Web Dynamique – Web Mapping

cours n°3

Web dynamique
Générer une page web en PHP

Emmanuel Fritsch – octobre 2008

1 Introduction

1.1 Objectif

Le document « La page web sur le poste client » nous a appris à écrire des pages HTML. Nous savons que le code HTML peut être décrit sous forme d'objets, qu'on peut lui associer des styles au moyen des feuilles de style et que les objets du code HTML peuvent être manipulés au moyen de javascript.

Les pages HTML ainsi décrites sont purement locales. Pour atteindre les capacités du web dynamique, nous devons maintenant gérer l'interaction entre le poste client et le serveur.

Nous allons étudier PHP, un langage qui permet d'écrire du code HTML. Le programme en PHP tourne sur le serveur

L'objectif de ce cours est de vous donner les bases vous permettant de comprendre et de pratiquer la programmation d'un site web dynamique simple.

1.2 prérequis

Pour faire une page dynamique, il faut mettre en œuvre des notions de programmation informatique. L'utilisation du présent cours n'est possible que si vous possédez déjà ces bases de programmation. Vous devez au minimum savoir ce qu'est une variable, une fonction, et être capable de programmer une procédure comportant une boucle. Pour aller plus loin, vous devrez avoir des notions d'orienté objet.

PHP est un langage de programmation procédurale classique, très proche de C ou de Java. Si vous n'êtes pas familier avec C ou Java, vous vous reporterez à l'un des nombreux tutoriaux sur PHP que l'on trouve sur internet.

1.3 Méthode

Une des difficultés du web dynamique réside dans la multiplicité des langages qu'il impose d'aborder. Outre les langages HTML, CSS et javascript, utilisés sur le post client, vous devez maintenant aborder PHP et SQL. PHP, en particulier, présente une syntaxe très proche de celle de Javascript. Mais ces deux langages ne servent pas du tout à faire la même chose, ainsi que nous l'avons vu dans le document « Introduction au fonctionnement du web ».

> Pour tirer le meilleur profit de ce cours, il est souhaitable que vous exécutiez les exemples et que vous réalisiez les exercices

Comme HTML et Javascript, PHP et SQL sont des technologies très utilisées, par des communautés ayant un sens fort du partage. Vous trouverez un grand nombre de ressources, tutoriel, et documentation techniques sur internet.

> Face à une difficulté, n'hésitez pas à utuiliser votre moteur de recherche sur internet. Au fil de vos recherches, sélectionner vos deux ou trois sites favoris sur lesquels vous irez chercher vos renseignements en priorité.

1.4 Matériel

Pour tester les exemples du cours et faire les exercices, vous devez les faire tourner sur un serveur web muni de PHP et MySQL. Vous pouvez utiliser pour cela des services d'un FAI comme free et freesurf, mais le plus simple est d'installer un serveur local, sur votre machine. Dans ce cas, le plus simple est d'utiliser easyPHP. Tous les exercices du cours sont conçus pour fonctionner sur un serveur local.
Vous aurez besoin aussi d'un éditeur de texte et d'un navigateur pour tester les exemples.

2 Premiers pas en PHP

Avant de voir comment on va utiliser PHP pour produire des pages HTML dynamiques, nous allons décrire rapidement les bases du langage.

2.1 L'affichage en PHP

> PHP est un langage de script côté serveur, qui sert essentiellement à produire du code HTML.

Nous avons vu dans l'introduction au web que PHP produit du code HTML au moyen de l'instruction `echo`. Voici le fonctionnement d'une page PHP élémentaire :

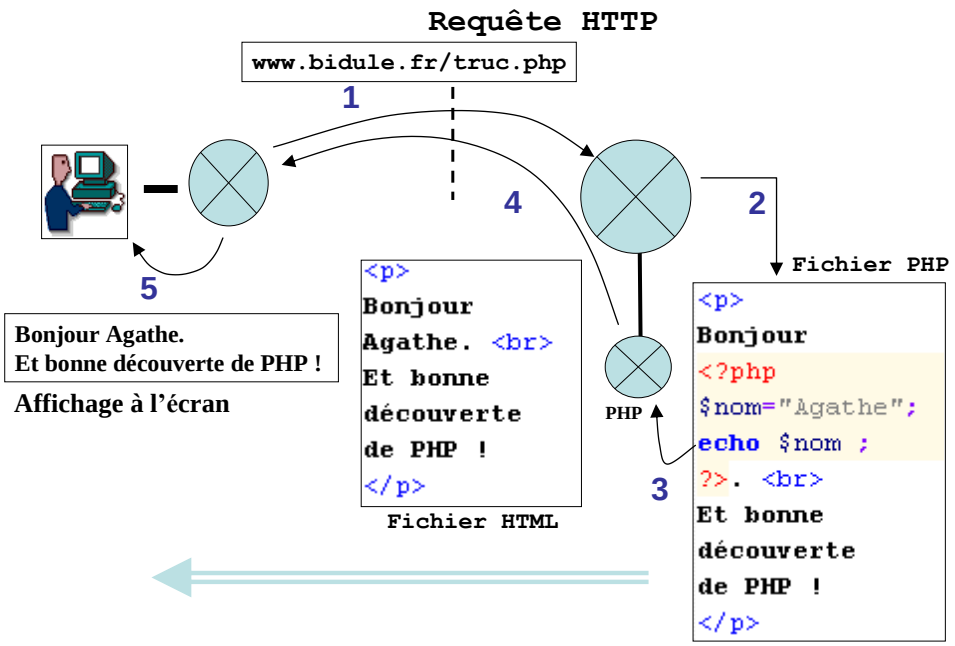


figure 1 – de droite à gauche, du fichier sur le serveur jusqu'à l'affichage chez le client, transformation d'une page dynamique.

La variable `$nom` ne sert pas à grand-chose, car sa valeur ne peut pas être modifiée pour le moment. PHP n'aura de l'intérêt que si nous sommes capables d'intégrer dans notre programme des valeurs qui viennent de l'extérieur.

De fait, PHP permet d'intégrer des informations issues :

- d'un formulaire, rempli par l'utilisateur
- d'une base de données
- d'autres sources de données (cookie envoyé par le navigateur, variables de session, fichier sur le serveur, connexion internet, heures du système, etc.)

Dans ce cours , nous allons examiner successivement comment on manipule :

- les informations émises depuis un formulaire
- les interactions avec une base de données.
- les données récupérées avec un cookie ou au moyen d'une session

Mais dans un premier temps, nous allons passer en revue les bases du langage PHP.

Dans le schéma précédent, nous avons représenté un navigateur envoyant une requête vers un serveur distant : la distance entre les deux est représentée par la ligne en tirets verticale. Dans le cadre de ce cours, nous vous conseillons d'utiliser un serveur local. Dans ce cas, le serveur et le navigateur sont sur la même machine, mais le fonctionnement de leur communication sera le même : il reposera sur HTTP.

Pour tester une page en PHP, il faudra la faire passer par le serveur, pour que le code PHP soit exécuté par le moteur PHP.

2.2 Les bases du langage

Nous allons lister ici quelques particularités de PHP :

PHP dans HTML

Nous avons vu dans le premier cours que le code PHP est inséré dans du code HTML :

```
<?php $nom="Victor Hugo" ; ?>
<p>Bonjour <?php echo $nom ; ?>, je suis heureux de vous voir
</p>
```

exercice 1 :

a. Si vous ne comprenez pas le code précédent, relisez le premier cours.
b. Trouvez le fichier httpd.conf : c'est le fichier de configuration d'Apache.
c. Dans ce fichier, identifiez la ligne déterminant le répertoire racine (cf. le cours n°1).
d. Dans le répertoire racine, créez votre propre répertoire (par exemple cours3/)
e. Recopiez le code dans un fichier exo1.php dans le répertoire cours3.
f. Testez ce code à l'adresse : http://localhost/cours3/exo1.php

Les variables

Toutes les variables commencent par `$`, et tout mot ce qui commence par `$` est une variable.

Substitution de variable

Une chaîne de caractère doit être encadrée par des guillemets simples ou par des guillemets doubles. Ces guillemets ne sont pas équivalents, car les guillemets doubles permettent de faire de la **substitution de variable** :

```
$prenom = "Agathe" ;
echo "Bonjour $prenom ! <br>" ;
echo 'Bonjour $prenom ! <br>' ;
```

Le résultat de ces instructions sera :

Bonjour Agathe !
Bonjour $prenom !

Avec les guillemets doubles, la variable est remplacée par sa valeur.

