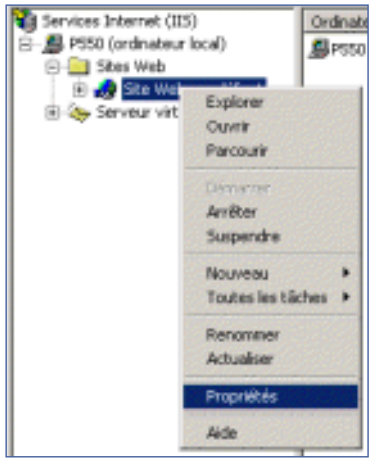
Si IIS n'est pas installé, allez dans le panneau de configuration, "**ajout/suppression de programmes**", puis "**ajouter ou supprimer des composants windows**", et cocher la case "**Services Internet IIS**". Il vous demandera le CD d'installation de windows.

**1)** Après installation de IIS, allez dans "**le panneau de configuration**", (outils d'administration si vous êtes sous windows 2000 ou +), "**Sercices Internet (IIS)**"

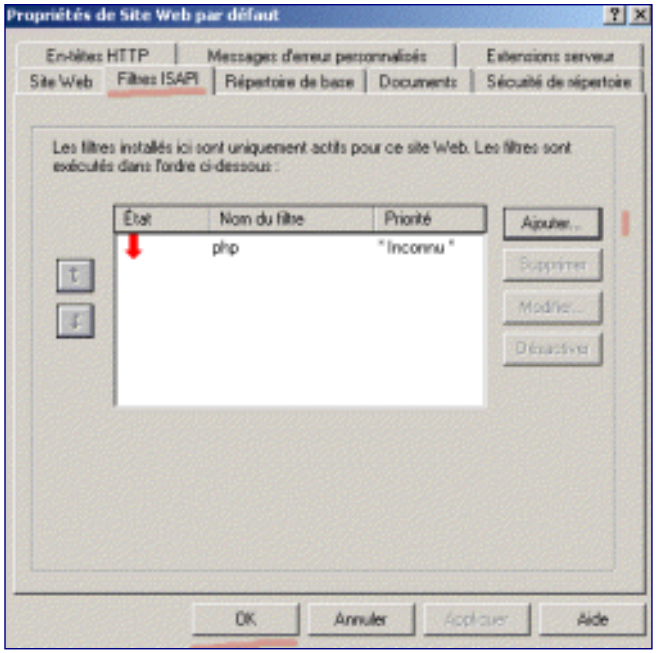
Déployez les menus correspondant au nom réseau de votre machine ici P550 (ordinateur local).




**2)** Clic droit sur "**Site Web par défaut**", sélectionnez "**Propriétés**"

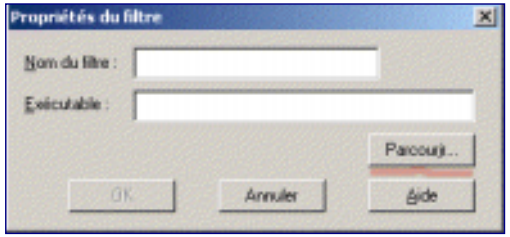



**3)** Cliquez sur l'onglet "**filtres ISAPI**", puis sur "**ajouter**"

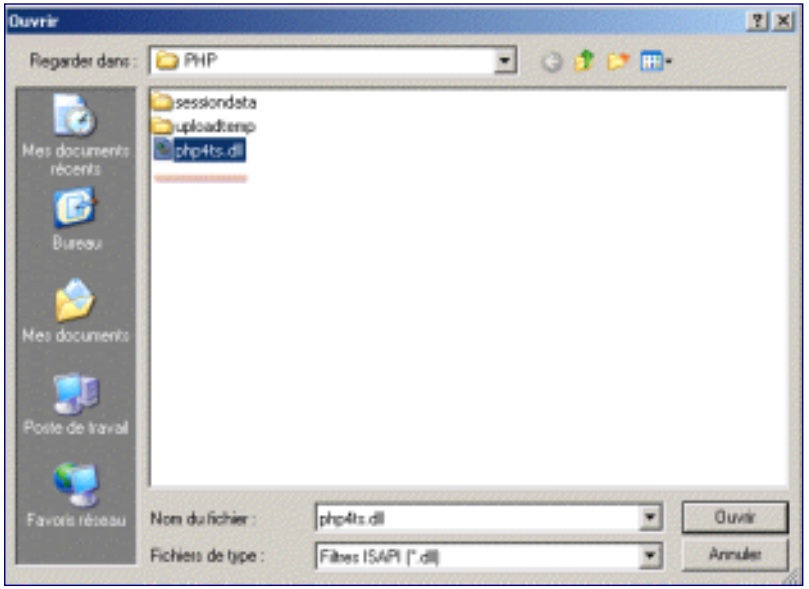



**4)** 3 étapes a) , b) et c)

a) cliquez sur "Parcourir"



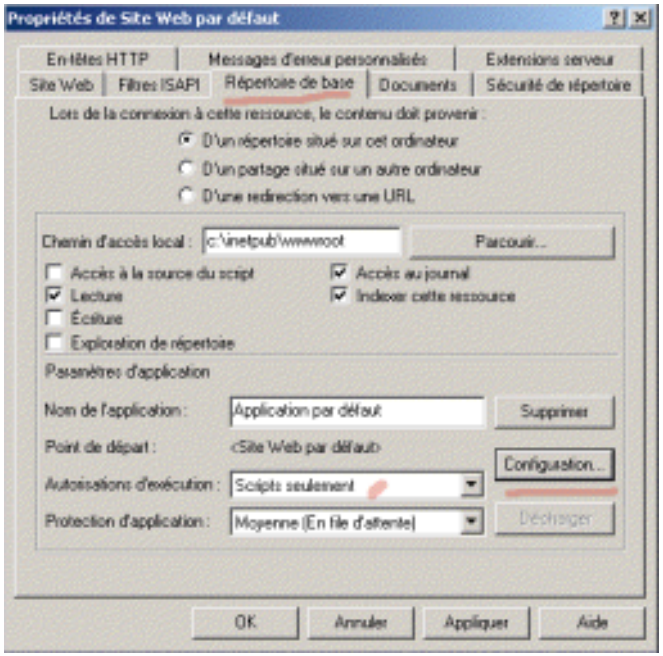

b) allez dans le répertoire de PHP pour sélectionner le fichier **php4ts.dll** et cliquez sur "Ouvrir"



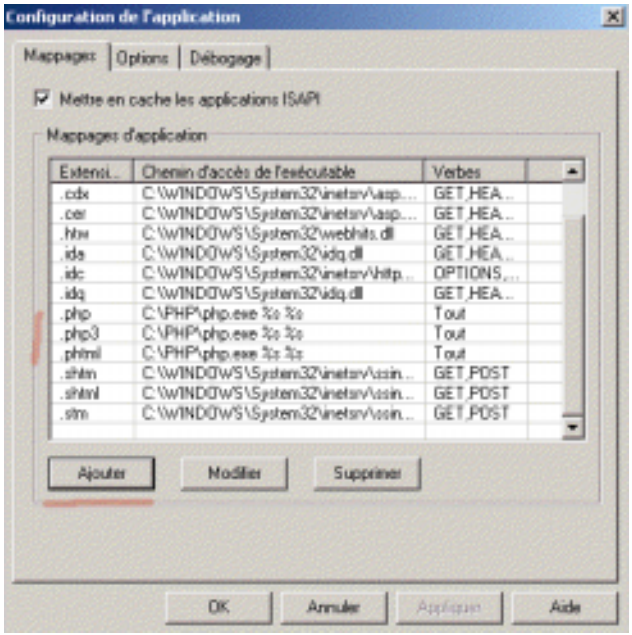

c) N
votr
pour
reco
en ca
beso
cliqu
OK

**5) Autorisation d'exécution de script PHP sur le répertoire de base du serveur web IIS**

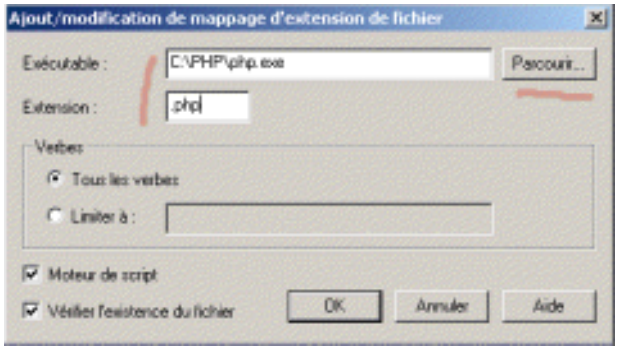
Cliquez sur l'onglet "**répertoire de base**", dans " Autorisations d'exécution" , sélectionnez "**Script seulement**", ou " **exécuter et scripts**" , cliquez ensuite sur "**Configuration**".




vérifiez si les fichiers d'extension **.php** existent dans la liste (le cas ci-dessous). S'ils n'existent pas, cliquer sur ajouter




cliquez sur parcourir, allez dans le répertoire de php, sélectionnez le fichier **php.exe** qui permettra d'interpréter vos fichiers, cliquez sur ouvrir, puis donnez l'extension des fichiers qui seront interprétés (il est conseillé d'inscrire .php)




IIS est prêt pour reconnaitre php. Placez vos fichiers dans le répertoire de base et, à l'aide d'un navigateur Web, tapez l'adresse de votre serveur suivi d'un nom de fichier php de votre site, il sera interprété.

exemple :

http://p550/nomdefichier.php

http://adresse IP de votre machine si vous êtes en réseaux LAN/nomdefichier.php

http://127.0.0.1/nomdefichier.php

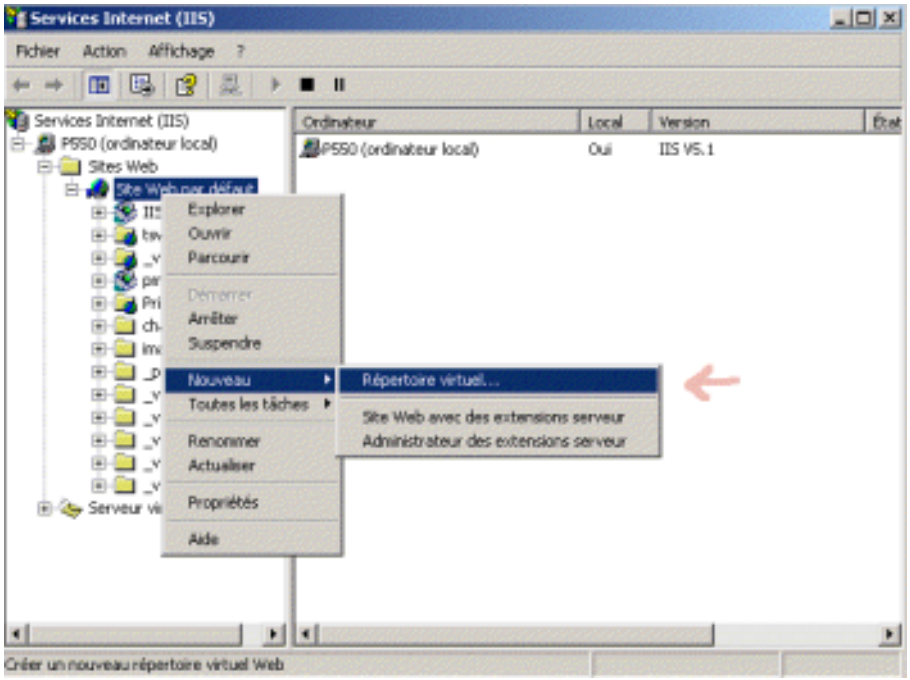
http://localhost/

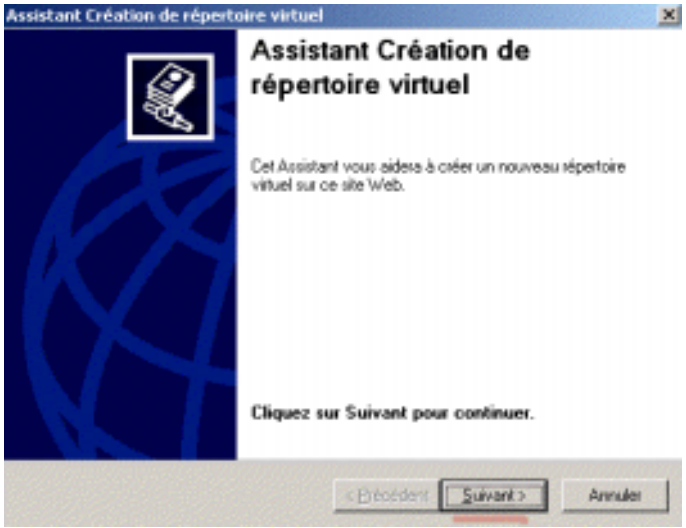
Parfois, notamment sous windows 2000, un seul de ces exemples ci-dessus fonctionne, cela dépend de la configuration.

## Répertoire Virtuel

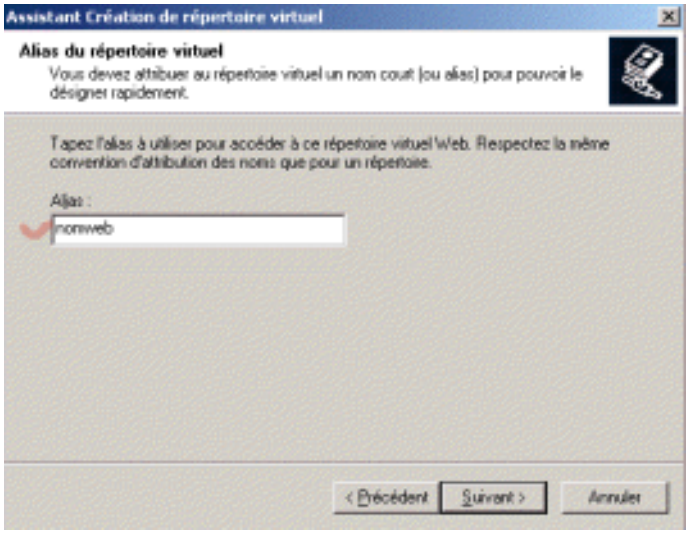
Vous pouvez placer vos fichiers Web dans un autre répertoire que le répertoire de base www.