Les opérateurs de base

Les opérateurs arithmétiques sont les mêmes qu'en C, en java et en javascript. Voir plus haut.

Concaténation

En PHP, le point est l'opérateur de concaténation de chaîne.

```
echo "Bonjour " . $prenom . "! <br>" ;
```

Cette instruction donne le même résultat qu'avec la substitution de variable :

Bonjour Agathe !

Egalité et affectation

Il ne faut pas confondre l'opérateur d'égalité avec l'instruction d'affectation. Nous l'avons dit pour javascript, répétons le pour PHP :

- Pour affecter une valeur à une variable, on utilise l'opérateur **=**
- Pour tester une égalité, on utilise l'opérateur **==**

```
if ($prenom=="Agathe") $message="C'est un très beau prénom" ;
```

Les tableaux

PHP présente plusieurs manières de définir un tableau :

```
$auteurs[0] = "Victor Hugo";
$auteurs[1] = "Jean de La Bruyère";
$auteurs[2] = "Blaise Pascal";
```

ou bien :

```
$auteurs = array( 0 => "Victor Hugo";
       1 => "Jean de La Bruyère"; 2 => "Blaise Pascal" ) ;
```

Si l'indice n'est pas précisé, la numérotation est automatique. Les deux écritures suivantes sont équivalentes aux deux précédentes :

```
$auteurs[] = "Victor Hugo";
$auteurs[] = "Jean de La Bruyère";
$auteurs[] = "Blaise Pascal";
```

ou bien :

```
$auteurs=array("Victor Hugo","Jean de La Bruyère","Blaise Pascal");
```

Tableaux associatifs

Dans un tableau classique, les objets sont indexés par des entiers positifs. Un tableau associatif accepte comme indice n'importe quel type alphanumérique.

```
$tabFruits["la fraise"] = "rouge";
$tabFruits["la banane"] = "jaune";
$tabFruits["la courgette"] = "verte";
echo $tabFruits["la fraise"] ;
```

Le résultat du code précédent est : rouge.

La substitution de variable fonctionne :

```
$fruit="la fraise" ;
echo "j'aime $fruit quand elle est bien $tabFruits[$fruit]" ;
```

Exercice 2 :

- en recopiant le code ci-dessus, faire le test de la substitution de variable avec des tableaux.

Attention aux conflits de guillemets :

```
echo "la fraise est $tabFruits["la fraise"]" ;
echo "la fraise est $tabFruits['la fraise']" ;
```

La première ligne provoque une erreur. C'est le conflit de guillemet.
La seconde ligne ne fonctionne pas non plus. C'est dû au fonctionnement interne de PHP : la lecture d'une constante est interprétée comme

Les structures

On peut construire des objets qui contiennent plusieurs valeurs :

```
$fruit->nom="fraise" ;
$fruit->article="la" ;
$fruit->couleur="rouge" ;
```

Et la substitution de variable ici aussi fonctionne parfaitement :

```
echo "$fruit->article $fruit->nom est $fruit->couleur" ;
```

Les boucles

PHP connaît les boucles classiques :
- la boucle **`while`** :

```
$a=1 ;
while ($a<1000) { echo "--  $a  " ; $a=$a*2 ;}
```

résultat:

-- 1 -- 2 -- 4 -- 8 -- 16 -- 32 -- 64 -- 128 -- 256 -- 512

- la boucle **`for`** :

```
for ($i=0; $i<5 ; $i++) echo "6 x $i = " . ($i*6) . "<br>";
```

résultat :

6 x 0 = 0
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24

Si on ne met pas les parenthèses, PHP prend les opérations dans l'ordre : il concatène la chaine avec $i, puis il essaye de multiplier le résultat avec 6, ce qui déclenche une erreur.

La boucle foreach

La boucle foreach permet de parcourir un tableau, sans s'occuper du compteur :

```
$tabFruits["la fraise"] = "rouge";
$tabFruits["la banane"] = "jaune";
$tabFruits["la courgette"] = "verte";

foreach ($tabFruits as $fruit => $couleur ) {
  echo "$fruit est $couleur <br> ; "
}
```

A chaque itération de la boucle, les variables **`$fruit`** et **`$couleur`** reçoivent des valeurs tirées du tableau : au premier passage, les deux valeurs sont **`"la fraise"`** et **`"rouge"`**. A l'itération suivante, **`"la banane"`** et **`"jaune"`**, et finalement, **`"la courgette"`** et **`"verte"`**.

Ces affectations sont automatiquement effectuées par la syntaxe de la boucle foreach

> La boucle foreach est extrêmement pratique pour parcourir les éléments d'un tableau.
> Elle est même indispensable lorsque le tableau est associatif.

La syntaxe de cette boucle est la suivante :

```
foreach ($tableau as $clef => $valeur ) { instructions }
```

Instructions conditionnelles

Simple et classique :

```
if (condition) {
  instructions si la condition est vraie
}
else
{
  instructions si la condition est fausse
}
```

On peut enchaîner les conditions :

```
if (condition1) {
  instructions si condition1 est vraie
}
else if (condition2) {
  instructions si condition1 est fausse et condition2 est vraie
}
else
{
  instructions si les 2 conditions sont fausses
}
```

2.3 PHP et les formulaires HTML

Dans le cours sur HTML, nous avons vu comment construire un formulaire. Nous allons étudier maintenant la façon dont nous allons récupérer et traiter ces informations dans un script PHP sur le serveur.

Nous allons voir successivement :

1- comment le client envoie les valeurs du formulaire vers le serveur
2- comment le programme PHP récupère ces valeurs dans des variables pour générer sur le serveur une page dynamique qui tienne compte de ces valeurs

2.3.1 Le transfert de l'information

Reprenons le formulaire qui nous avait servi d'exemple :

```
<form action="page2.php" method=GET >
  votre nom : <input type=text name="nom">
  <br>
  votre prénom : <input type=text name="prenom">
  <br>
  <input type=submit value="Clic !">
</form>
```

> L'attribut **action** de l'objet formulaire **form** donne le nom de la page qui traite, sur le serveur, les informations postées par le formulaire. Les informations traitées sont définies par les objets **input**, **textarea** et **select** définis à l'intérieur du formulaire.

Les informations postées par ce formulaire doivent donc être traités par la page **page2.php**. Créons donc une page nommé **page2.php**, qui contient simplement du code HTML :

```
<html>
<head>
<title> Ma première page </title>
</head>

<body>
<p> Bonjour ! </p>
</body>
</html>
```

Pour le moment, avec cette page, le serveur ne peut pas traiter les informations du formulaire, puisque la page ne contient pas de code PHP. Mais nous pouvons déjà vérifier que les variables sont transmises au serveur, en regardant la barre d'adresse de notre navigateur.

En effet, si je remplis les champs **nom** et **prenom** avec les valeurs « Sepoher » et « Agathe », et que je clique sur le bouton **submit**, voici ce qui s'affiche dans ma barre d'adresse :

```
page2.php?nom=Sepoher&prenom=Agathe
```

C'est le navigateur qui a construit cette adresse :

- **page2.php** est le nom de la page qui se trouve dans l'attribut **action** du formulaire,
- **?** : ce point d'interrogation signale le début des variables transmises,
- les variables sont séparées par des éperluettes (**&**). Chaque champ du formulaire donne une variable.
- **nom=Sepoher** : la variable **nom** porte la valeur **Sepoher**. Chaque variable est transmise sous la forme :

  nomDeVariable=valeurDeLaVariable.
- le nom de la variable est fixé par l'attribut **name** du champ correspondant,
- la valeur correspond à celle entrée dans le formulaire. Par défaut, c'est l'attribut **value** de chaque champ.

Le serveur recevant cette adresse va la décomposer pour reconstituer le nom de la page et la liste des variables. Le nom de la page permet de retrouver celle-ci et de la faire interpréter par PHP. La liste des variables est transmise au programme PHP.