Pour ce faire, créez un répertoire virtuel pour votre nouveau site (clic droit sur Site Web par défaut, Nouveau, répertoire virtuel). (Même principe pour PWS)

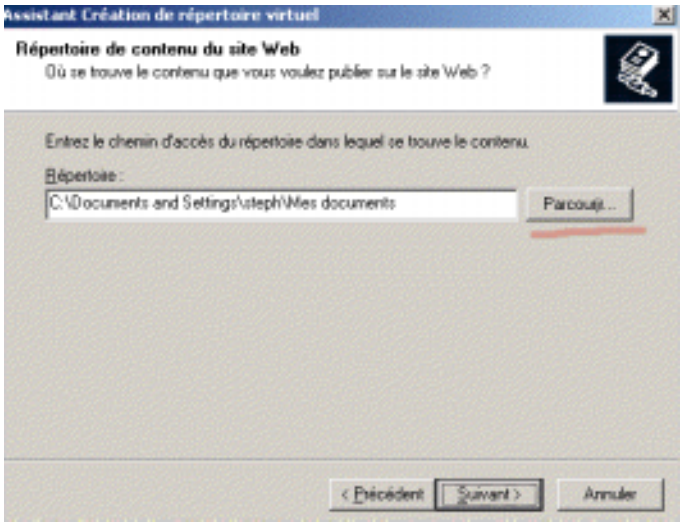


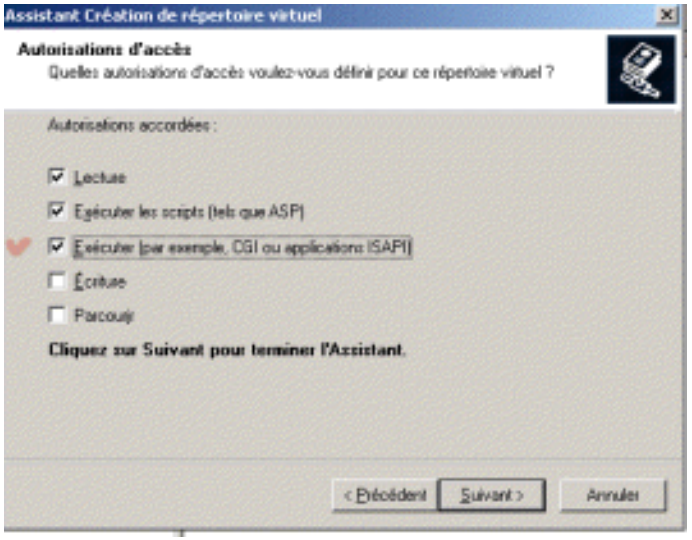

l'alias est le nom qui permettra d'accèder aux fichiers par l'intermédiaire du navigateur Web




sélectionnez le répertoire




cochez la case exécuter




L'accès aux fichiers de ce répertoire se fait ainsi :

http://localhost/nomweb/nomdeficihier.php

où nomweb est l'alias du répertoire virtuel

---

# Technologies de Script

## Les bases du langage PHP

---

Objectifs

1. Les types
2. Les constantes
3. Les variables
4. Les tableaux
5. Les instructions de contrôle
6. Les boucles

---

Mr Mohamed SIDIR

# 1 - Les types

---

Contrairement à des langages comme le C ou le Java, PHP dispose de peu de type de données différentes. De plus, il n'est pas nécessaire de déclarer les types de chaque variable. Le type d'une variable ou d'une constante est défini par la valeur qui lui a été affectée lors de sa création. Il existe cinq types de données :

| type | mots clés |
|---|---|
| Entier | **integer** |
| Décimal | **real, double, float** |
| chaîne de caractère | **string** |
| tableau | **array** |
| Objet | **object** |
| Booléen (PHP4) | **bool** |

Dans l'exemple : **$var = 5**, la variable $var est de type integer. Il est inutile de déclarer le type de la variable lors de sa création.

---

# 2 - Les constantes

---

Une constante est un identifiant qui représente une valeur simple. Comme leur nom le suggère, cette valeur ne peut jamais être modifiée durant l'exécution du script (les constantes magiques __FILE__ et __LINE__ sont les seules exceptions). Le nom d'une constante est sensible à la casse ( le même nom en majuscule et en minuscule sont différents).

Les noms de constantes suivent les mêmes règles que n'importe quel nom en PHP. Un nom valide commence par une lettre ou un souligné (_), suivi d'un nombre quelconque de lettres, chiffres ou soulignés.

Syntaxe :

Pour créer une constante, il faut un identifiant (un nom) valide et la créer en utilisant la fonction :

**define("nom_de_la_constante", valeur_de_la_constante )**

**nom_de_la_constante :** peut être une chaîne de caractères donc entre guillemets ou un nombre

**valeur_de_la_constante :** la constante a le type de la valeur qu'on lui a affectée. Cette valeur n'est plus modifiable par la suite.

L'appel de la constante ainsi créée se fait par le nom choisi (sans le signe dollar $).

**Exemple 0**

```php
<?php

define("MA_CONSTANTE", "Do bee do bee do ...");

define("annee", 1970);

echo CONSTANTE, " est du type ", gettype (MA_CONSTANTE), "<br>";

echo annee, " est du type ", gettype(annee), "<br>";

?>
```

**Résultat**

La fonction **gettype()** permet de définir le type de la constante (ou la variable).

## Les constantes prédéfinies

PHP définit un certain nombre de constantes prédéfinies dont certaines fournissent des informations utiles. Ces constantes sont toujours disponibles. En voici la liste non exhaustive.

| | |
|---|---|
| PHP_VERSION | La chaîne de caractères de présentation de la version du PHP qui est actuellement utilisée. Par exemple '4.0.0'. |
| PHP_OS | Nom du système d'exploitation du serveur qui fait tourner le PHP. Parmi les valeurs possibles : "AIX", "Darwin" (MacOS), "Linux", "SunOS", "WIN32", "WINNT". (liste non exhaustive) |

| | |
|---|---|
| E_ERROR | Dénote une erreur autre qu'une erreur d'analyse ("parse error") qu'il n'est pas possible de corriger. |
| TRUE et FALSE | représentent les valeurs booléennes (vrai et faux) |
| NULL | représente le type retourné par la fonction gettype() pour une variable qui n'a pas encore été affectée |
| __FILE__ | insensible à la casse, il revoie le nom du fichier qui est actuellement exécuté |
| __LINE__ | insensible à la casse, il revoie le numéro de la ligne qui est actuellement exécutée |

# 3 - Les variables

---

PHP n'est pas un langage fortement structuré. A ce titre, il ne contient pas de partie déclarative clairement définie, comme on peut en trouver dans du code écrit en C ou Visual Basic. Pour définir une variable, il suffit de l'initialiser.

Les variables PHP sont toutes précédées du signe $. Ainsi pour déclarer une variable nommée ma_variable on écrit :

```
$ma_variable=1
```

La variable $ma_variable est donc définie, sa valeur est 1 et elle est de type integer.

Toutefois, il est possible de réaliser des conversions. Dans ce cas, le type de données ciblées doit être explicitement mentionné :

**Exemple :**

```
<?php

$ma_variable=1;      // $ma_variable est de
type integer et vaut 1

$chaine = (string) $ma_variable ;  /*   $chaine
est de type "string" chaîne
de                                   caractère
et vaut "1" */
?>
```

Depuis la version 4 de PHP, il est possible de supprimer une variable, à tout moment, en utilisant la fonction unset.

```
unset($ma_variable) ; // $ma_variable est supprimée du
programme
```

**a) Portée d'une variable :**

Une variable définie au sein d'une fonction par exemple n'est pas accessible en dehors de celle-ci. De même une variable définie en dehors d'une fonction n'est pas accessible par celle-ci. (Plus loin, nous verrons en détails comment définir et utiliser les fonctions en php).

Pour comprendre la notion de portée d'une variable, examinons cette exemple :

```
le résultat est :

<?php

$a =1;
$b = 2;

Function somme() {
$c = $a + $b;
echo "$c";
}
somme ();

?>
```

**Résultat**

Dans cette exemple la valeur affichée est 0 car les deux variables **$a** et **$b** ne sont pas connues à l' intérieur de la fonction somme().
Pour les utiliser dans la fonction, il faut les déclarer comme variables globales.

## b) Variables globales

Les variables globales sont visibles dans l'intégralité du code d'une page PHP, et persistent pendant tout le temps d'exécution du code.

Exemple :

```
le résultat est :

<?php

$a = 1;
$b = 2;

Function somme() {

global $a, $b, $c;
$c = $a + $b;
}
somme ();
echo "$c";

?>
```

**Résultat**

On peut définir les variable globales de la façon suivante :

```
Function somme() {
$GLOBAL["c"] = $GLOBAL["a"] + $GLOBAL["b"];
}
```

## c) Variables Statiques :

Les variables statiques sont visibles uniquement à l'intérieur de la fonction où elles sont définies. La fonction **static** garde leur valeur courante, d'un appel de la fonction à l'autre.

Exemple :

```
<?php

<?php
function essai () {
static $a=0;
echo "$a<br>";
$a++;
}
essai();
essai();
essai();
essai();
essai();

?>
```

**Résultat**

Dans cette exemple, le premier appel de la fonction initialise la variable $a à 0, puis incrémente celle-ci à chaque appel. (la valeur de la variable $a de l'appel précédent est sauvegardée pour l'appel suivant).

## d) Les variables "variables"

Il est possible de redéfinir une variable comme variable. A ce titre, le nom des variables est construit dynamiquement tout au long du code PHP :

exemple :

```
<?php

$nom = "Monsieur Dupond";
$$nom ="n'est pas généreux";
echo " $nom ${$nom}.";

?>
```

**Résultat**

## e) Les variables en provenance d'un formulaire

Le point de départ de la programmation de pages dynamiques est constitué par les formulaires HTML. Outre la possibilité de transférer des données par l'intermédiaire d'un lien hypertexte ou à l'aide des cookies, sur lesquels nous aurons l'occasion d'y revenir, les formulaires HTML constituent une excellente méthode de transfert des données d'un poste client à un serveur web.

L'une des principales raisons du succès du langage PHP est que les scripts PHP permettent de manipuler très facilement les données de formulaires. Cela est tout d'abord lié au fait que chaque nom indiqué dans un formulaire pour identifier une zone de saisie (champ texte, case à cocher, champ masqué, bouton,...) est automatiquement transformé en une variable par PHP.

Création d'un formulaire simple de saisie en HTML :

```
<form action="passge-var.php"
method="POST">

Nom: <input type="text"
name="nom"><BR>

Prénom: <input type="text"
name="prenom"><BR>

<input type="submit">

</form>
```

Le formulaire est défini par la balise ouverte **<form>** est la balise fermé **</form>.**

La balise **FORM** possède des attributs suivantes :

- **action** : spécifie l'adresse à laquelle le contenu du formulaire doit envoyé, dès que l'utilisateur appuie sur le bouton correspondant.
- **method** : décrit la méthode HTTP (GET ou POST) utilisée pour le transfert des données du formulaire
- **input** : : il s'agit de champ de saisie. Il est constitué d'une seule balise accompagnée de plusieurs attributs.
- **name** : définit le nom de champ de saisie

.....la liste est non exhaustive.

le code PHP de la page **passge-var.php** permet le passage des variables de formulaire ci-dessous :

```
<table border="0" width="100%" bgcolor="#99FF66">
<tr>
<td width="18%"><font face="Verdana"
size="2"><b>Nom :</b></font></td>
<td width="82%"><?php echo $nom?></td>
</tr>
<tr>
<td width="18%"><font face="Verdana"
size="2"><b>Prénom :</b></font></td>
<td width="82%"><?php echo $prenom?></td>
</tr>
</table>
```

**Résultat**

Le PHP a crée automatiquement des variables du même nom que les champs de saisie du formulaire. Si vous tapez "toto" et"tata" dans les deux champs de formulaire, les variables **$nom** et **$prenom** dans le code PHP de la page **passge-var.php** ont pour valeurs respectivement "toto" et"tata". Nous reviendrons sur les méthodes de transferts des données GET et POST car elles méritent plus d'intentions.

---

# 4 - Les tableaux

---

Un tableau est en fait une variable avec plusieurs valeurs regroupées sous forme d'un tableau en lignes et en colonnes. Comme dans d'autres langages, l'utilisation des tableaux est parfois incontournable. Dans PHP leur manipulation est particulièrement simple et efficace.

La déclaration d'un tableau se fait par affectation, de la même manière que la déclaration d'une variable.

```
$tableau[0]=5 ;   /* le tableau est créé et sa première valeur est 5 */
```

Pour un tableau à 2 dimensions, il suffit de mettre un second indice au moment de l'affectation

```
$tableau[0][0]=2 ;   /* le tableau à 2 dim est créé et sa première valeur est 2 */
```

La déclaration et l'initiation d'un tableau peuvent également se faire par l'intermédiaire de la fonction **array()**. Cette fonction permet de préciser les indices à l'aide de l'opérateur **=>** ainsi que les valeurs du tableau :

exemple :

```
$tableau[0]=5 ;
$tableau[1]="coucou" ;
$tableau[2]=6 ;
$tableau[3]="ville" ;
```

ou s'écrire d'une façon plus simple :

```
$tableau=array(0=>5,1=>"coucou",2=>6,3=>"ville")
```

**exemple :**

```php
<?php

$Personnel[0] = "N°1" ;
$Personnel[1] = "MILLE" ;
$Personnel[2] = "Patrick" ;
$Personnel[3] = "service comptabilité" ;
$Personnel[4] = "M" ;
$Personnel[5] = "03 22 80 81 39" ;

$i=0;
while ($i<count($Personnel))
{
echo $Personnel[$i], "<br>";
$i++;
}

?>
```

**Résultat**

---

# 5 - Les instructions de contrôle

---

## a) Opérateurs

Les opérateurs de logiques permettent de combiner plusieurs tests entre eux. Dans les tableaux suivants, les variables a et b peuvent prendre les valeurs vrai ou faux (variables booléennes).

Les tableaux suivants facilitent la recherche de code ou de valeurs utilisées par les fonctions du langage. Ils regroupent l'ensemble des opérateurs reconnus par PHP.

### i) Opérateurs logiques

| Opérateurs | Opération | Effet |
|---|---|---|
| AND && | $a and $b ou $a && $b | Vrai(TRUE) si $a et $b sont vrais tous les deux |
| OR \|\| | $a OR $b ou $a \|\| $b | Vrai(TRUE) si $a est vrai ou si $b est vrai ou si $a et $b sont vrais tous les deux. |
| XOR $a xor $b | $a xor $b | Vrai(TRUE) si $a est vrai ou si $b est vrai. Mais Faux (False) si $a et $b sont vrais tous les deux. |
| NOT ! | !$a(Not) | vrai (True) si $a n'est pas vrai |

### ii) Opérateurs de comparaison :

| Opérateurs | Opération | Effet |
|---|---|---|
| == | $a == $b | Vrai(TRUE) si $a égle à $b. |
| != | $a !=$b | Vrai(TRUE) si $a est différent de $b. |
| < | $a < $b | Vrai(TRUE) si $a inférieur à $b. |
| > | $a > $b | Vrai(TRUE) si $a supérieur à $b. |
| <= | $a <= $b | Vrai(TRUE) si $a inférieur ou égale à $b. |
| >= | $a >= $b | Vrai(TRUE) si $a supérieur à égale à $b. |

### iii) Les opérateurs mathématiques

| Opérateurs | Opération | Effet |
|---|---|---|
| + | addition $a + $b | 21 + 5 = 26 |
| - | soustraction $a + $b | 21 - 5 = 16 |
| * | multiplication $a * $b | 21 * 5 = 105 |
| / | division $a / $b | 21 / 5 = 4,2 |
| % | modulo: $a % $b | 21 % 5 = 4 |

### iv) Raccourcis pour l'utilisation des opérateurs

| Opérateurs | Opération | Effet |
|---|---|---|
| ++ | Incrément de 1 $a ++ | Ajout 1 à $a et affecte le résultat à $a |
| +=n | Incrément de n $a += | Ajout n à $a et affecte le résultat à $a |
| -- | Décrément de 1 $a-- | Retranche 1 de $a et affecte le résultat à $a |
| -=n | Décrément de n $a-=n | Retranche n de $a et affecte le résultat à $a |
| *=n | Multiplication par n $a*=n | Multiplie $a pr n et affecte le résultat à $a |
| /=n | Division par n $a*/=n | Divise $a par n et affecte le résultat à $a |

## b) L'instruction if

IF est l'instruction conditionnelle de base. Elle permet de n'exécuter une ou plusieurs instructions que si une condition (ou une expression contenant plusieurs conditions) est remplie.