2.3.2 Récupération des variables dans PHP

> Lorsque la page PHP est interprétée par le serveur, les variables transmises par le formulaire sont accessibles dans un tableau associatif nommé **$_GET**

Nous utilisons maintenant le nom et le prénom envoyés par le formulaire pour construire notre première vraie page dynamique :

```
<html>
<head>
<title> Ma première page </title>
</head>

<body>
<p> Bonjour <?php echo $_GET["prenom"] ?>, </p>
<p> vous vous appelez vraiment <?php echo $_GET["nom"] ?> ? C'est un
nom très original ! </p>
</body>
</html>
```

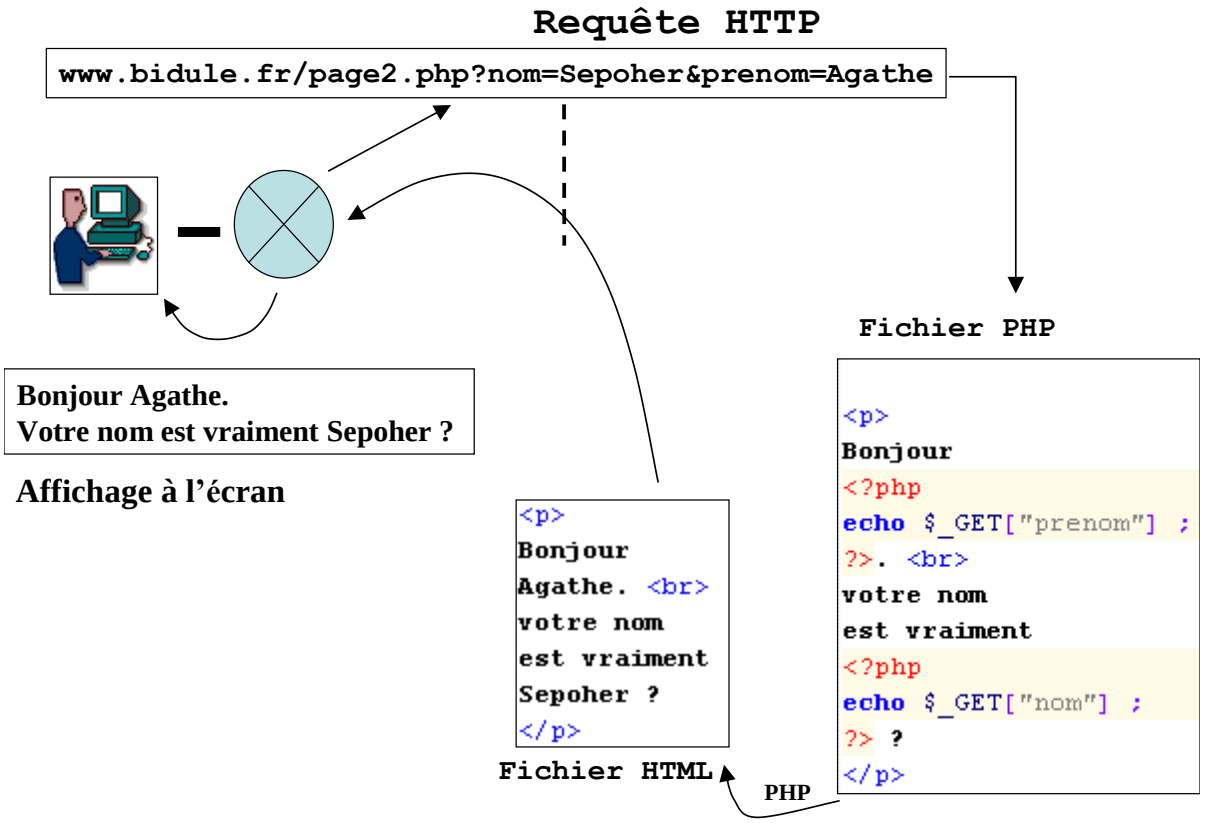


figure 2 – circulation des requêtes et des variables lors de l'utilisation d'un formulaire

Exercice 3 :

a. Utilisez la boucle foreach pour afficher le nom et la valeur de tous les champs du formulaire. Vous pouvez vous appuyer sur le code ci-dessous :

```
<html>
<head>
<title> Ma première page </title>
</head>

<body>
<p> Le formulaire contient : </p>
<?php
            // ici mettre la boucle "foreach"
?>
</body>
</html>
```

Vous trouverez facilement des éléments de solutions sur internet.

b. Mettez un menu déroulant à choix multiple dans un formulaire (voir l'exercice sur les formulaires dans le cours n°2). Si on sélectionne plusieurs choix dans ce menu, comment la variable est-elle transmise dans l'adresse vers le serveur ? Toutes les variables sont elles transmises à PHP (utilisez le programme de 3.a pour répondre)

c. Chercher sur internet une méthode qui permette à PHP de récupérer toutes les valeurs du choix multiple.

d. Modifiez le programme du 3.a pour visualiser la transmission de ces variables.

Dans certains tutoriaux, dans certains cours sur PHP, `$_GET["nom"]` et `$_GET["prenom"]` sont remplacés par `$nom` et `$prenom`. Ce fonctionnement, extrêmement répandu, n'est pas universel. **Il correspond à une configuration spéciale de PHP, qui est de plus en plus souvent désactivé, sous des motifs de sécurité**.
Par surcroît, autre avantage, l'utilisation du tableau `$_GET` exprime de manière explicite que la variable est issue d'un formulaire : l'écriture `$_GET` doit donc être préférée aussi pour des raisons de lisibilité.

Exercice 4 :

- Cherchez l'option de configuration de PHP qui permet d'utiliser directement les variables `$nom` et `$prenom` à la place de `$_GET["nom"]` et `$_GET["prenom"]` (ne passez pas plus de 10 minutes sur cet exercice).

2.3.3 Méthode GET et méthode POST

Il nous reste à parler de l'attribut `method` du formulaire.

Dans le protocole HTTP, il existe deux méthodes pour envoyer une requête : la méthode POST et la méthode GET. Toutes les requêtes HTTP comporte d'une part un entête, d'autre part un corps. L'entête contient les méta-données de la requête, et elle doit rester de petite taille. La méthode POST contient le corps de la requête, et sa taille n'est théoriquement pas limitée.

Dans le protocole HTTP, il existe deux types de requêtes pour appeler une page web : la requête POST et la requête GET. Quelle est la différence entre ces deux requêtes ? En général, les requêtes HTTP comportent d'une part un entête, d'autre part un corps.

- l'entête ne contient que les méta-données de la requête, et elle doit rester de petite taille ;
- le corps de la requête sert à faire transiter les informations volumineuses. La taille du corps de la requête n'est théoriquement pas limitée.

Une requête GET, c'est une requête dont le corps est vide, car toute l'information de la requête est condensée dans l'entête. Par exemple, lorsque vous cliquez sur un lien internet, votre navigateur utilise une requête GET. Une requête POST, c'est une requête qui met des données dans le corps de la requête.

Lorsque les valeurs saisies dans un formulaire sont de petites tailles, ce qui est le cas la plupart du temps, on a donc le choix :

- soit on stocke les variables dans l'adresse de la page, comme nous venons de le voir au chapitre précédent, et dans ce cas il n'y a pas besoin d'un corps de requête : c'est la méthode GET ;
- soit on les stocke dans le corps de la requête, et on choisit pour cela la méthode POST.

Pour envoyer un mot de passe, par exemple, il vaut mieux utiliser la méthode POST. En effet, les utilisateurs n'aiment pas du tout voir leur mot de passe apparaître dans la barre d'adresse. Ils s'imaginent que c'est plus sûr si on ne voit pas le mot de passe. Ils n'ont pas totalement tort.

Notez que dans HTML, le nom et la valeur de l'attribut **`method`** ne sont pas sensibles à la casse. On peut écrire indifféremment **`method=GET`**, ou **`method=get`**, ou même **`mEthoD=gEt`** (même si c'est affreux). Mais attention, PHP en revanche est sensible à la casse et les majuscules sont obligatoires pour le tableau **`$_GET`**, ainsi que pour tous les tableaux bâtis sur le même modèle.

Exercice 5 :

a. comment s'appelle le tableau associatif dans lequel PHP va stocker des variables envoyées avec la méthode POST ? (Question facile !)
b. transformer **`page1.html`** et **`page2.php`** pour transférer les champs du formulaire en méthode POST.
c. Vérifier que la page fonctionne ET que le formulaire est bien passé en méthode POST
d. Comment avez-vous vérifié que la requête est bien en méthode POST ?

2.4 Conclusion

Nous avons appris à programmer une page simple, et à récupérer de l'information issue d'un formulaire posté par l'utilisateur sur le poste client.

Notre deuxième source d'information pour fabriquer une page web sera la base de données. Mais avant d'apprendre à faire interagir PHP avec une base de données, nous allons faire le point sur ce qu'est une base de données.

3 Premiers pas en SQL

Avertissement : ceci n'est pas un cours complet sur SQL ni sur les bases de données. Nous allons donner ici les informations qui permettent de mettre en œuvre la partie SQL d'un projet Web simple.

3.1 Présentation

3.1.1 Qu'est-ce que SQL ?

« Structured query language » (SQL), ou langage structuré de requêtes, est un pseudo-langage informatique (de type requête) standard et normalisé, destiné à interroger ou à manipuler une base de données relationnelle » (*d'après Wikipedia, le 26 juin 2007*)

> SQL est un langage de requête permettant de manipuler des données rangées sous forme de **tables** selon un modèle relationnel.

3.1.2 Les tables

> Une table permet de stocker les caractéristiques d'une collection d'objets. Chaque objet sera une ligne dans la table. Chaque colonne représentera une caractéristique dont la valeur est différente pour chaque objet.

Voyons cela avec un exemple ; nous voulons stocker des informations sur les membres d'une fédération sportive. Pour chaque membre, nous voulons stocker :

- le nom
- le prénom
- la date de naissance
- la civilité (Mr / Mme / Mlle)
- le club auquel il est affilié

Si nous rangeons nos données en tableau, elles vont ressembler à ceci :

nom	prénom	civilité	né le	club
Némar	Jean	Mr	12 déc. 1970	Les nageurs du Loiret
Thiedgut	Justine	Melle	5 janv. 1996	Les Barboteurs de Vesoul
Conda	Anna	Mme	12 juin 1946	Le Plongeon d'Ajaccio
Térieur	Alain	Mr	5 avril 1962	Les Canards du val de Seine
Menvussa	Gérard	Mr	17 fév. 1975	Les Canards du val de Seine
Thimes	Vincent	Mr	26 sept. 1964	Les barboteurs de Vesoul

Vous pouvez vous dire que pour obtenir ce résultat, il suffit d'utiliser un tableur comme excel, mais l'utilisation d'un SGBD va apporter de nombreux avantages :

- on va pouvoir contrôler les données que l'on met dans le tableau pour empêcher par exemple qu'on se trompe de colonne, ou que la date soit incompréhensible, ou qu'un même objet apparaisse avec deux noms différents (ce qui est le cas ici pour les Barboteurs de Vesoul).

- on va pourvoir extraire facilement des morceaux de tableaux, et trouver des réponses à des questions complexes (quels sont les adhérents mineurs, quel sont les clubs qui ont moins de 3 adhérents, etc.)
- on va pouvoir gérer rapidement de gros volumes de données, ce que les tableurs ne savent pas faire.

> Le SGBD assure donc :
> . la cohérence interne des données
> . l'exécution de requêtes complexes
> . la gestion rapide de gros volumes de données

Soignons plus précis :

- La **cohérence interne** est fixe par le **modèle** auquel doivent se conformer les données.
- L'exécution de **requêtes complexes** est assurée par les **extractions**, et en particulier les **jointures**.
- La **rapidité** de la manipulation est assurée par des **index**.

Nous allons maintenant détailler ces trois points.

3.2 Modèle physique de données

> Le modèle physique de données, c'est la manière dont vous allez structurer les données dans les tables du SGBD.

Normalement, vous devez toujours faire dériver votre modèle physique de données d'un modèle conceptuel de données que vous aurez construit au préalable.

3.2.1 Les identifiants

> Un identifiant est une colonne dont la valeur est unique pour chaque objet de la table. Il permet de retrouver un objet de manière univoque, donc garantie.

Votre numéro de sécurité sociale, votre numéro de compte bancaire, votre numéro de passeport, la numérotation de vos chèques, sont autant d'identifiants : ils servent à identifier, sans équivoque possible, les objets auxquels ils s'appliquent.

Votre adresse postale, par contre, n'est pas un identifiant : plusieurs personnes peuvent habiter à la même adresse.

Le code barre d'un produit est-il un identifiant ? Ce n'est pas un identifiant de chaque objet, puisque tous les produits semblables ont le même code barre. Par contre, dans un magasin, tous les objets ayant le même code barre doivent être du même type : le code barre est un identifiant de la référence produit.

Dans la base de données, un identifiant n'est pas obligatoirement un entier. L'avantage des identifiants en nombres entiers, c'est qu'on peut facilement en générer de nouveaux, toujours différents, par incrémentation du précédent.

3.2.2 Modèle conceptuel de données

Il n'est pas question de faire ici un cours sur les modèles conceptuels de données. Limitons nous à constater une chose : dans notre exemple, la civilité, le nom, le prénom et la date de naissance sont des informations qui appartiennent en propre à chaque membre. Certains membres peuvent avoir la même date de naissance ou le même prénom, mais on est ici dans l'ordre de l'accidentel, du fortuit, du contingent.
Au contraire, le fait que deux membres aient le même club n'est pas un accident, et le nom du club n'appartient pas en propre à chaque membre.

La bonne méthode, pour représenter correctement les données, conduit à isoler d'un côté tous les membres, avec les données qui leur sont propres, et de l'autre côté les clubs, avec les données propres à eux.

Nous constituons ainsi deux classes d'objets. Entre ces deux classes, nous avons une relation, qui indique pour chaque membre le club auquel il appartient.

En UML (le langage graphique dédié à ce type de modélisation) nous obtenons :

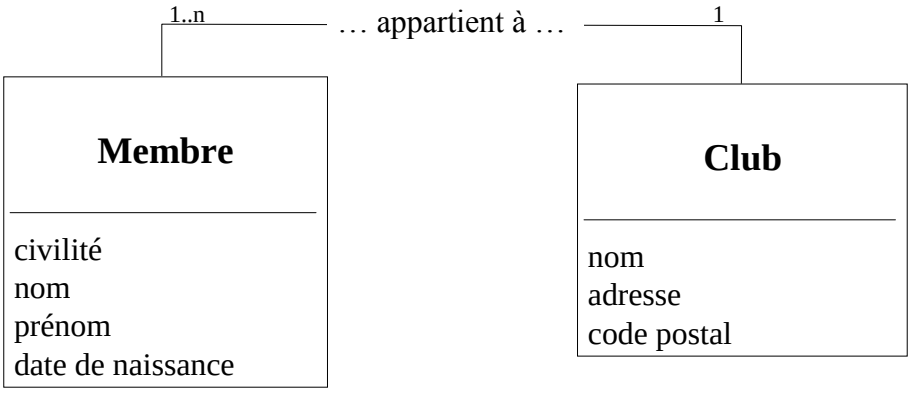


figure 3. diagramme UML à deux classes. Un lien (en UML : une association) relit les deux classes pour représenter la relation d'appartenance d'un membre à un club.
Du côté des membres, la relation est 1-n, car un club contient au moins un membre, et souvent beaucoup plus. Du côté des clubs, la relation est « unaire », car un membre appartient à un club et un seul.

3.2.3 Conversion vers le modèle physique de données

Une classe dans mon modèle conceptuel de données sera représentée par une table dans mon SGBD. Nous aurons donc la table des clubs et la table des membres. Les attributs de chacune de mes classes vont apparaître sous forme de colonne dans la table correspondante.

Le codage des liens est un peu plus complexe. Comme chaque membre appartient à un club (et un seul), le lien peut être codé de la manière suivante :

- on place un identifiant sur la table des clubs
- on ajoute un attribut "identifiant de son club" dans la table des membres.

On obtient les tables suivantes :

nom	prénom	civilité	né le	idClub
Némar	Jean	Mr	12 déc. 1970	1
Thiedgut	Justine	Melle	5 janv. 1996	2
Conda	Anna	Mme	12 juin 1946	3
Térieur	Alain	Mr	5 avril 1962	4
Menvussa	Gérard	Mr	17 fév. 1975	4
Thimes	Vincent	Mr	26 sept. 1964	2

et :

id	nom	Adresse	codePostal
1	Les nageurs du Loiret	16 rue de la mare	45000
2	Les barboteurs de Vesoul	39 av Weismueller	70000
3	Le Plongeon d'Ajaccio	2 pl. de la	20200
4	Les Canards du val de Seine	3 rue Hamelin	75005

Pour retrouver le club d'un membre, on regarde la colonne **`idClub`** de la table **`membre`**, et on cherche le numéro correspondant dans la colonne **`id`** de la table **`club`**. Par exemple :

Quel est le nom du club auquel appartient Justine Thiedgut ? On regarde dans la table **`membre`** pour voir qu'elle appartient au club d'identifiant 2. On regarde ensuite dans la table **`club`** pour voir que le club d'identifiant 2 s'appelle « les barboteurs de Vesoul ».