Cas simple de l'utilisation de de l'instruction if.

```
<?php

if ($a > $b)

print "a est supérieur à b"; // ne s'affichera que si $a
est supérieur à $b

   ----------------------------------------

if (($a > 4) and ($b > 4))

print "a et b sont supérieurs à 4"; // ne s'affiche que si
$a et $b sont supérieurs à 4


----------------------------------------

if ($a > $b)

print "a est supérieur à b"; // plusieurs instructions
conditionnées

print "b est inférieur à a";

   ----------------------------------------

}

?>
```

Le listing ci-dessous fournit un exemple d'utilisation de l'instruction if. Celle-ci exécute un bloc de code si l'expression comprise entre parenthèse est évalue à la valeur true. A l'inverse, si la valeur est false la totalité du bloc est ignorée.

### c) L'instruction if...elseif ....else

Pour tester une série de conditions, on peut également utiliser l'instruction elseif. Deux syntaxes différentes peuvent être employées :

La première est très semblable à celle du langage C. Elle se présente de la façon suivante :

```
<?php

if (condition1) {

action 1

} elseif
(condition2)  {

action 2

} else {

action 3

}

?>
```

la seconde est une variante de la première dans laquelle les accolades ont été supprimées :

```
<?php

if
(condition1) :

action 1

elseif
(condition2) :

action 2

else :

action 3

endif ;

?>
```

Dans les deux cas, le principe est le même : l'action exécutée est la première dont la condition est respectée. Si aucune des conditions n'est respectée, l'action "sans condition" (celle engendrée par else) est exécutée.

NB :

Dés qu'une condition est respectée, son action est exécutée et les conditions suivantes ne sont pas évaluées, Ainsi, si deux conditions sont respectées, seule la première verra son action exécutée.

---

# 6 - Les Boucles

---

Les boucles sont, avec les instructions de contrôle, très utilisées en programmation PHP. Elles permettent la répétition d'une série d'instructions. Le nombre de répétitions peut être prédéterminé ou dépendre d'une condition particulière.

En PHP, trois types de boucles sont disponibles : while, do....while et for.

### while

L'instruction la plus simple pour traiter les boucles est l'instruction while. Sa syntaxe est la suivante :

```
<?php

while
(conditions)

}

....instructions

}

?>
```

La répétition se fait tant que la condition placée entre parenthèses après while est vraie.

---

# Technologies de Script

## Accés aux bases de données en PHP

---

Accés aux bases de données en PHP

---

Concepteurs :

Mohamed SIDIR

# Chapitre 5

# Accés aux bases de données en PHP

En PHP l'accés à une base de données peut s'effectuer de deux manières, soit en passant par la couche ODBC, soit avec des fonctions spécifiques qui agissent directement sur la base de données. La seconde méthode est beaucoup plus rapide que la première, mais ne peut être employée dans tous les cas.

L'utilisation de la couche ODBC est obligatoire pour des bases du style ACCESS. Examinos un exemple simple utilisant une base de données ACCESS (le code reste valable pour toute base passant par la couche ODBC)

Nous prendrons comme exemple une base que nous avons déclarée sous l'alias base1 dans la couche ODBC, composée de deux tables, "personnel" et "services". (voir les tables)

En général, pour des bases de données mises en place sur le Web, il est d'usage d'avoir différents types d'accès:

- un accès pour les personnes consultant les données en format HTML.

- un deuxième accès "Sécurisé" reservé à l'administration de la base.

## 1. Accès à la base de donnée : Consultation

le code ci-dessous permet de la connexiuon et la consultation :

```
<?php


$user=""; //utilisateur autorisé à accéder à intervenir sur la base de données
//$host="localhost"; //adresse du serveur, ici inutile car liaison ODBC
$password=""; //mot de passe d'accès à la base de données
$db="base1"; // nom de la base de donnée défini dans le DSN systeme

$conn=odbc_connect($db,$user,$password); //connexion à la base
```

```
$sql="select * from personnel,services ";//requête
$sql=$sql."where personnel.id_pers=services.id_pers;";//suite de la
requête
$res=odbc_exec($conn,$sql);//exécution de la requête
?>


<html>
<!-- date de creation: 12/08/01 -->
<head>
<title>consulter les données de la base</title>
</head>
<body>
Personnel responsable d'un ou plusieurs services :<br><br>
<table border="1">
<tr>
<td>Nom </td>
<td>Prénom</td>
<td>Ville</td>
<td>âge</td>
<td>dirige le service</td>
<td>heures par semaine</td>
</tr>
<?
while(odbc_fetch_row($res)) //création d'un objet permettant
// d'accéder aux différents champs d'une ligne de données
{
?>
<tr>
<td><?echo odbc_result($res,"nom") //affichage du contenu du champ
nom

?></td>
<td><?echo odbc_result($res,"prenom")?></td>
<td><?echo odbc_result($res,"ville")?></td>
<td><?echo odbc_result($res,"age")?></td>
<td><?echo odbc_result($res,"service")?></td>
<td><?echo odbc_result($res,"duree")?></td>
</tr>
<?
}
```

```
?>


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

exécuter le code

A noter la fonction **odbc_fetch_row()**, utilisée pour accéder aux données d'une ligne de la base. L'équivalent existe pour toutes les bases de données, mais sous des bases de données du style SQL Server ou Mysql (cette dernière ayant été développée à la base pour le système d'exploitation Unix avec un server Apache) il est plus judicieux d'utiliser la fonction mssql_fetch_object() (respectivement mysql_fetch_object()) qui permet un accès aux données avec la notion de pointeurs (ici nous aurions écrit $row=mssql_fetch_objet($res); et pour afficher le résultat du champ prenom, echo $row->prenom;

Voyons maintenant un exemple d'accès pour un administrateur :

```
<?php

//afin de ne pas aller chercher la page dans le cache :
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // Date du passé
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT"); // toujours modifié
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Pragma: no-cache"); // HTTP/1.0
$user="";//utilisateur autorisé à accéder à intervenir sur la base de données
//$host="localhost";//adresse du serveur, ici inutile car liaison ODBC
$password="";//mot de passe d'accès à la base de données
$db="base1"; // nom de la base de donnée
$conn=odbc_connect($db,$user,$password);//connexion à la base
if (!isset($param))//vérification de la définition du paramètre param qui vaut 1 si l'on vient de la page //insertion,modification ou suppression, n'est pas défini au moment de se loguer
```

```
{
$sql = "SELECT * FROM personnel WHERE ((login='$UserName') AND
(password='$UserPassword'));";
$res=odbc_exec($conn,$sql);

if(odbc_fetch_row($res))
{
?>
<html>
<!-- date de creation: 12/08/01 -->
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
</head>
<body>

bienvenue <?echo odbc_result($res,"prenom")." ".odbc_result
($res,"nom")?><br>
<?
$sql2 = "SELECT * FROM personnel;";
$res2=odbc_exec($conn,$sql2);
?>
insérer un nouveau membre :<br><br>
<form action="inserer.php" method=post>
<table bgcolor="silver">
<tr>
<td>Nom :</td>
<td><input type="text" name="nom" size=40 maxlength=40>
</td>
<td>Prénom :</td>
<td><input type="text" name="prenom" size=40 maxlength=40></td>
</tr>
<tr>
<td>Age :</td>
<td><input type="text" name="age" size=40 maxlength=40></td>
<td>Ville :</td>
<td><input type="text" name="ville" size=40 maxlength=40></td>
</tr>
<tr>
```

```
<td>service :</td>
<td><input type="text" name="service" size=40 maxlength=40></td>
<td colspan="2" align="center"><input type="submit" value="insérer"></td>
</tr>
</table>

</form>
<hr>
membres du personnel :<br><br>
<table border="1">
<tr>
<td>Nom </td>
<td>Prénom</td>
<td>Ville</td>
<td>âge</td>
<td></td>
<td></td>
<td></td>
</tr>
<?
while(odbc_fetch_row($res2)) //création d'un objet permettant
// d'accéder aux différents champs d'une ligne de données
{
?>
<tr>
<td><?echo odbc_result($res2,"nom")?></td>
<td><?echo odbc_result($res2,"prenom")?></td>
<td><?echo odbc_result($res2,"ville")?></td>
<td><?echo odbc_result($res2,"age")?></td>
<?
if ((odbc_result($res2,"login")!="carlier")&&(odbc_result($res2,"login")!
="sidir"))
{
?>
<td><a href="modifier.php?id=<?echo odbc_result($res2,"id_pers");?
>">modifier</a></td>
<td><a href="supprimer.php?id=<?echo odbc_result($res2,"id_pers");?>"
onclick="alert('Pour être rigoureux il faudrait demander confirmation à l
\'utilisateur avant de supprimer');">supprimer</a></td>
<?
```

```
}
else
{
?>
<td></td>
<td></td>
<?
}
?>
</tr>
<?
}
?>
</table>
</body>
</html>
<?
}
else
{
?>
<html>
<!-- date de creation: 12/08/01 -->
<head>
<title></title>
<SCRIPT LANGUAGE="JAVASCRIPT">
function verif() {
if (formul1.UserName.value == "") {
alert ("Veuillez rentrer votre nom d'utilisateur");
return false;
}
if (formul1.UserPassword.value == "") {
alert ("Veuillez rentrer votre mot de passe");
return false;
}
return true;
}
</SCRIPT>
</head>
<body>
```

```
<center>
erreur d'identification<br>
<form method="POST" name="formul1" action="identification.php"
ONSUBMIT="return verif();">
<center>
<table border="0" width="253" cellspacing="2" cellpadding="4"
height="111" bgcolor="#FF7D07">
<tr>

<td align="right" valign="bottom" bgcolor="#FF7D07"><font face="Arial"
size="2" color="white">Login</font></td>
<td valign="bottom" bgcolor="#FF7D07"> <input type="text"
name="UserName" size="14"></td>
</tr>
<tr>
<td align="right" valign="middle" bgcolor="#FF7D07"><font face="Arial"
size="2" color="#FFFFFF">Mot de passe</font></td>
<td valign="middle" bgcolor="#FF7D07"> <input type="password"
name="UserPassword" size="14"></td>
</tr>
<tr>
<td bgcolor="#FF7D07"></td>
<td>
<center>
<input type="submit" value="Connexion" name="B1">
</center>
</td>
</tr>
</table>


</center>
</form>
</center>
</body>
</html>
<?
}
}
else
{
```

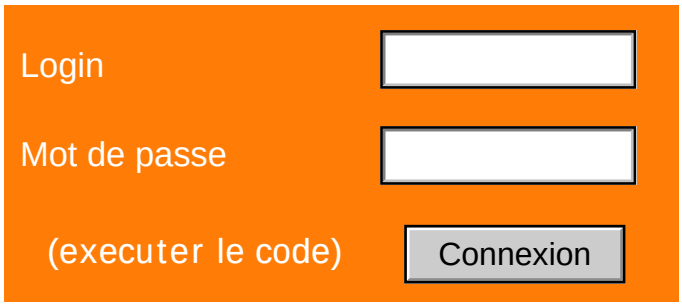
```
?>
<html>
<!-- date de creation: 12/08/01 -->
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
</head>
<body>
<?
$sql2 = "SELECT * FROM personnel;";
$res2=odbc_exec($conn,$sql2);
?>
insérer un nouveau membre :<br><br>

<form action="inserer.php" method=post>
<table bgcolor="silver">
<tr>
<td>Nom :</td>
<td><input type="text" name="nom" size=40 maxlength=40>
</td>
<td>Prénom :</td>
<td><input type="text" name="prenom" size=40 maxlength=40></td>
</tr>
<tr>
<td>Age :</td>
<td><input type="text" name="age" size=40 maxlength=40></td>
<td>Ville :</td>
<td><input type="text" name="ville" size=40 maxlength=40></td>
</tr>
<tr>
<td>service :</td>
<td><input type="text" name="service" size=40 maxlength=40></td>
<td colspan="2" align="center"><input type="submit" value="insérer"></td>
</tr>
</table>

</form>
<hr>
membres du personnel :<br><br>
```

```
<table border="1">
<tr>
<td>Nom </td>
<td>Prénom</td>
<td>Ville</td>
<td>âge</td>
<td></td>
<td></td>
<td></td>
</tr>
<?
while(odbc_fetch_row($res2)) //création d'un objet permettant
// d'accéder aux différents champs d'une ligne de données
{
?>
<tr>
<td><?echo odbc_result($res2,"nom")?></td>
<td><?echo odbc_result($res2,"prenom")?></td>
<td><?echo odbc_result($res2,"ville")?></td>
<td><?echo odbc_result($res2,"age")?></td>
<?
if ((odbc_result($res2,"login")!="carlier")&&(odbc_result($res2,"login")!
="sidir"))
{
?>
<td><a href="modifier.php?id=<?echo odbc_result($res2,"id_pers");?
>">modifier</a></td>
<td><a href="supprimer.php?id=<?echo odbc_result($res2,"id_pers");?>"
onclick="alert('Pour être rigoureux il faudrait demander confirmation à l
\'utilisateur avant de supprimer');">supprimer</a></td>
<?
}
else
{
?>
<td></td>
<td></td>
<?
}
?>
```