Exercice 6 :

a. Modifiez le modèle conceptuel et le modèle physique de données pour qu'on puisse signaler dans la base qu'un membre est président de son club. Pour le modèle physique de données, il y a au moins deux possibilités : on peut faire un lien entre les deux tables. On peut aussi s'en passer en considérant qu'un président doit être obligatoirement membre du club qu'il préside. Traiter les deux cas.
b. Quelle est la meilleure solution en terme d'espace occupée ?
c. Quelle est la meilleure solution en terme de rapidité des requêtes ? (répondre à cette question après avoir lu tout le chapitre SQL).

Ici, le codage des liens est simple car un membre appartient au maximum à un seul club. Si on considérait qu'un membre peut être adhérent à plusieurs clubs en même temps, la représentation du lien serait un peu plus complexe. Reportez-vous à votre cours de base de données pour aller plus loin.

3.2.4 Création des tables en SQL

Le langage SQL permet de créer et de modifier des tables avec les commandes **CREATE TABLE** et **ALTER TABLE**. Vous vous reporterez à la documentation SQL si vous avez besoin de ces commandes, mais pour commencer, vous construirez vos tables avec une interface graphique, comme PhpMyAdmin.

3.3 Manipulation des données

3.3.1 Consulter les données : la requête SELECT..FROM

> Une extraction SQL est une opération de sélection de certaines données dans un tableau.

Attention, extraire une ligne d'un tableau ne signifie pas que la ligne a été retirée du tableau. Une extraction, en SQL, consiste à définir dans le tableau un sous-ensemble de données qu'on veut plus spécifiquement étudier. Ce sous-ensemble qu'on extrait du tableau n'est pas détruit dans le tableau.

SELECT..FROM est l'instruction SQL pour l'extraction. L'extraction la plus simple consiste à prendre le tableau en bloc :

```
SELECT * FROM club
```

Le langage SQL est assez intuitif à interpréter. L'instruction précédente se lit de la manière suivante :

SELECT	=>	affiche
*****	=>	la totalité des colonnes
FROM	=>	extraite de
club	=>	la table club

Pour définir des sous-ensembles dans le tableau, nous allons apprendre à restreindre l'extraction à certaines colonnes et à certaines lignes.

Attention, le langage SQL est insensible à la casse. Dans le cadre de ce cours, pour des raisons de lisibilité, nous mettons systématiquement en majuscules la syntaxe SQL. Les noms de variable, de colonnes et de tables sont par contre sensibles à la casse.

3.3.1.1 Restriction sur les colonnes (projection)

Avec la requête suivante, nous allons restreindre l'extraction aux colonnes **id**, **nom** et **codePostal** :

```
SELECT id,nom,codePostal FROM club
```

Si une colonne porte un nom avec des caractère spéciaux ou un espace, il doit être écrit entre guillemets. Ainsi, si on remplaçait la colonne **codePostal** par une colonne **code postal**, la requête deviendrait :

```
SELECT id,nom, 'code postal' FROM club
```

Cette règle s'applique aussi sur les noms de tables, et tous les noms d'objet SQL. Pour éviter ces guillemets qui alourdissent la lecture du code, il est recommandé d'utiliser des noms de colonnes et de tables en un seul mot, sans caractère spéciaux.

Extraire la totalité des colonnes est coûteux en temps de calcul et en espace mémoire. Pour optimiser, il faut impérativement limiter le nombre de colonne extraite au nombre nécessaire. Si le travail de conception est mené correctement, le programmeur doit savoir quelles sont les colonnes qu'il a réellement besoin d'extraire. L'extraction de toutes les colonnes avec la syntaxe **`SELECT * FROM`** doit être évitée.

3.3.1.2 Restriction sur les lignes

Plutôt que d'extraire tous les objets d'une table (toutes les lignes), on peut souhaiter n'en extraire qu'une partie. Ici, nous allons extraire les colonnes **`id`**, **`nom`** et **`codePostal`** :

Extraire les identifiants et les noms des membres prénommés Anna :

```
SELECT id,nom FROM membre WHERE prenom="Anna"
```

Extraire les noms et les identifiants des clubs dont le président cumule aussi la fonction de secrétaire

```
SELECT id,nom FROM club WHERE idPresident=idSecretaire
```

Remarquez le piège : en SQL, l'opérateur d'égalité est représenté par '=', alors que pour PHP et javascript, l'égalité est codé par '==', et '=' est une affectation de variable.

Exercice 7 (difficile) :

- Extraire les noms et les ages des membres mineurs.

(pour les opérations sur les dates, consultez la documentation sur internet)

3.3.1.3 Alias de colonne

Pour des raisons de commodité, de lisibilité du résultat, nous pouvons souhaiter que dans le résultat de la requête la colonne n'ait pas le même nom que dans la table. Par exemple pour des raisons de compatibilité avec un programme anglo-saxon, nous souhaitons que la colonne codePostal **`codePostal`** s'appellen **`zipCode`** dans le résultat. Nous allons utiliser un alias :

```
SELECT id, nom, codePostal AS zipCode FROM club
```

Exercice 8 :

- on veut que **`nom`** et **`prenom`** s'appellent s'appellent **`firstName`**, **`lastName`** dans le résultat. Ecrire la requête SQL.

Corrigé :

```
SELECT id,nom AS lastName,prenom AS firstName FROM membre
```

3.3.2 Produit cartésien

```
SELECT * FROM club, membres
```

Cette requête renvoie tous les couples (club,membre) qu'il est possible de construire : avec 10 membres et 4 clubs, nous aurons donc 40 paires (club,membre) possibles.

Ce type de requête est très peu utile : à quoi cela sert-il de construire toutes les combinaisons (club,membre), y compris les combinaisons où le membre n'a aucun lien avec le club ?

Ce qui nous intéresse par contre, c'est d'extraire de l'ensemble de toutes les paires possible (membre,club), celles pour lesquelles le membre a une relation spéciale avec le club. Par exemple : extraire tous les membres avec le nom du club auxquels ils appartiennent. Ou bien extraire tous les clubs avec le nom de leur président.

L'opération qui permet d'extraire ces informations présentes sur deux tables s'appelle la jointure.

Pour plus d'information sur les jointures, cherchez la documentation sur internet.

Exercice 9

a. écrire la jointure qui permet de trouver tous les membres, avec le nom de leur club
b. écrire la jointure qui permet de trouver tous les clubs, avec le nom de leur président (cherchez sur internet ce que signifie « jointure à gauche », et utilisez cette technique pour répondre à la question)
c. (difficile) normalement, le président d'un club doit être membre de ce club. Ecrire la requête SQL qui trouve les noms et identifiants des présidents inscrits comme membres d'un autre club que celui qu'ils président. (jointure de trois tables, mais sans auto-jointure)

solution de la question c :

```
SELECT id, membre.nom FROM membre, club as clubM, club as clubP
                WHERE membre.idClub = clubM.id
                     AND membre.id = clubP.idPresident
                     AND clubM.id <> clubP.id
```

3.3.3 Insertion, suppression et mise à jour

Les instructions d'insertion, de suppression et de mise à jour sont respectivement :

```
INSERT INTO club(nom, codePostal)
       VALUES ('Les barboteurs de Vesoul',70700)

DELETE FROM club WHERE id=1

UPDATE club nom='Les barboteurs de Vesoul' WHERE id=2
```

Vous vous reporterez à la documentation lorsque vous en aurez besoin.

3.4 Les index

3.4.1 La recherche séquentiel

Pour gérer rapidement les objets dans une base, il faut pouvoir les retrouver très vite. Si je cherche à retrouver un objet dans un tableur, par exemple si je cherche la date de naissance de Gérard Menvussa, je suis obligé de lire chaque ligne, pour regarder si elle correspond à la personne que je cherche, pour lire dans la bonne case la date de naissance qui lui correspond.

En moyenne, je vais lire environ la moitié du tableau pour chaque requête. Si j'ai un grand nombre d'enregistrements, ca peut durer longtemps.

Pour faire comprendre avec une image, c'est comme si, lorsque je cherche une définition dans le dictionnaire, je commençais par le premier mot sur la première page, puis que je passais au deuxième mot, puis après avoir contrôlé tous les mots de la première page, je passais à la seconde page, puis à la troisième, etc. Ceci n'est vraiment pas la bonne méthode pour trouver une définition. Ce n'est pas non plus la bonne méthode pour explorer un gros volume de données.