```
</tr>
<?
}
?>
</table>
</body>
</html>
<?
}
?>
```

le paramètre param est présent car cette page est accessible au moment de la connexion (dans ce cas le paramètre est indéfini), après une insertion, une modification ou une suppression, ceci afin que les données dynamiques présentes dans cette page soient bien actualisée par le navigateur, car les fonctions d'actualisation présentes en début de fichier ne fonctionnent pas toujours bien à cause du cache de la machine. et même ce paramètre param n'est pas la meilleur solution car il ne varie pas, d'où un risque minime d'interférence du cache. La meilleure solution, tout en utilisant ce paramètre, est de le faire varier à chaque affichage de la page, par exemple en envoyant comme paramètre l'heure courante en heures minutes secondes. Dans ce cas l'actualisation de la page se fera comme il faut.

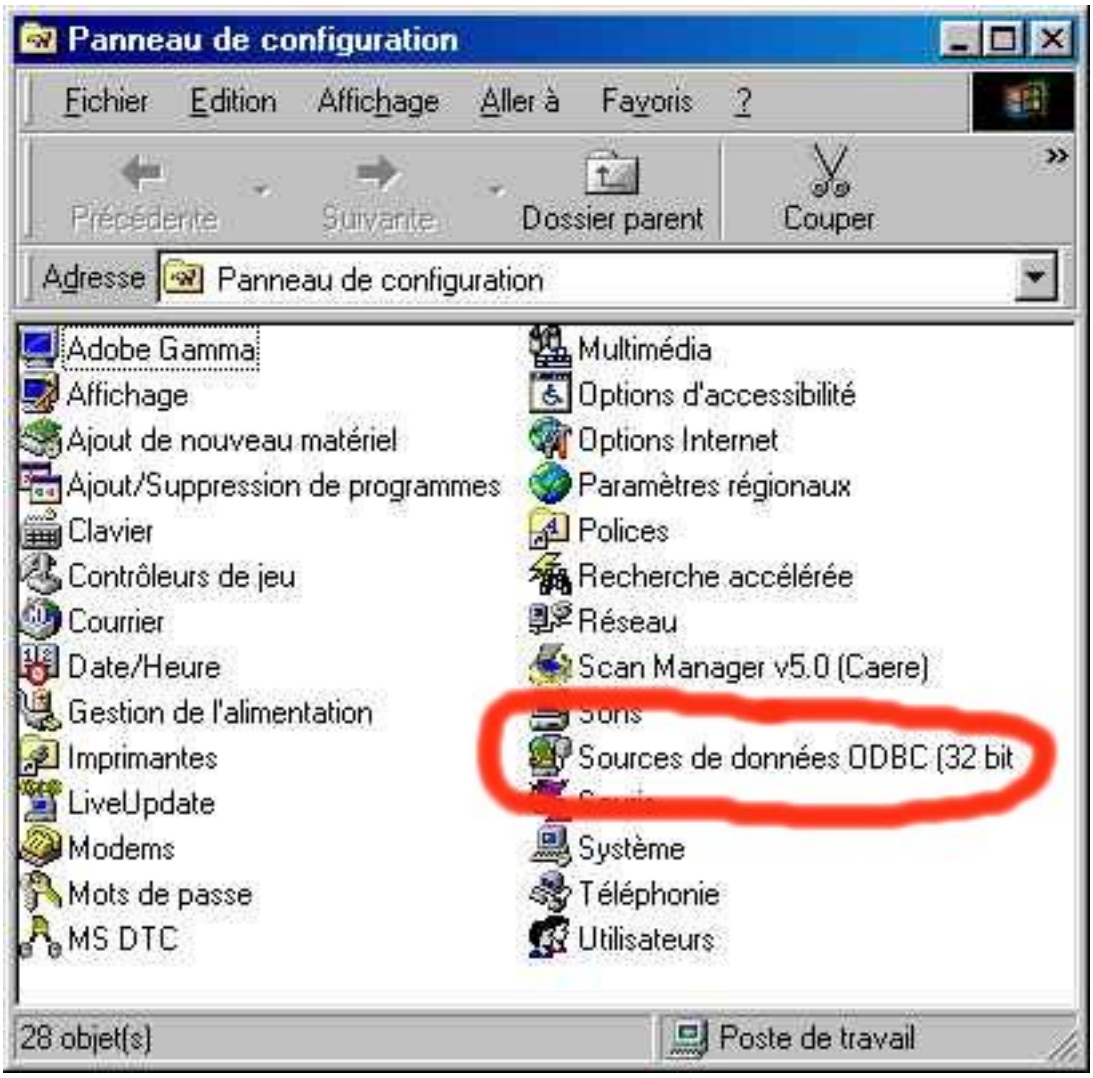
Pour l'exemple, utilisez en login : **carlier** et en mot de passe : **carlierg**



# Créer le lien de la base dans la couche ODBC

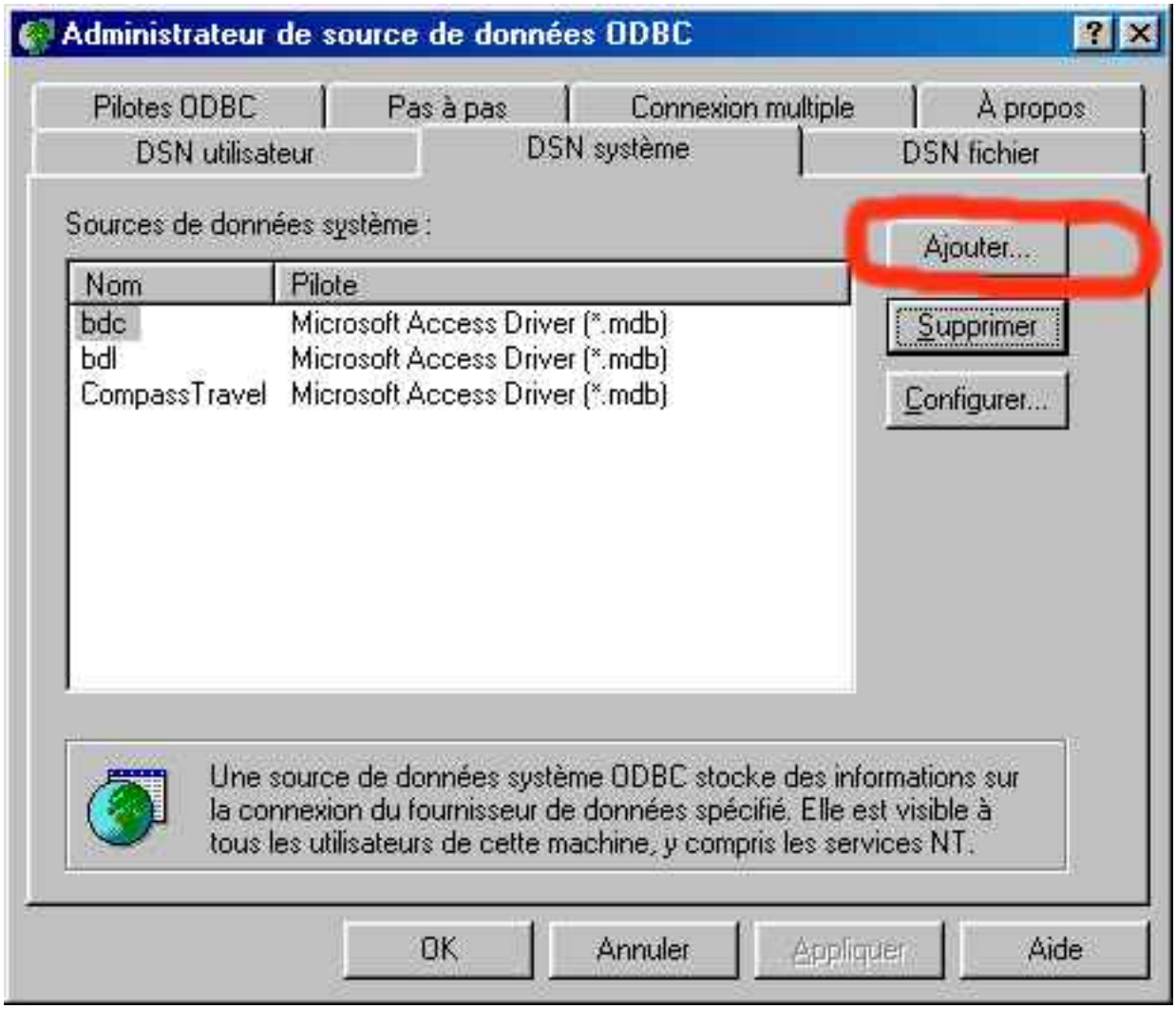
Pour accéder à une base Microsoft Access via PHP ou ASP, il faut la déclarer sous un alias dans la couche ODBC.

Pour ce faire, aller dans le Panneau de configuration (menu Démarrer - Paramètres - Panneau de configuration)

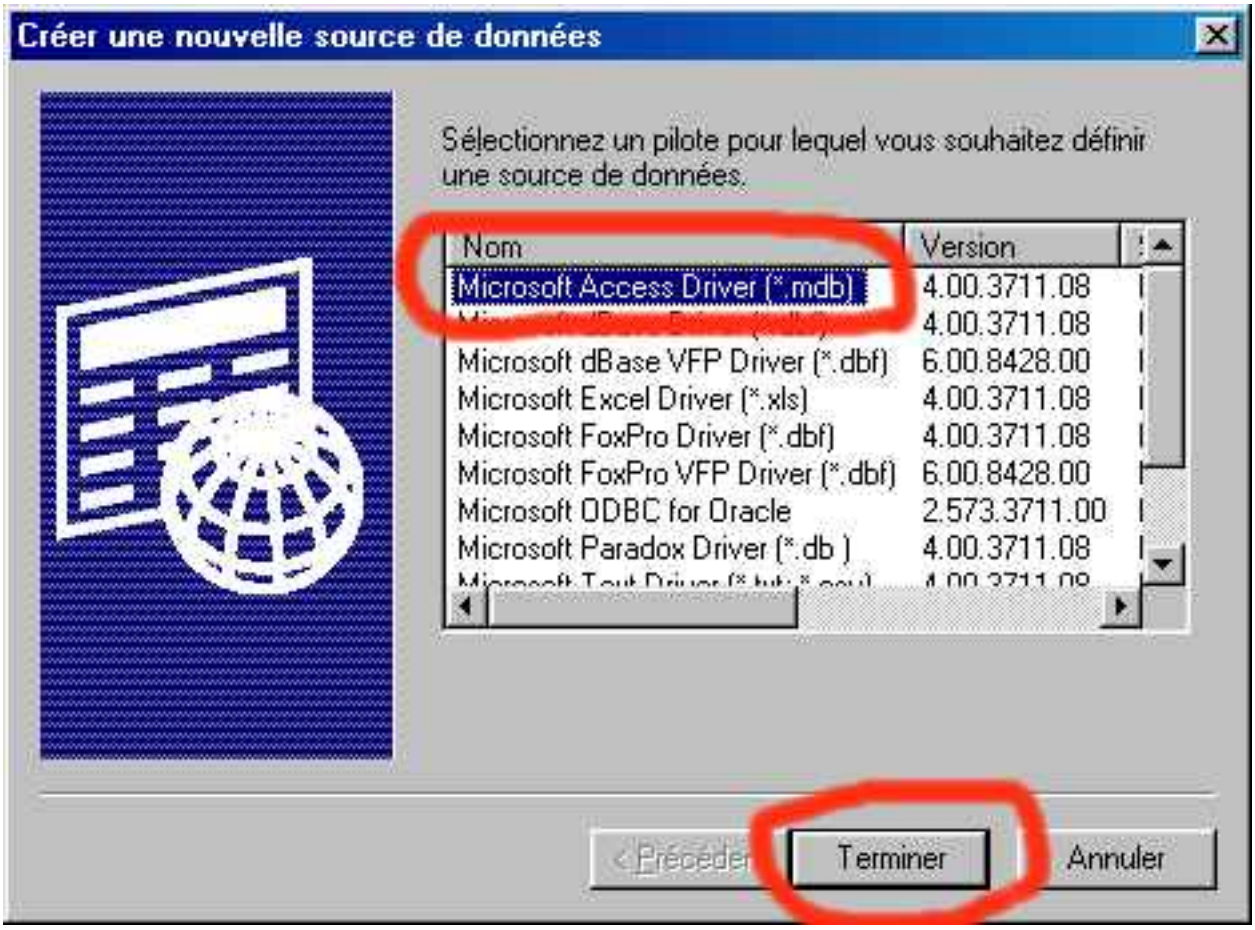



Cliquez sur sources de données ODBC

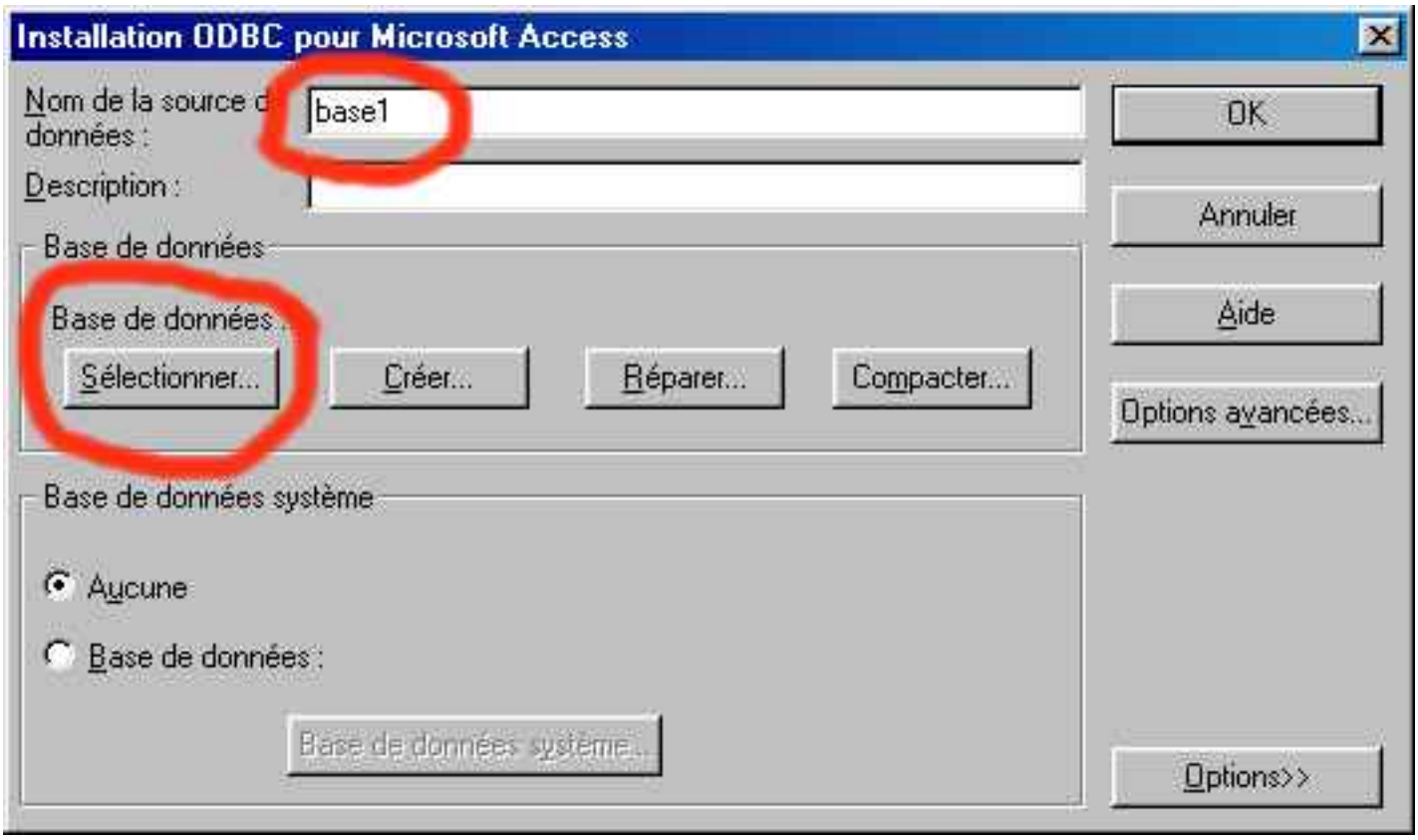

Cliquez sur DSN système




Cliquez sur Ajouter




sélectionner le driver Microsoft Access et cliquez sur terminer



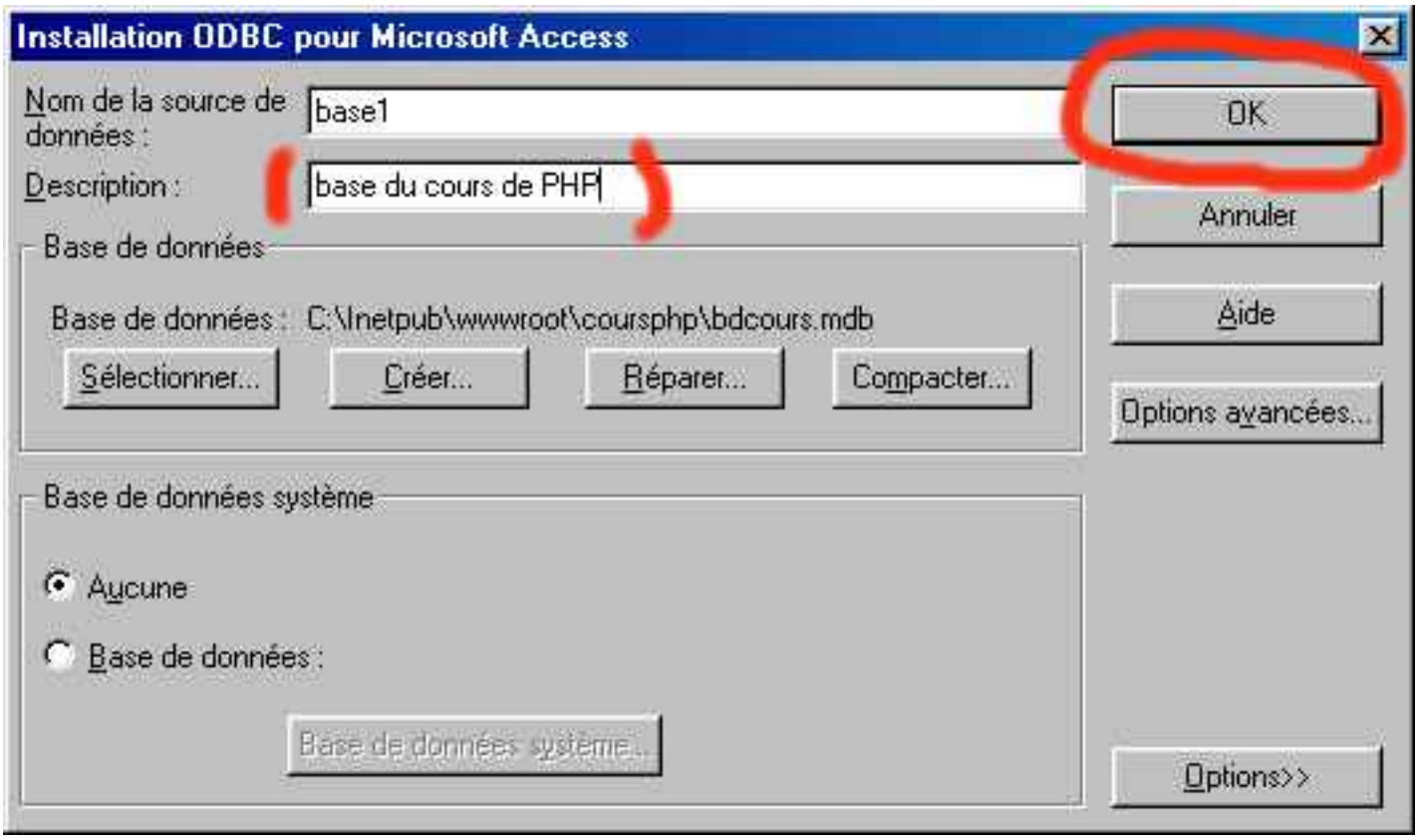

donnez un alias à la base (le nom utilisé dans les fichiers php pour faire référence à la base)

et sélectionnez la base à laquelle l'alias fait référence


parcourez les répertoires jusqu'à celui contenant la base

sélectionnez la boas

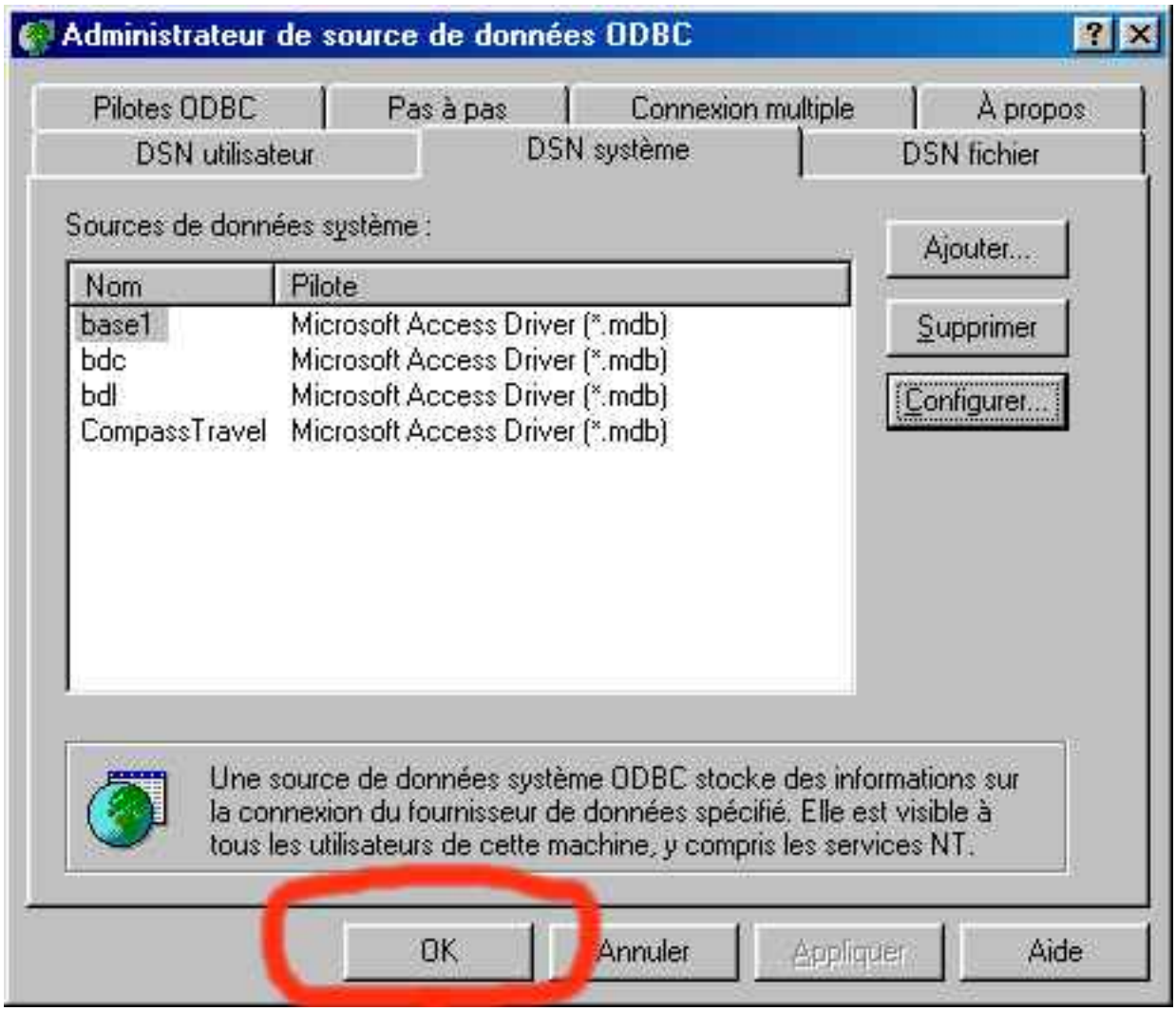
cliquez sur OK




vous pouvez si vous le souhaitez donner une description de la base, ceci afin de déduire

instantanément à quelle base correspond quel alias si vous avez de nombreuses bases.

Cliquez sur OK




Cliquez une dernière fois sur OK

L'alias est déclaré, vous pouvez maintenant utiliser vos scripts sur la base de données.

# Les tables de la base de données crée pour l'exemple.

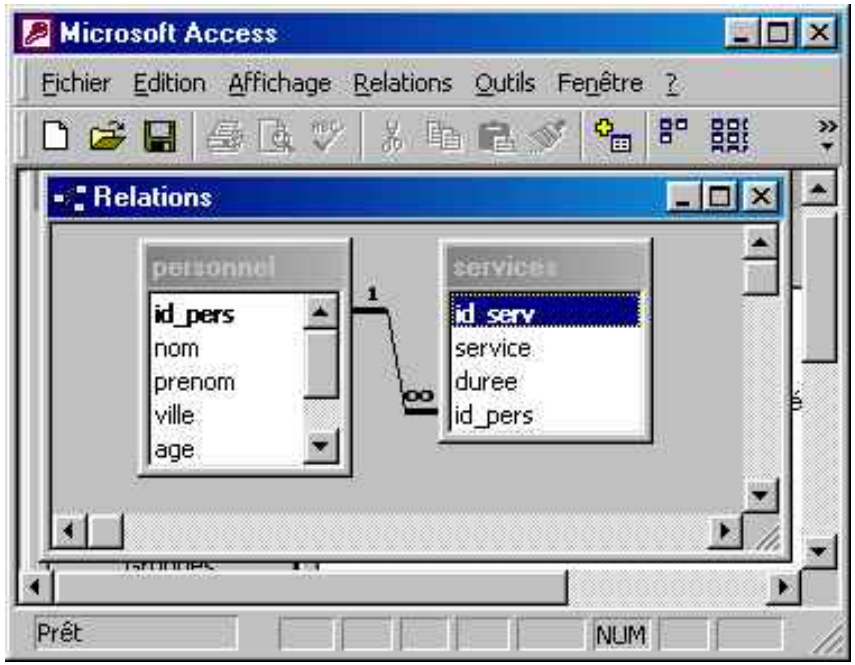

Microsoft Access
Fichier Edition Affichage Insertion Outils Fenêtre ?
personnel : Table
Nom du champ
Type de données
id_pers
NuméroAuto
nom
Texte
prenom
Texte
ville
Texte
age
Numérique
login
Texte
password
Texte
Propriétés du champ
Général
Liste de choix
Taille du champ
50
Format
Masque de saisie
Légende
Valeur par défaut
Valide si
Message si erreur
Null interdit
Non
Chaîne vide autorisée
Oui
Indexé
Non
Compression unicode
Oui
Mode Création
NUM


Microsoft Access
Fichier Edition Affichage Insertion Outils Fenêtre ?
services : Table
Nom du champ
Type de données
id_serv
NuméroAuto
service
Texte
duree
Numérique
id_pers
Numérique
Propriétés du champ
Général
Liste de choix
Taille du champ
50
Format
Masque de saisie
Légende
Valeur par défaut
Valide si
Message si erreur
Null interdit
Non
Chaîne vide autorisée
Oui
Indexé
Non
Compression unicode
Oui
Mode Créat
NUM

déclarations des clés étrangères :

cliquer sur 

dans la barre d'îcones

<? while(odbc_fetch_row($res)) //création d'un objet permettant // d'accéder aux différents champs d'une ligne de données { ?>

| Nom | Prénom | Ville | âge | dirige le service | heures par semaine |
|---|---|---|---|---|---|

# Technologies de Script

## Accès aux SGBD MySQL avec PHP

---

Objectifs


1. Introduction
2. Connexion
3. Mise à jours
4. Affichage du résultat
5. Suite...


---

Mr Mohamed SIDIR

# 1 - Introduction

---

Un des très grand avantage de PHP, est sans aucun doute l'accès facile à différentes bases de données telles que Oracle, Sybase, PostgreSQL ou MySQL que nous allons utiliser dans la suite de ce cours.

L'interface fonctionnelle MySQL/PHP est un ensemble des fonctions qui permettent de se connecter à MySQL, d'exécuter des requêtes SQL et récupérer le résultat que l'on peut ensuite afficher dans une page HTML.

Le tableau suivant donne la liste des principales fonctions utiliser pour interfacer une base de données Mysql avec de php.

| Fonctions | Description |
|---|---|
| mysql_connect | ouvre une connexion à un serveur MySQL pour un compte utilisateur et un serveur donné. Elle revoit une valeur qui peut être utilisée pour dialoguer avec le serveur |
| mysql_pconnect | Ouvre une connexion persistante à un serveur MySQL. Très utiliser lorsque le serveur php et inclus dans Apache |
| mysql_select_db | Sélectionne une base de données MySQL |
| mysql_query | Envoie une requête SQL au serveur MySQL sous forme d'une variable |
| mysql_fetch_object | Retourne une ligne de résultat sous la forme d'un objet. Elle positionne le curseur sur la ligne suivante |
| mysql_num_rows | Retourne le nombre de ligne d'un résultat |
| mysql_fetch_row | Récupère une des lignes du résultat, et la retourne sous la forme d'un tableau |
| mysql_drop_db | Efface une base de données MySQL |
| mysql_close | Fermer la connexion MySQL |

| | |
|---|---|
| mysql_error | Retourne le texte associée avec l'erreur générée lors de la dernière requête |

# 2 - Connexion

---

Avant de faire le moindre accès à une base de données, il faut impérativement établir une connexion au serveur de base de données avec un compte utilisateur et un mot de passe correspondant, et choisir sa base de données puisque ce serveur peut, éventuellement, donner accès à plusieurs bases. Nous utiliserons la base de données Base1 créée dans le chapitre précédent :

```
<?php

$base_de_donnees = "basedoccours"; //nom de la base de données
$serveur="localhost"; //adresse du serveur. Ici localhost est le serveur local. Mais en général, on utilise l'adresse IP ou le nom de la Hote
$nom="root"; //nom de l'utilisateur ayant droit sur la base de données, ici c'est l'administrateur général root
$mot_de_passe=""; //mot de passe de cet utilisateur

// connexion au serveur de base de données (mysql)
$connexion = mysql_connect ($serveur, $nom, $mot_de_passe);

//sélection de la base de données à utiliser pour les requête
mysql_select_db ($base_de_donnees);

.....

mysql_close(); //  fin de la connexion
?>
```

Dans l'exemple ci dessous on utilise une forme plus contractée des fonctions php/MySQL:

```
<?php

$Connexion = mysql_connect
("localhost", "root", "");

if ($Connexion) echo "Connexion au
serveur est réussie";

else echo "Désolé la connexion au
serveur est impossible";

$connexionbase = mysql_select_db
("basedoccours");

if ($connexionbase) echo "Connexion à
la base de données est réussie";
else echo "Désolé, la connexion à la
base est impossible";

....
mysql_close();

?>
```

NB. mysql_select_db() renvoie un booléen utile pour savoir si la connexion a été réussie

# 3 - Mise à jours d'une base MySQL

---

L'interaction avec un site dynamique implique la possibilité d'effectuer des mises à jours sur la base de données. un exemple est l'inscription d'un visiteur afin de lui accorder un droit d'utilisation de votre site ou une partie de site. L'utilisation d'un formulaire est la méthode normale de saisie des valeurs à insérer dans la base de données.

## Le formulaire

On utilise le formulaire HTML : (saisie.htm)

```
<html>

<body>
<form action="inserer.php"
method="post">

Message : <input type="text"
name="contenu" size="90"
maxlength="255">

<input type="submit" value="insérer">
<input type="reset" value="rétablir">

</form>

</body>
</html>
```

## Le script PHP

Comme indiquer dans l'attribut **action** de formulaire (saisie.htm), le script php d'insertion des données est contenu dans le fichier (inserer.php)

```
<?php

$base_de_donnees="base1"; //nom de la base de données
$serveur="localhost"; //adresse du serveur

$nom="root"; //nom de l'utilisateur qui a des droits sur la base de données
$mot_de_passe=""; //mot de passe de cet utilisateur

// connexion au serveur de base de données (mysql)
$connexion = mysql_connect ($serveur, $nom, $mot_de_passe);

//sélection de la base de données à utiliser pour les requête
mysql_select_db ($base_de_donnees);

//requête sql d'insertion du document
$requete="insert into messages (contenu) ";
$requete .= "values ('".$contenu."');";

// exécution de la requête : on envoie la requête au serveur qui nous retourne le résultat
if ($resultat = mysql_query ($requete))
echo "l'insertion s'est bien déroulée";
else
echo mysql_error();

?>
```

tester l'exemple

# 4 - Affichage du résultat

Si la requête réussit, il ne reste plus qu'à récupérer le résultat et les afficher grâce au script suivant :

```
<?php
header("Pragma:no-cache");
$base_de_donnees="base1"; //nom de la base de données
$serveur="localhost"; //adresse du serveur

$nom="root"; //nom de l'utilisateur qui a des droits sur la base de données
$mot_de_passe=""; //mot de passe de cet utilisateur