> La recherche séquentielle n'est pas la bonne méthode pour explorer un gros volume de données

3.4.2 La recherche indexée

Pour trouver un mot dans le dictionnaire, la bonne méthode consiste à couper au milieu, à décider, en fonction de l'ordre alphabétique, si le mot est à gauche ou à droite, à prendre le côté où l'on sait que se trouve ce mot que l'on cherche, et à recommencer l'opération sur cette moitié de dictionnaire.

Avec cette méthode, que l'on appelle **dichotomique**, à chaque fois que nous lisons un mot, nous supprimons de la recherche la moitié des mots qui restent. Comparée à la méthode précédente, notre méthode dichotomique va très très vite.

Dans une base de données, un index est un mécanisme qui permet de faire des recherches rapidement, en imitant la façon dont nous cherchons dans un dictionnaire : couper en deux les données, et déterminer de quel côté se trouve les objets que l'on cherche. Il n'est pas question de décrire ici le mécanisme des index. De toute façon, l'utilisateur d'une base de données n'a pas besoin de le connaître. Retenons simplement le principal :

> Un index est un mécanisme qui permet d'accélérer les recherches dans les bases de données.

3.4.3 Index et identifiant : les clefs

Dans une table, on indexe les identifiants en premier lieu, puisque ce sont les informations spécifiquement destinées à identifier les objets. Mais d'autres colonnes peuvent recevoir des index.

Par exemple, dans le cas du dictionnaire, chaque mot du dictionnaire n'est pas un identifiant, puisque deux mots différents peuvent avoir exactement la même orthographe. De même, dans

la table des membres, certains membres peuvent avoir le même nom. Si on demande à retrouver tous les gens qui portent ce nom, on en trouvera plusieurs.

Par contre, lorsqu'on crée un identifiant sur un objet, c'est précisément pour retrouver facilement cet objet. Si on crée un identifiant, on doit généralement lui donner un index. Dans ce cas, notre index s'appelle une **clef**.

> Dans un SGBD, on appelle **clef** un index portant sur un identifiant

Avec SQL, lorsqu'on a besoin de plusieurs clefs sur une table, il y en a une qui joue un rôle particulier, et qu'on appelle la **clef primaire**. La clef primaire, c'est celle qui donne l'ordre naturel des données.

Il y a une seule clef primaire par table, mais on peut mettre ensuite autant de clefs et d'index que l'on veut.

3.4.4 Où dois-je créer un index ?

> Il faut mettre des clefs et des index sur les données qui en ont besoin, c'est-à-dire sur les colonnes de mes tables que j'utilise souvent dans les clauses WHERE de mes requêtes.

3.5 Pour aller plus loin

Ce que nous avons vu jusqu'ici est largement suffisant pour débuter avec SQL dans un projet web, mais si vous voulez utiliser pleinement la puissance de SQL, vous devez maîtriser les concepts suivants :

- L'autojointure
- Les jointures externes, à gauche et à droite (requêtes qui permettent par exemple d'extraire tous les clubs avec leur président, y compris les clubs qui n'ont pas de président).
- Le regroupement de lignes (la syntaxe SELECT – GROUP BY permettant de répondre aux questions du type : trouver, pour chaque club, le nom du membre le plus jeune).
- La gestion des droits
- Les Schéma et les Vues (sous Oracle ou PostGreSQL)
- La programmation en PL(sous Oracle ou PostGreSQL)
- Les contraintes, les relations et les trigger

Pour aller plus loin, je vous conseille les liens suivants :
Un point d'entrée théorique sur SQL :
http://fr.wikipedia.org/wiki/SQL

Une ressource très large :
http://sql.developpez.com/
- Le SQL de A à Z pour une vue complète du langage SQL
- Les petits papiers pour toute une série de trucs techniques très puissants

Sur les jointures, découvrez la syntaxe SQL 2.0 recommandée :
http://sql.developpez.com/sqlaz/jointures/

Pour en savoir plus sur les index :
http://tecfa.unige.ch/guides/mysql/fr-man/manuel_MySQL_indexes.html#MySQL_indexes

Optimisation :
http://dev.mysql.com/doc/refman/5.0/fr/mysql-optimization.html
http://www.trucsweb.com/ASP/trucs.asp?no=104&type=7

Pour les contraintes, les relations et les trigger, une initiation :
http://www.commentcamarche.net/sql/sqlcontr.php3

Tous ses liens ont été contrôlés le 18 septembre 2007.

4 PHP et MySQL

Nous avons vu comment intégrer dans une page dynamique de l'information issue d'un formulaire. Nous allons maintenant gérer avec notre page dynamique l'information issue d'une base de données.

Pour cela, nous allons afficher le contenu de la base de données que nous avons définie dans la section « premiers pas en SQL ». Auparavant, nous devons apprendre à nous connecter depuis PHP sur la base de données.

Pour les exercices, nous vous recommandons de charger la base de données qui se trouve dans le fichier **`BDexercice.txt`**.

4.1 Connexion à la base de données

Avant de lancer une requête vers une base de données, nous devons ouvrir une connexion depuis PHP vers cette base. Pour voir comment on accède à la base de données, regardons ce que contient le fichier **`connexion.php`** :

```
mysql_connect("adresseDuServeur","monLogin","monMotDePasse")
        or die ("connexion impossible") ;
mysql_selectdb("nomDeLaBaseDeDonnées")
        or die ("base de données introuvable") ;
```

Pour accéder à ma base de données, je vais exécuter toujours le même code. Plutôt que de répéter partout ces deux lignes, je les ai mises dans ce fichier **`connexion.php`**, et pour lancer la connexion, je n'aurai besoin que d'appeler le fichier.

4.2 Fonction include() et partage de code

Pour appeler le code du fichier **`connexion.php`**, j'utilise la fonction **`include()`** :

```
include("connexion.php") ;
```

Comme ma connexion me prend seulement 2 lignes, vous pourriez penser que je me donne bien du mal pour économiser juste une petite ligne. En réalité, les bénéfices de l'opération vont bien au-delà.

En effet, imaginez que je décide de changer de mot de passe. Si j'avais mis les deux lignes de connexion dans tous mes fichiers PHP, il faudrait que je change dans tous les fichiers, au risque d'oublier d'en supprimer un. Ici, il suffit que je fasse la modification sur un seul fichier pour que toutes mes applications la prennent en compte.

Vous pouvez consulter la documentation de PHP pour comprendre la différence entre les fonctions **`include()`**, **`include_once()`**, **`require()`** et **`require_once()`**. Ici, on aurait pu utiliser **`require()`** à la place de **`include()`**.
Une référence à lire : http://frederic.bouchery.free.fr/?2004/07/20/7-RequireOuInclude

4.3 Afficher le contenu de la base

4.3.1 Extraction d'une table

On commence très simple avec l'affichage de la table des clubs :

```
$repClub = mysql_query("SELECT nom,codePostal FROM club") ;
while ($club = mysql_fetch_object($repClub) )
  echo "$club->nom ($club->codePostal) <br>" ;
```

La première ligne envoie à la base de données une requête **SELECT**. Le résultat de cette requête est stockée dans la variable **$repClub**. Ce résultat est un tableau qui n'est pas directement exploitable, mais que nous allons parcourir au moyen d'une boucle **while**.

Comment fonctionne cette boucle **while** ? Il y a en fait deux instructions condensées dans la même ligne :

1- la fonction **mysql_fetch_object()** extrait du tableau un club qui n'a pas encore été traitée. Elle stocke ce club dans la variable **$club**. C'est l'équivalent de l'instruction suivante :
```
$club = mysql_fetch_object($repClub) ;
```

2- La boucle **while** vérifie que l'objet **$club** est un objet bien définie. La boucle :
```
while ($club)
```
s'arrête dès que l'objet **$club** n'est pas un vrai objet.

Ainsi, à chaque itération de la boucle, **$club** reçoit une nouvelle valeur tirée du résultat de la requête. Lorsque toutes les valeurs ont été sorties de **$repClub**, **$club** reçoit la valeur **NULL**, et la boucle s'arrête.

C'est difficile à décoder lorsqu'on n'a jamais vu ce type d'instruction, mais c'est extrêmement puissant et très utilisé. Relisez bien les trois lignes de code de ce paragraphe, avec les explications, jusqu'à être sûr de bien avoir compris.

Quel est le résultat final ? Nous avons écrit chaque nom de club suivi de son code postal entre parenthèse. Les clubs sont séparés les uns des autres par la balise **
**, c'est-à-dire par un saut de ligne.