// connexion au serveur de base de données (mysql)
$connexion = mysql_connect ($serveur, $nom, $mot_de_passe);

//sélection de la base de données à utiliser pour les requête
mysql_select_db ($base_de_donnees);

//requête sql de sélection des messages
$requete="select * from messages;";

// exécution de la requête : on envoie la requête au serveur qui nous retourne le résultat
$resultat = mysql_query ($requete);

?>

<html>

<body>
<?php
//echo mysql_error(); si on souhaite déboguer la dernière requête exécutée, cette fonction permet de savoir l'erreur dans la requête
```

```
for ( $i = 0 ; $i < mysql_num_rows ($resultat) ; $i++)
{
// pour chaque ligne de résultat, on affiche les données
// mysql_num_rows ($resultat) retourne le nombre de ligne
de résultat de la requête

// positionnement sur la i ème ligne de résultat de la requête
mysql_data_seek ($resultat, $i);

// on stocke cette ligne dans un objet sous forme de pointeur
$ligne = mysql_fetch_object ($resultat);

echo "ligne de résultat numéro ".$i."<br>";
echo "Message : ".$ligne->contenu."<br><br>";
echo "<br><hr><br>";
}
?>

</body>
</html>
```

tester l'exemple

---

# 5 - Suite... :


Si vous avez bien compris "la philosophie" de l'interfaçage d'une base de données Mysql en utilisant les fonctions PHP, je vous invite à la séquence des travaux pratiques.

# Technologies de Script

## Le Système de Gestion de Base de Données MySQL

---

Objectifs


I. MySQL : Serveur et SGBG
II. Création d'une base de données avec MySQL
III. Autres commandes de MySQL
IV. Les fonctions mysql
V. Gestion des droits sous MySQL
VI. phpMyAdmin : Interface de gestion de MySQL


---


Concepteurs :

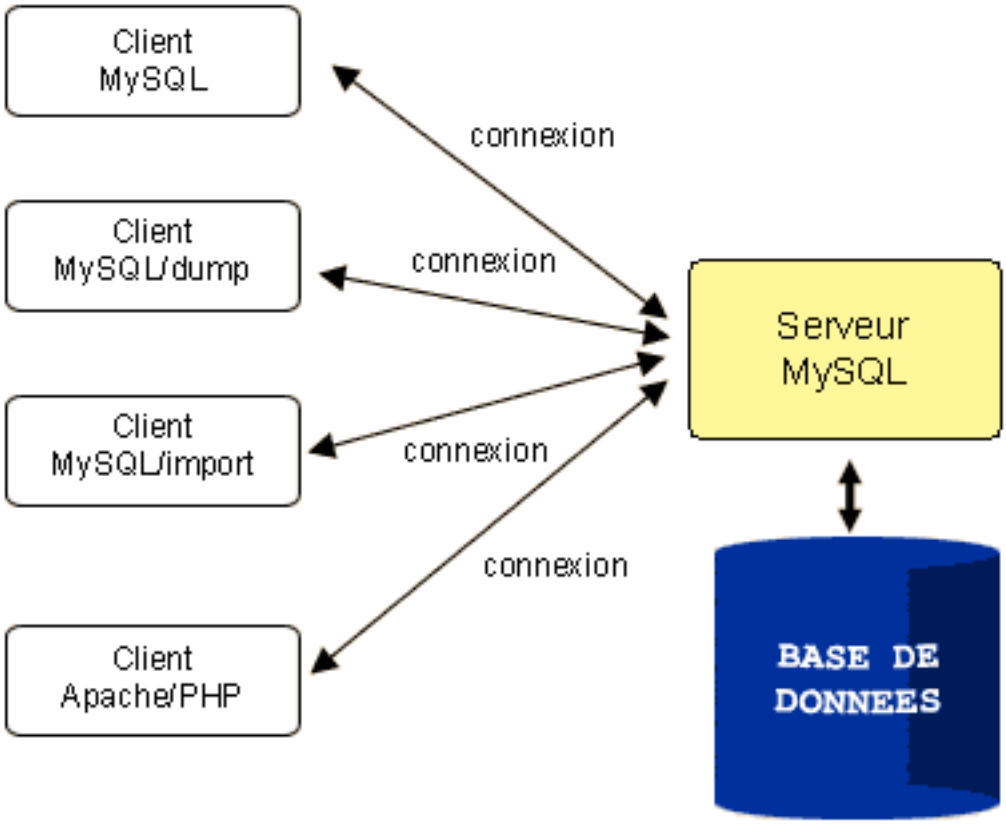
Mr Mohamed SIDIR

# I - MySQL : Serveur et SGBD

MySQL est un système de gestion et conception de base de données. Il est libre et gratuit. Il reste le serveur de base de données le plus apprecié des webmestres grâce à sa facilité, sa rapidité et sa compatibilité avec les systèmes d'exploitation les plus utilisés comme Windows et Unix/Linux.

Facile à installer, il s'accompagne de nombreux produits permettant sa gestion, tels que PhpMyAdmin, un driver MyODBC implémenté par défaut dans PHP.

MySQL est un véritable serveur de base de données SQL multi-utilisateur et multi-threaded. Il est consitué d'un ensemble de programmes chargés de gérer une ou plusieurs bases de données, et qui fonctionnent selon une architecture client/serveur (image ci-dessous).




- **mysqld** est le serveur de MySQL, il est le seul à accéder aux fichiers stockant les données pour lire et écrire des informations. Les autres utilitaires sont chargés de dialoguer ?avec lui? ;les plus utiliser sont :
- **mysqldump** qui permet d'effectuer des sauvegardes
- **mysqlimport** pour importer des fichiers ASCII dans une base de données

- **mysql** est le plus utile, il permet d'envoyer directement des commandes au serveur

Les cliens de MySQL communiquent avec le serveur pour effectuer des recherches ou actualiser une base de données.

MySql est une base de données relationnelle, elle répresente donc les informations sous forme de tables. Comme toute base de données relationnelles. MySQL utilise le langage SQL pour interroger ou mettre à jour les données. SQL est un langage déclaratif qui permet de traiter les données sans se soucier de la répresentation interne de ces données, de leur location ?, des chemins d'accès ou des algorithmes nécessaires.

---

# II - Création d'une base de données avec MySQL

Pour tous nos travaux pratiques, MySQL sera installé en utilisant easyphp qui installe en même temps le serveur web apache et le gestionnaire de script php. Mais on ne peut installer Mysql qu'en télechargeant sur le site officiel www.mysql.com un package en format **tar.gz** ou **rpm** pour linux, ou un exécutable pour un système d'exploitation sous windows.

## 1. Installation de MySQL sous Windows :

Selon les versions de Windows, mysql s'installe comme un service et s'active automatiquement à chaque démarrage de l'ordinateur (comme sous Linux). Pour installer MySQL :

Permière Méthode :

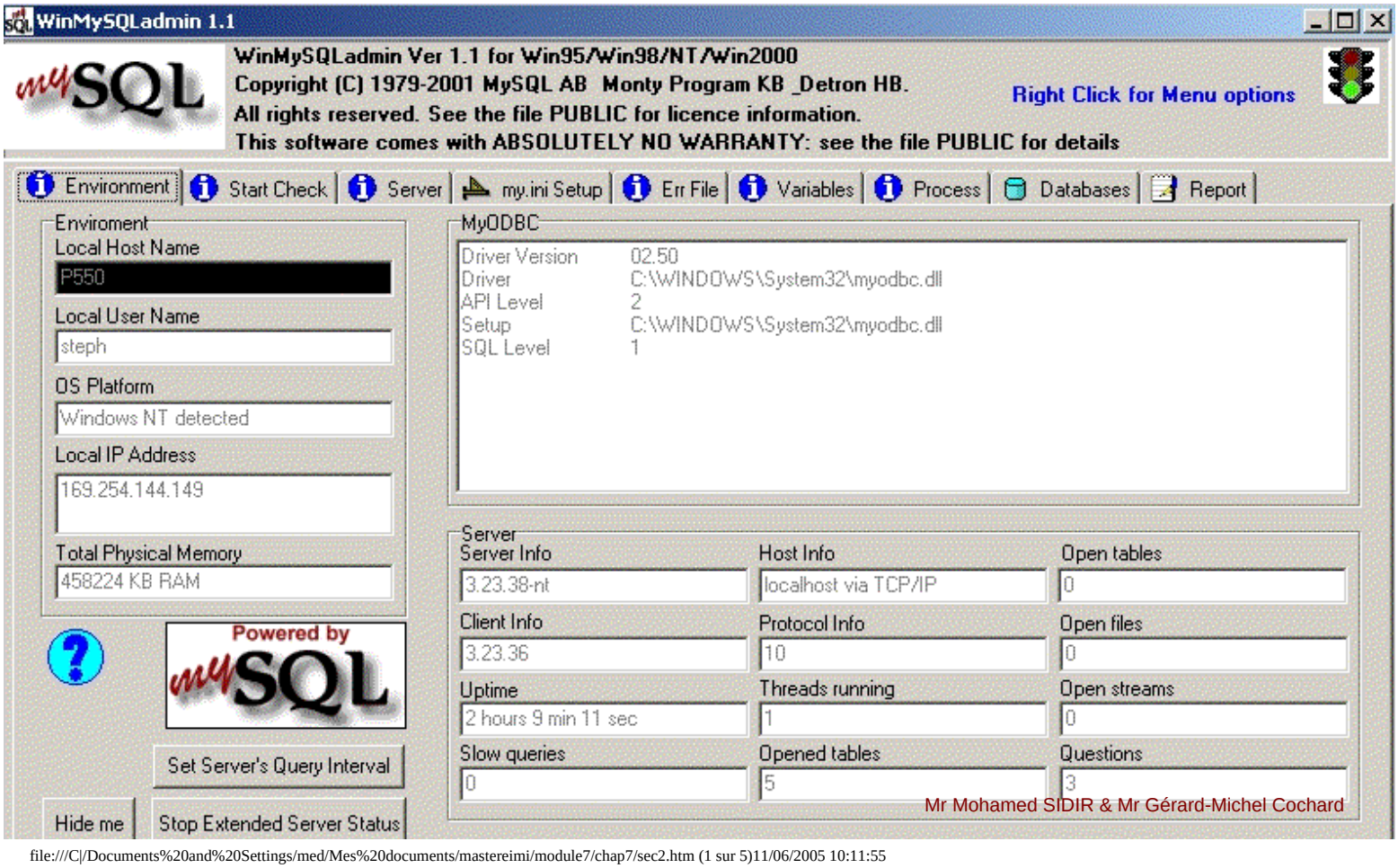
- ouvrir une fenêtre de commande de type dos,
- dans le répertoire d'installation de mysql (bien souvent c:\mysql), puis dans le sous-repertoire [bin] et exécuter la commande : **msqld-nt.exe --install**
  suivi de sa déclaration comme service : **net start mysql**
- MySQL s'installe ...

Deuxième Méthode :

Il est plus simple d'utiliser le gestionnaire d'installation **winmysqladmin**, fourni avec mysql, qui se trouve dans le sous-répertoire [bin] de mysql. IL se charge d'installer et de déclarer convenablement MySQL pour votre système .

Ce procédé évite de taper les commandes de la première méthode.

**NB**. l'installation de mysql sous windows 2000 est bien souvent problèmatique, sauf par le biais du programme **winMysqlAdmin**.




**Quelques informations :**

LocalHost Name : nom de la machine

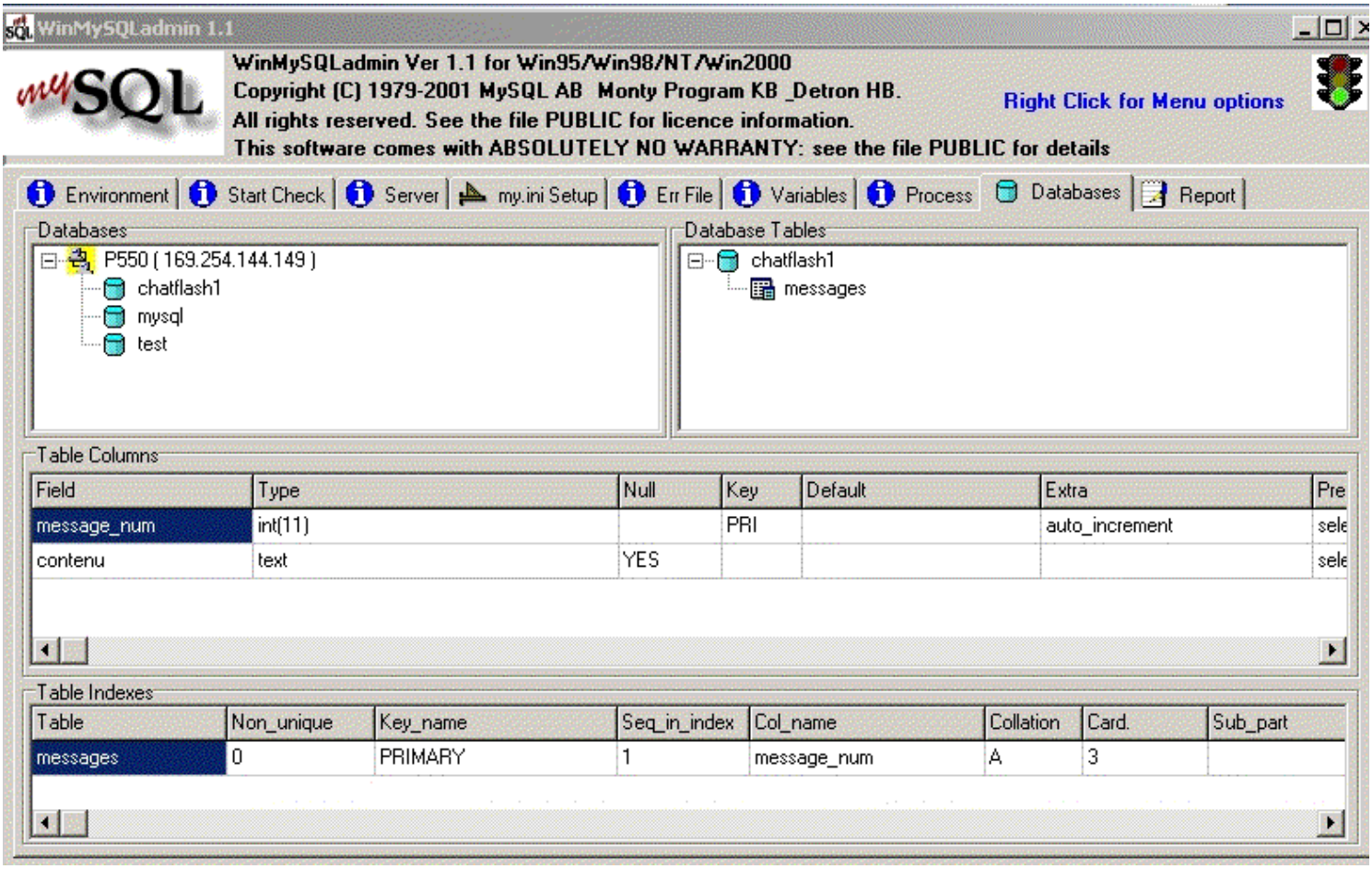
Local User Name : Nom d'un utlisateur ?

OS Platform : système d'exlpoitation détécté sur la machine

LocalIP adresse : L'adresse IP de la machine

Mysql est maintenant installé correctement.

En cliquant sur l'onglet **Databases**, WinMysqlAdmin permet de montrer la structure des tables, mais il ne gére ni les tables et ni les données.




Dans ce cas, la gestion de base de données (SGBD) se gère en ligne de commandes sql, en lançant la console d'interprétation des commandes SQL.

On peut accèder à la console en ligne de commande en lançant le programme mysql.exe, situé dans le répertoire [bin] où sont situés tous les programmes en rapport avec mysql.

Méthode :

```
C:\>cd mysql

C:\mysql>cd bin

C:\mysql\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.23.38-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql>
```

Le système est, maintenant, prêt à créer des bases de données.

**Attention :** Toutes les commandes sql doivent se terminer par un point virgule.

## 2. Une première base MySQL

Une fois que l'on a établi sur le papier les relations normalisées de la Base de Données, on dispose de renseignements nécessaires à la création de la base.

On pourrait créer et gérer directement une base par un utilitaire de type **phpmyadmin**, mais il est intéressant de voir la manière d'implémenter les requêtes en lignes de commande d'insertion d'images

On crée une simple base de données nommée **base1**, constituée de deux tables **messages** et **messages_consultation**

**1ére Etape** : création d'une base de données

syntaxe : **create database nom_de_la_base;**

```
C:\mysql\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 3.23.38-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> create database base1;
Query OK, 1 row affected (0.00 sec)
```

Pour manipuler une base de données (créer des tables, exécuter des réquettes,..), on utilise la commande **use nom_de_la_base;**

```
mysql> use base1;
Database changed
mysql>
```

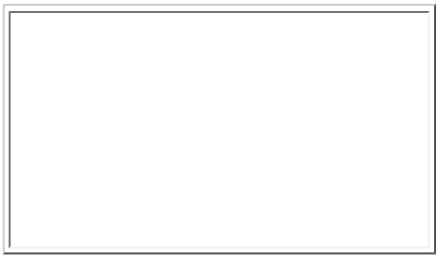
La commande **use base1**, indique que l'on s'apprête à travailler sur la base **base1**

La commande : **drop database nom_de_la_base;** permet de supprimer **définitivement** la base de données. Pour détruire une table, on dispose de la commande **drop table nom_de_la_base;**

**2émé Etape** : création des tables attachées à la base de données **base1**.

Pour créer des tables, on utilise une partie de SQL dite "Langage de Définition de Données" (DDL) dont la commande principale est le **CREATE TABLE.**

**Création de la table messages**



**CREATE TABLE** : commande de création d'une table, il indique le nom de la table puis la liste des attributs avec leur type. Pour l'instant, on n'ulisera que quelques types de base : INTEGER, que l'on peut abréger en **int**, est un entier et **text** pour un texte multiligne.

**Not NULL** signifie que le champ nommé **message_num** doit obligatoirement contenir une valeur,

**auto-increment** permet d'attribuer une valeur automatique et unique à chaque enregistrement du champs **message_num**.

**PRIMARY KEY** définit la clé primaire, comme dans la syntaxe classique du SQL. La clé primaire étant le moyen d'identifier une table.

La table **messages** est maintenant créee, on peut consulter son schéma avec la commande DESCRIBLE, **DESC** en abrégé, et obtenir l'affichage ci dessous :

ssssssssssssssssssssssssssssssssssssss

**Création de la table messages_consultation**

**Remarque** : La déclaration des clés étrangères qui permet de gérer les relations entre les tables, est possible par le même principe que PRIMARY KEY :
le syntaxe : **FOREIGN KEY (nom_du_champs) REFERENCES table_d_origine** où **nom_du_champs** est la clé primaire de la **table_d_origine**.

MySQL n'a pas encore implémenté le concept de la clé étrangère et n'en tient donc pas compte. Cela ne saurait tarder (peut-être implémenté à l'instant où vous lisez ces lignes).
La meilleure solution actuellement est de déclarer les clés étrangères même si elles ne sont pas reconnues, de manière à ne pas avoir à les saisir à nouveau lorsque mysql les implémentera correctement.

Rappellons les principaux types de données disponibles dans mysql.

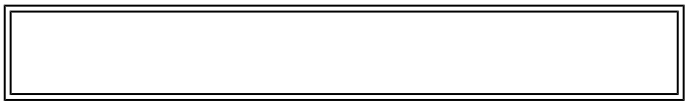
| Type | Description |
|---|---|
| int | entier (de taille 11 par défaut, utilisé notamment pour les clés primaires auto-incrémentables) |

| | |
|---|---|
| char | chaîne de caractères de longueur fixe |
| varchar | chaîne de caractères de longueur variable (symbolisée par un pointeur, espace mémoire moindre par rapport au char). longueur max de 255 |
| datetime | date et heure |
| text | texte multiligne |
| longtext | comme texte, mais acceptant une longueur de texte beaucoup plus importante |
| float | réel |
| tinyint | entier court (4) |

3émè étape : insertion des données

Pour insérer des données, On utilise la commande **insert** qui fonctionne comme en SQL classique.

a) insertion complète :

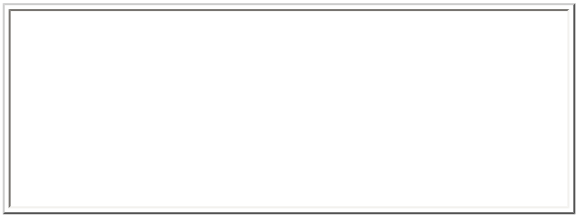


NB. le **message_num** n'est pas nécessaire car la valeur est **auto_increment**, donc il est préfèrable de se contenter de la commande suivante :

b) insertion sans préciser les arguments ni l'ordre

mysql suppose que tous les arguments soient présents et les prennent en compte dans l'ordre de création dans la table.

c) insertions étendues :



une manière d'insérer plusieurs enregistrements "d'un seul coup" sans préciser à chaque fois les arguments à prendre en compte ni leur ordre.

---

# III- Autres commandes de MySQL

---

a) mysql_dump : sauvegarde d'une base Mysql

Pour sauvegarder une base de données sous un autre nom , on peut faire "un dump" avec la commande suivante :

```
mysql_dump.exe nom_de_la_base > nom_de_fichier_de_sauvegarde
```

cela permet par exemple de réutiliser un fichier de dump de la base de données après une erreur de manipulation, ou utiliser une copie de la base.

**Attention**, sous windows, les tables sont des répertoires, la casse (majuscules / minuscules) n'a pas d'importance lors de la création et de la réutilisation de la base dans des scripts. Le problème se pose sous Linux, où la casse a de l'importance. Si vous souhaitez utiliser la même structure de base de données pour un serveur Windows et ensuite l'exporter sous Linux, il y aura une incompatibilité dans vos scripts. Faites donc attention dès la conception de la base de données, ne mettez pas de majuscules dans les noms de tables.

A l'aide de ces commandes, il est possible de programmer le serveur pour faire des sauvegardes quotidiennes des bases de données par exemple en utlisant un script batch (.bat) sous windows pour sauvegarder une base sous un nom différent chaque jour :

ce fichier est à lancer dans le gestionnaire des tâches à des dates / heures précises.

on peut faire de même avec des commandes bourne-shell sous linux associé à l'anacrontab comme gestionnaire de tâches.

### b) Modification de la structure d'une table

Les commandes de modification des tables fonctionnent comme en SQL classique. Un exemple :

```
ALTER TABLE messages MODIFY contenu LONGTEXT NULL;
```

## c) Supprimer ou actualiser les données

- Le langage SQL propose des commandes de mise à jour et de suppression qui sont des variantes du SELECT. On utilise la même clause WHERE, en remplaçant dans un cas le SELECT par UPDATE, et dans l'autre cas par DELETE exemple :

Les données détruites son vraiment perdues. Lorsque l'on a l'habitude d'un système gérant les transactions, on garde en mémoire qu'il n'y a pas de possiblité de retour en arrière avec rollback dans MySQL.

- Pour faire des modifications, on utilise la commande suivante :

la ligne dont le champs **message_num = 1** est modifiée, sans possiblité d'annuler cette modification.

- La grande différence avec le langage SQL classique est que les commandes de sélection (SELECT) ne peuvent pas être imbriquées.

**exemple : SELECT * FROM messages;** fonctionne parfaitement mais **SELECT * FROM messages where message_num in (SELECT messages_num from autre_table);** ne fonctionne pas sous mysql

Avec des jointures, il est possible de se passer des select imbriqués. Un exemple qui fonctionne :

**SELECT * FROM messages,autre_table where message.message_num = autre_table.message_num;**

Il est aussi possible de réaliser des SELECT sur des valeurs nulles :

**SELECT * FROM messages where contenu IS NULL;** va sélectionner les messages vides (IS NOT NULL réalise l'opération inverse)

- D'autres commandes :

**show databases;** établit la liste des base de données disponibles sur le serveur (du moins celles dont vous avez le droit d'accès)

**show tables;** établit la liste des tables de la base de données en cours d'utilisation

```
mysql> show databases;
+------------+
| Database   |
+------------+
| base1      |
| chatflash1 |
| mysql      |
| test       |
+------------+
4 rows in set (0.00 sec)

mysql> use  base1;
Database changed
mysql> show tables;
+----------------------+
| Tables_in_ base1;    |
+----------------------+
| messages             |
+----------------------+
1 row in set (0.00 sec)

mysql> DESC messages;
+-------------+---------+------+-----+---------+----------------+
| Field       | Type    | Null | Key | Default | Extra          |
+-------------+---------+------+-----+---------+----------------+
| message_num | int(11) |      | PRI | NULL    | auto_increment |
| contenu     | text    | YES  |     | NULL    |                |
+-------------+---------+------+-----+---------+----------------+
2 rows in set (0.01 sec)