A l'écran, nous devons avoir quelque chose qui ressemble à ceci :

Les nageurs du Loiret (45200)
Les barboteurs de Vesoul (70000)
Le Plongeon d'Ajaccio (20200)
Les Canards du val de Seine (75005)

4.3.2 Restriction sur une table

Maintenant, pour chaque club, nous allons afficher ses membres. Le nom de chaque club apparaîtra comme un titre, et les membres seront affichés dans un tableau : civilité, nom, prénom, date de naissance.

Voici le code qui permet cet affichage :

```
$repClub = mysql_query("SELECT id,nom,codePostal FROM club") ;
while ($club = mysql_fetch_object($repClub) )
{
  echo "<h1 class='nomClub'> $club->nom ($club->codePostal) </h1>" ;
  echo '<table border=1>' ;
  $repMembre = mysql_query(
    "SELECT nom,prenom FROM membre WHERE idClub=$club->id") ;
  while ($memb = mysql_fetch_object($repMembre) )
    echo "<tr><td> $memb->nom </td><td> $memb->prenom </td></tr>" ;
  echo '</table>' ;
}
```

Nous avons ici deux boucles imbriquées. La première boucle reprend ce que nous avons fait précédemment : après l'extraction de la table des clubs, la boucle affiche tous les clubs les uns à la suite des autres. Le seul point qui change dans l'extraction, c'est que notre requête porte aussi sur la colonne **`id`**, car cet identifiant va nous permettre de retrouver les membres de chaque club.

En effet, pour chacun des clubs décrits par la boucle externe, on fait une extraction de la table des membres. La clause WHERE entraîne une restriction : on ne prend que les membres du club en cours de traitement. Ensuite, la boucle interne affiche les informations de chaque membre.

Exercice 10 :

a. Relisez bien le code ci-dessus, avec les explications, jusqu'à être sûr de bien avoir compris.
b. Faites le tourner dans une page PHP : il faut ajouter les informations de connexions.
c. Ici, nous n'avons mis que les noms et prénoms de chaque membre. Modifiez le code PHP pour ajouter les civilités en première colonne, et la date de naissance en dernière colonne.

4.3.3 Extraction d'une jointure

Exercice 11 :

a. Modifiez le code PHP précédent pour ajouter, entre le nom du club et la liste des membres, une ligne qui donne le nom du président de ce club (aide : il faut modifier la première requête, en y introduisant une jointure).
b. Vous devez observer alors que les clubs qui n'ont pas de président n'apparaissent pas sur la page.
c. Cherchez sur internet comment on fait une jointure externe (syntaxe LEFT JOIN) et à quoi elle sert. Appliquez ces informations pour afficher tous les clubs, avec leur président, y compris les clubs qui n'ont pas de président.

Corrigé

Exercice 11 (suite) :

d. Modifiez le code PHP précédent pour ajouter, avec le nom du président, le nom du secrétaire. Faites l'exercice avec une jointure interne, puis avec une jointure externe.

4.4 Agir sur les données

Pour modifier ma base (introduire de nouveaux éléments, mettre à jour, supprimer des éléments) je vais constituer un outil d'administration. Cet outil d'administration va être constitué de formulaire.

Nous allons créer un formulaire différent pour chaque opération possible :

- ajouter un club
- ajouter un membre dans un club
- saisir le président d'un nouveau club, ou changer le président d'un club existant.
- etc.

Pour faire ce travail de saisie, nous allons utiliser deux pages php :

- **`admin.php`** sera la page avec tous les formulaires
- **`admin2.php`** sera la page qui traitera les informations issues des formulaires.

Pour identifier dans la page **`admin2.php`** le formulaire de **`admin.php`** qui a posté l'information, chaque formulaire va contenir un champ "**`hidden`**", qui permettra au serveur d'identifier l'opération demandée par le client.

4.4.1 Ajouter un club

Pour ajouter un club, nous devons entrer :

- son nom
- son code postal

Pour ces deux champs, nous allons utiliser des champs de type **`text`**.

Nous nous occuperons de l'identité du président plus tard.

Voici le formulaire correspondant, qui va apparaître dans la page **`admin.php`** :

```
<H2>Ajouter un club </h2>
<form action="admin2.php" method=POST>
  <input type=hidden name=operation value="ajout club">
  nom du club : <input type=text name=nomClub>
  code postal : <input type=text name=codePostal>
  <input type=submit value="Valide ce club">
</form>
<hr width=50%>
```

Trois variables sont transmises au serveur :

- les variables **`nomClub`** et **`codePostal`** portent les valeurs qui ont été saisies par l'utilisateur,

- la variable **nomClub** porte la valeur **"ajout club"**. Cette valeur permet de savoir quel est le formulaire qui a été mis en œuvre, et donc quelle est l'opération d'administration que le serveur doit effectuer.

Comme l'attribut **method** du formulaire vaut **POST**, les variables transmises par le formulaire seront rangées dans le tableau **$_POST**.

Dans la page **admin2.php** nous aurons donc le traitement de ce formulaire :

```
$operation = $_POST["operation"] ;
if ($operation=="ajout club")
{
  $nomClub    = $_POST["nomClub"] ;
  $codePostal = $_POST["codePostal"] ;
  $query = "INSERT INTO club(nom,codePostal)
            VALUES('$nomClub','$codePostal')" ;
  if ( $nomClub && $codePostal && mysql_query($query) )
       $message = "opération réussie" ;
  else $message = "échec" ;
}
echo $message ;
```

La condition **if ($nomClub && …)** illustre le fonctionnement de l'opérateur **&&**. Il permet :

- de tester successivement si les variables **$nomclub** et **$codePostal** existent et ne sont pas vides
- d'exécuter la requête
- de tester si la requête s'est bien passée.

Si l'une des premières étapes échoue, la requête n'est pas exécutée. Si la requête est exécutée, et correctement exécutée, on affich « opération réussie ». Sinon, on signale l'échec.

4.4.2 Ajouter un membre

Lorsque nous ajoutons un membre, outre son nom, son prénom, nous devons spécifier :

- sa civilité (Mr/Mme/Mlle)
- le club auquel il appartient.

Fixer la civilité, c'est choisir l'une des trois possibilités. Spécifier un club, c'est faire un choix dans la liste des clubs. Dans les deux cas, nous allons donc faire un choix dans une liste. Pour cela, nous allons utiliser l'objet **<select>**.

> Attention, il existe un objet **<select>** en HTML, et une instruction **SELECT** en SQL. C'est le même mot, pour deux concepts qui n'ont absolument rien à voir. Prière de ne pas mélanger !

Si la différence entre les deux concepts n'est pas claire pour vous, prenez un bout de papier et écrivez tout ce que vous savez sur l'objet **<select>** en HTML, et sur l'instruction SELECT en SQL.

Reprenons. Voici à quoi doit ressembler le code du formulaire qui va nous permettre de saisir un nouveau membre :

```
<H2>Nouveau membre </H2>
<form action="admin2.php" method=POST>
  <input type=hidden name=operation value="ajout membre">
```

```
  civilité : <select name=civ>
    <option value=Mr> Mr
    <option value=Mme> Mme
    <option value=Mlle>Mlle
  </select>
  nom du membre : <input type=text name=nomMembre>
  prénom : <input type=text name=prenom>
  club : <select name=club>
    <option value=1> Les nageurs du Loiret
    <option value=2> Les barboteurs de Vesoul
    <option value=3> Le Plongeon d'Ajaccio
    <option value=4> Les Canards du val de Seine
  </select>
  <input type=submit value="Valide ce membre">
</form>
<hr width=50%>
```

Le gros problème, avec ce formulaire, c'est que la liste des clubs peut bouger. Si je rajoute un club dans ma base de données, il devrait apparaître comme nouvelle option dans mon **`<select>`**. Faudrait-il alors que je modifie à la main mon formulaire ? Hors de question !

Nous allons demander à PHP de générer automatiquement les options du **`<select>`**.

Nous allons donc remplacer le **`<select name=club>...</select>`** par le code suivant :

```
  <select name=club>
  <?php
    $repClub = mysql_query("SELECT id,nom FROM club") ;
    while ($club = mysql_fetch_object($repClub) )
      echo "<option value=$club->id > $club->nom " ;
  ?>
  </select>
```

Le code PHP est un copié/collé du tout premier code que nous avons écrit (l'affichage du contenu de la table **`club`**). Nous introduisons ensuite deux modifications :

- au lieu de sélectionner le nom du club et son code postal, nous sélectionnons son nom et son identifiant.
- au lieu d'afficher le nom et le code postal du club, nous écrivons un objet **`<option>`**. Cet objet affichera le nom du club, et pour cette option la valeur postée par le formulaire sera l'identifiant du club.