mysql> _
```

**SELECT LAST_INSERT_ID( );** retourne le numéro de clé de la dernière insertion réalisée (extrêmement utile pour réaliser des insertions dans plusieurs tables liées en terme relationnel (par les clés étrangères))

**INSERT INTO** messages **(**contenu**) VALUES ('bonjour tout le monde');**

**INSERT INTO** autre_table **(**message_num**,**truc**) VALUES (LAST_INSERT_ID(),'test');**

---

# IV - Les fonctions mysql

---

Sous MySQL, on dispose également de nombreuses fonctions mathématiques, de gestion des dates et des chaînes de caractères, ...

Ces fonctions forment l'une des forces de MySQL, mais étant données la quantité et la diversité de ces fonctions. on ne peut en retrouver une documention très complète que sur le site **www. nexen.net**

quelques fonctions :

| | |
|---|---|
| **ABS(nom_de_champ_numerique)** | retourne la valeur absolue du champ numérique. |
| **concat(chaine1,chaine2,...)** | concatène les chaines passées en arguments |
| **length(chaine)** | retourne la taille de la chaine passée en argument |
| **now()** | retourne la date et l'heure courante |
| **dayofyear(date)** | retourne le numéro du jour dans l'année (de 1 à 366) |
| .....ect | |

---

# V - Gestion des droits sous MySQL

---

Pour gérer les droits d'accès, il faut se connecter en temps que root (administrateur de bases).

La commande de création des droits est GRANT

**GRANT ALL PRIVILLEGES ON base1.* TO utilisateur@localhost IDENTIFIED by 'mot_de_passe';**

Cette commande donne tous les droits à l'utilisateur de login, appartenant au domaine **localhost** et ayant pour mot de passe **'mot_de_passe'**. Cette commande crée l'utilisateur s'il n'existe pas et lui affecte un mot de passe.

Il est possible de donner des droits plus restreints à l'utilisateur (cf: www.nexen.net).

Pour retirer un droit à un utilisateur, la syntaxe est la même que pour GRANT, mais le mot-clé est **REVOKE**.

---

# VI - phpMyAdmin : Interface de gestion de MySQL

---

PHPMyAdmin est une interface écrite en php permettant de gérer (même à distance) mysql, ceci de manière simplifiée, ce qui permet de ne pas écrire toutes les requêtes SQL à la main..

De plus, cet outil permet d'exporter le schéma de la base au format XML ou PDF et de faire des dumps de manière simple, ...

La plupart des commandes de l'utilitaire msql peuvent s'effectuer par l'intermédiare de phpMyAdmin. Les opérations possibles dépendent bien sûr des droits de l'utilisateur qui se connecte à la base

L'utlisateur a la possibilité de :

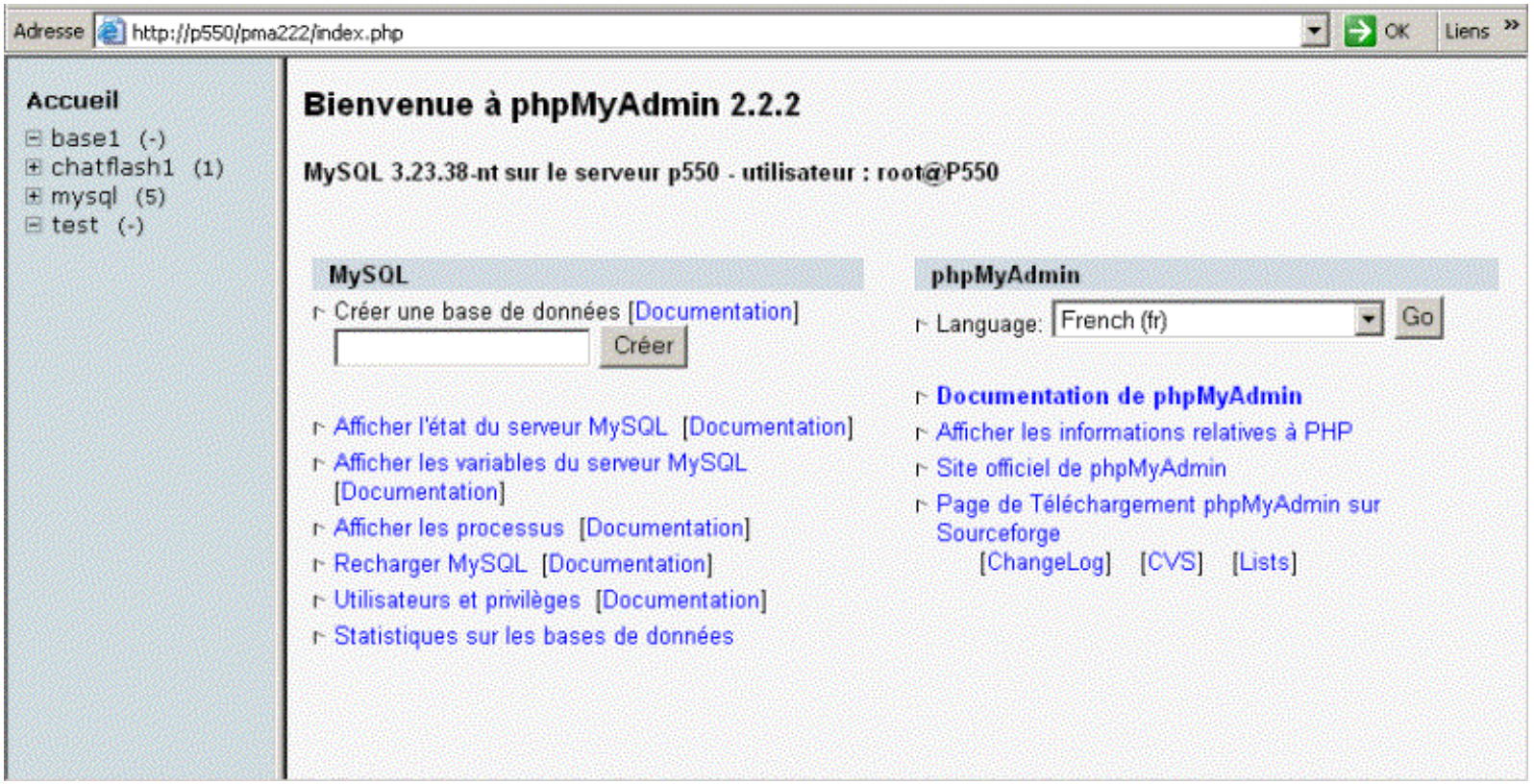
- créer, détruire et modifier des bases de données et leurs tables (sous le compte root de MySQL)
- Consulter les contenus des tables, modifier certaines lignes ou les détruire
- Exécuter des requêtes SQL
- Charger des fichiers dans les tables ou récupérer leur contenu dans des fichiers ASCII

Le grand avantage de **phpMyAdmin** est la gestion à distance de la base de données à l'aide d'un ordinateur connecté au réseau Intenet. C'est la méthode utilisée par les fourniseurs d'accès pour permettre à leurs clients de créer et gérer leurs bases de données MySQL.

La **page d'accueil de phpMyAdmin** ci-dessous est divisée en deux parties :

- A gauche, on voit toutes les bases de données gérées par le serveur (accès root) en particulier, la base **base1** qu'on vient de créer. La base **mysql** contient un ensemble de tables déstinées à gérer les utilsateurs MySQL et leurs droits d'accès aux différentes bases du serveur. Cette partie gauche reste affichée en permanence.

- A droite, on retouve l'ensemble des opérations disponibles en fonction du contexte. Si le compte est root, on accède à toutes les fonctions de phpMyAdmin permettant de consulter la situation du serveur et des clients MySQL.




**Afficher la structure d'une table :** en cliquant sur une des bases, on obtient la liste des tables, et toute une liste d'actions à effectuer sur cette base. (figure ci-dessous). on retouve la base de données **Base1** avec ses deux tables **messages** et **messages_consultation**..

Accueil

base1 (2)

**base1**

- messages
- messages_consultation

# Base de données *base1* - table *messages* sur le serveur *localhost*

[ SQL | Afficher | Structure | Sélectionner | Insérer | Vider | Exporter | Opérations | Options ] [ Supprimer ]

| | Champ | Type | Attributs | Null | Défaut | Extra | Action | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | message_num | int(11) | | Non | | auto_increment | Modifier | Supprimer | Primaire | Index | Unique | Texte entier |
| ☐ | contenu | text | | Oui | *NULL* | | Modifier | Supprimer | Primaire | Index | Unique | Texte entier |

*Pour la sélection :* Modifier *Ou* Supprimer

Index : [Documentation]

| Nom de la clé | Type | Cardinalité | Action | | Champ |
|---|---|---|---|---|---|
| PRIMARY | PRIMARY | 3 | Supprimer | Modifier | message_num |

Créer une clef sur 1 colonne(s) Exécuter

Espace utilisé :

| Type | Espace | |
|---|---|---|
| Données | 120 | Octets |
| Index | 2 048 | Octets |
| Total | 2 168 | Octets |

Statistiques :

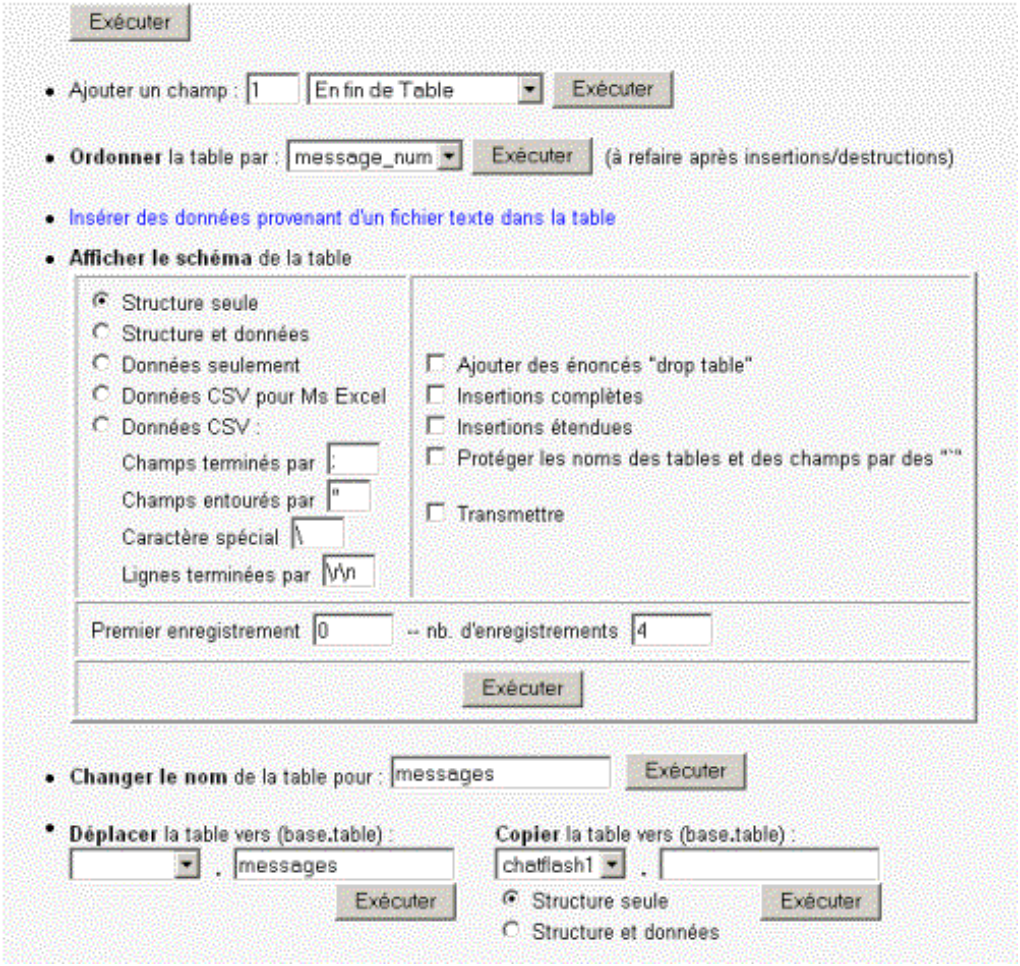
| Information | Valeur |
|---|---|
| Format | dynamique |
| Enregistrements | 3 |
| Longueur enr. ø | 40 |
| Taille enr. ø | 723 Octets |
| Suivant Autoindex | 4 |

- Version imprimable
- Ajouter un champ : 1 En fin de Table Exécuter
- Gestion des relations
- Suggérer des optimisations quant à la structure de la table [Documentation]

On peut y effectuer les opérations suivantes :

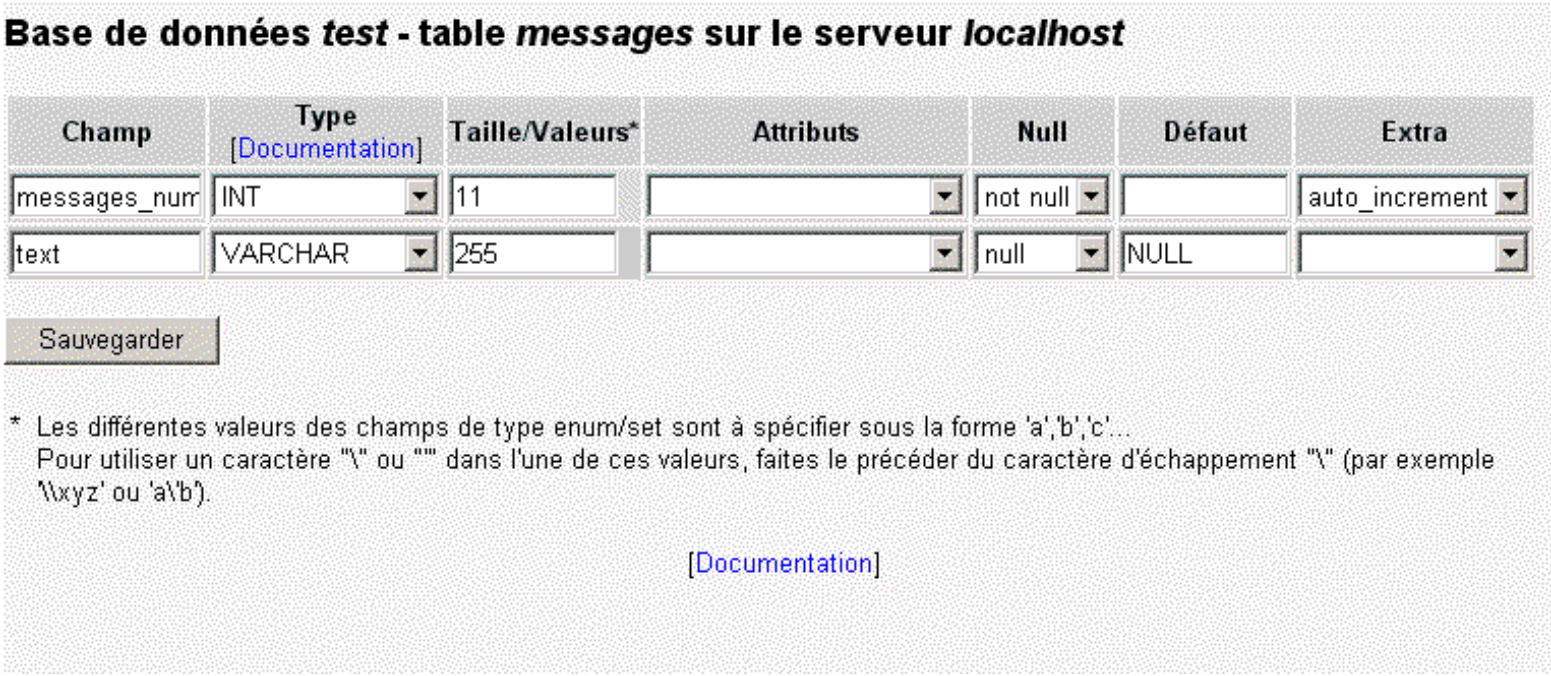
- Afficher : donne le contenu de la table
- Sélectionner : propose un formulaire permettant de sélectionner une partie de la table
- Insérer : permet l'insertion des données dans la table
- Vider : détruit toutes les lignes
- Supprimer : détruit la table après comfirmation.

## afficher le schéma ou créer des champs :




Mr Mohamed SIDIR & Mr Gérard-Michel Cochard

## ajouter ou modifier des champs :



**Base de données *test* - table *messages* sur le serveur *localhost***

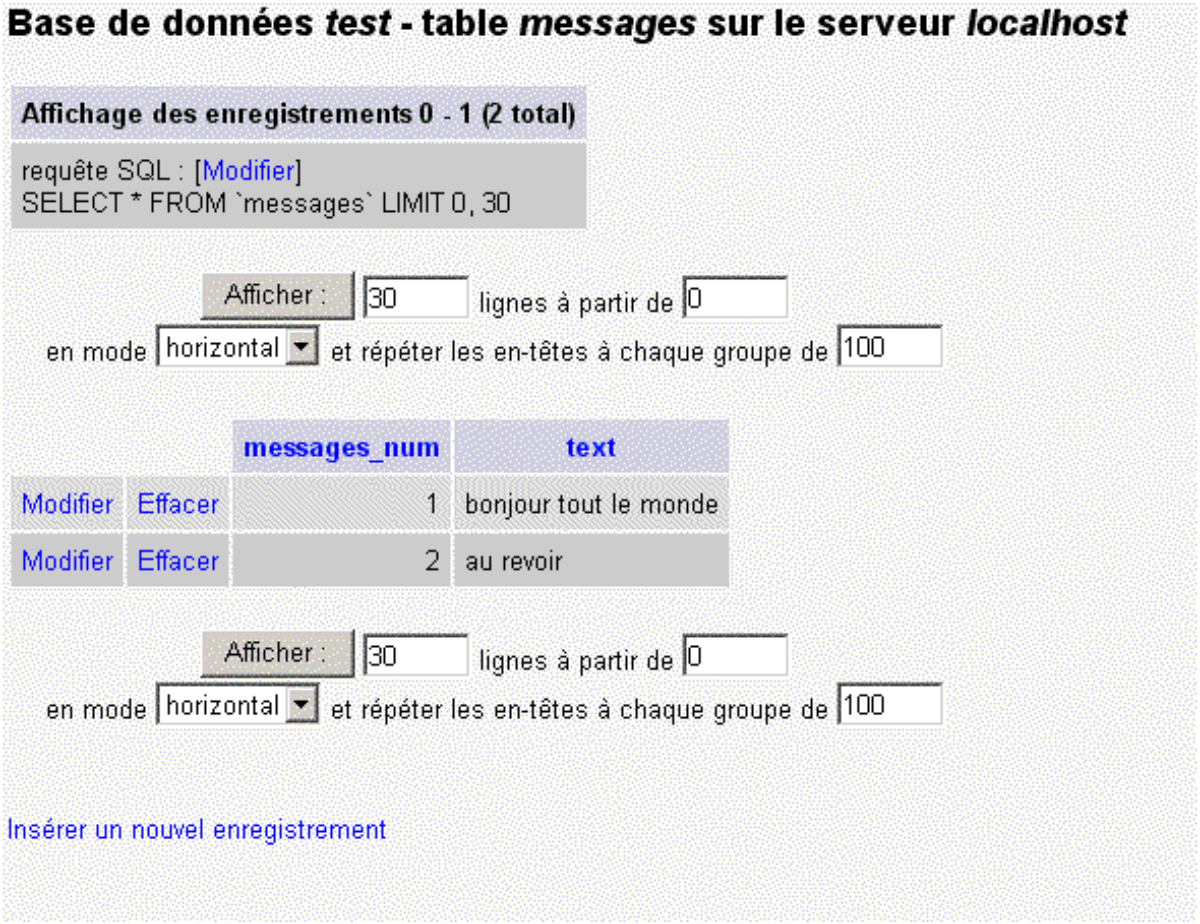
| Champ | Type [Documentation] | Taille/Valeurs* | Attributs | Null | Défaut | Extra |
|---|---|---|---|---|---|---|
| messages_num | INT | 11 | | not null | | auto_increment |
| text | VARCHAR | 255 | | null | NULL | |

Sauvegarder

* Les différentes valeurs des champs de type enum/set sont à spécifier sous la forme 'a','b','c'...
Pour utiliser un caractère "\" ou "'" dans l'une de ces valeurs, faites le précéder du caractère d'échappement "\" (par exemple '\\xyz' ou 'a\'b').

[Documentation]

## afficher le résultat d'une requête (select) :




**dump d'une base de données :**

## Base de données *test* - table *messages* sur le serveur *localhost*

```
# phpMyAdmin MySQL-Dump
# version 2.2.6
# http://phpwizard.net/phpMyAdmin/
# http://www.phpmyadmin.net/ (download page)
#
# Serveur: localhost
# Généré le : Vendredi 22 Novembre 2002 à 15:11
# Version du serveur: 3.23.49
# Version de PHP: 4.2.0
# Base de données: `test`
# --------------------------------------------------------

#
# Structure de la table `messages`
#

CREATE TABLE messages (
  messages_num int(11) NOT NULL auto_increment,
  text varchar(255) default NULL,
  PRIMARY KEY  (messages_num)
) TYPE=MyISAM;

#
# Contenu de la table `messages`
#

INSERT INTO messages VALUES (1, 'bonjour tout le monde');
INSERT INTO messages VALUES (2, 'au revoir');
```