La liste des civilités, elle, est fixe. Elle n'a pas besoin d'être générée par PHP.

L'intégration dans la base de données est très simple. C'est une simple transposition du code utilisé pour insérer un nouveau club :

```
$operation = $_POST["operation"] ;
if ($operation=="ajout membre")
{
  $nom      = $_POST["       "] ;
  $prenom   = $_POST["       "] ;
  $civ      = $_POST["       "] ;
  $idClub   = $_POST["       "] ;
  $query = "INSERT INTO membre(nom,prenom,civilite,idClub)
            VALUES('$nom','$prenom','$civ',$idClub)" ;
  if ( $idClub && $nom && $prenom && $civ && mysql_query($query) )
```

```
        $message = "opération réussie" ;
  else $message = "échec" ;
}
else if ($operation=="ajout club")
{
  // Ici, on laisse le code développé précédemment
}
echo $message ;
```

Exercice 12 :

- dans le code ci-dessus, les indices du tableau **$_POST** ont été oubliés.

Complétez le code. Tester le.

4.4.3 Saisie du président

Pour saisir correctement un club, nous devons indiquer aussi qui est son président. Dans le formulaire correspondant, nous aurons :

- un menu **<select>** qui permet de choisir le club dont nous sommes en train de saisir le président
- un menu **<select>** qui permet de choisir le membre qu'on veut indiquer comme président de ce club
- un bouton de validation (**input** de type **submit**)

Le code que nous voulons avoir sur notre poste client va ressembler à ceci :

```
<h2>Saisir le president d'un club : </h2>
<form action="admin2.php" method=POST>
  <input type=hidden name=operation value="saisie président">
Nom du club :
<select name=club>
  <option value=""> -- choisir un club --
  <!-- ici, on met les balises <option>, une pour chaque club, -->
</select>
<br>
Nom du président :
<select name=membre>
  <option value=""> -- choisir un membre --
  <!-- ici, on met les balises <option> une pour chaque membre, -->
</select>

<br>

<input type=submit value="SAISIR">
</form>
<hr width=50%>
```

Les balises options du menu des clubs sont générées par le même code que dans le formulaire de saisie des membres :

```
  <?php
    $repClub = mysql_query("SELECT id,nom FROM club") ;
    while ($club = mysql_fetch_object($repClub) )
      echo "<option value=$club->id > $club->nom " ;
  ?>
```

Pour le menu des membres, on transpose facilement :

```
<?php
  $repMemb = mysql_query("                        ") ;
  while ($mb = mysql_fetch_object($repMemb) )
    echo
    "<option value=$mb->id > <b>$mb->nom</b>, $mb->prenom";
?>
```

Exercice 13 :

a. dans le code précédent, la requête SQL a été oubliée. Ecrivez-la.
b. écrire le code PHP à placer dans **`admin2.php`** pour traiter le formulaire de saisie du président d'un club, selon le modèle vu précédemment :

```
else if ($operation==" ???? ")
{
  ????
}
```

Correction :
Placer le code PHP suivant dans **`admin2.php`** pour traiter le formulaire de saisie du président d'un club.

```
else if ($operation=="saisie président")
{
  $idClub   = $_POST["       "] ;
  $idPres   = $_POST["       "] ;
  $query = "UPDATE club SET idPresident=$idPres WHERE id=$idClub" ;
  if ( $idClub && $idPres && mysql_query($query) )
       $message = "opération réussie" ;
  else $message = "échec" ;
}
```

4.5 Interaction SQL / PHP / javascript

Dans l'exemple précédent, le formulaire de saisie des présidents présente un gros inconvénient : nous avons mis dans un menu sélect la totalité des membres de notre fédération. Pour une toute petite association, ce n'est pas grave, mais dès que nous atteignons quelques dizaines de membres, le menu des membres devient ingérable.

Je veux donc ajouter à mon interface une fonctionnalité supplémentaire : le menu des membres ne devrait pas afficher tous les membres, mais seulement ceux du club actuellement sélectionné. Si je change le club dans le menu des clubs, toutes les options du menu des membres doivent changer.

4.5.1 Méthode sur le serveur

La première méthode pour faire cela est de faire recalculer la page par le serveur : pour cela, **lorsque je change la valeur du menu club, je force le recalcul de la page `admin.php`.**

Répétons en décomposant :

lorsque je change la valeur du menu club	=> je vais utiliser un gestionnaire d'événement **`onChange`** sur le menu des clubs.
je force le recalcul de la page `admin.php`	J'envoie mon formulaire non pas à la page **`admin2.php`**, qui intervient sur la base, mais à la page **`admin.php`** pour qu'elle calcule un autre menu des membres.

On remplace donc la balise **`<select name=club>`** par cette version :

```
<select name=club onChange="changeSelectMembre(this.form)" >
<script>
function changeSelectMembre(f) {
  f.action='admin.php';
  f.submit() ;
}
</script>
```

Lorsqu'on change la valeur du club sélectionné, on active le gestionnaire d'événement **`onChange`**, ce qui lance la fonction javascript **`changeSelectMembre()`**.

Analysons le code de cette fonction :

- Nous savons que **`this`** est l'objet représenté par la balise courante, c'est-à-dire ici le menu des clubs. **`this.form`**, le paramètre de la fonction, est le formulaire auquel appartient ce menu des clubs. Dans la fonction, **`f.action`** est donc l'attribut **`action`** de ce formulaire. La première instruction, **`f.action='admin.php'`**, change l'adresse de destination du formulaire.
- La seconde instruction, **`f.submit()`** poste le formulaire vers le serveur.

Que va faire le serveur ? Il doit afficher le menu des membres en fonction du club sélectionné. Si il n'y a pas de club sélectionné, il n'affiche rien dans la liste des membres. Voici donc le code PHP qui permet de générer le menu des membres :

```
Nom du président :
<select name=membre>
  <!-- ici, on met les balises <option> une pour chaque membre, -->
  <?php
    $idClub=$_POST("club"] ;
    $query="SELECT id,nom,prenom FROM membre WHERE idClub=$idClub";
    $repMemb = mysql_query($query) ;
    if ($idClub && mysql_num_rows($repMemb))
    while ($mb = mysql_fetch_object($repMemb) )
      echo
      "<option value=$mb->id > <b>$mb->nom</b>, $mb->prenom";
    else echo '<option disabled>
      -- choisir un club avant de choisir un membre --' ;
  ?>
</select><br>
```

Au lieu d'afficher dans le menu **`<select>`** la totalité des membres, la requête est restreinte aux seuls membres qui appartiennent au club d'identifiant **`$idClub`**. La valeur de **`$idClub`** est transmise par le formulaire, on la récupère dans **`$_POST["club"]`**.
Si cette valeur n'existe pas, cela signifie qu'aucun club n'est sélectionné : on affiche alors une invitation à sélectionner un club.

Exercice 14 :

a. à quoi sert l'attribut **disabled** ?
b. Modifiez les fichiers **admin.php** et **admin2.php** pour faire fonctionner cet exemple. Observez : lorsqu'on sélectionne un club, les membres du club apparaissent dans le select des membres (c'est ce qu'on voulait). Mais on perd le club qu'on avait sélectionné.
c. Modifiez le code PHP qui génère le select des clubs pour préselectionner le club qui a été transmis par **$_POST**. Pour préselectionner une option d'un sélect, on écrit : **<option selected value= etc.>**

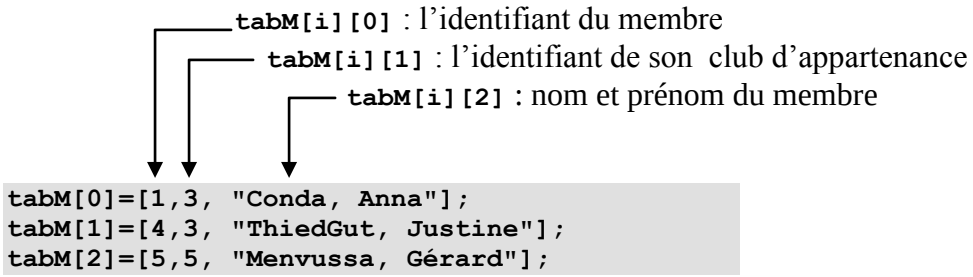
> L'inconvénient de la méthode présentée ici, c'est qu'elle entraîne un rechargement de la page alors que seul un menu <select> va changer. Ce rechargement entraîne une consommation de CPU sur le serveur, et surtout un rafraîchissement du navigateur fatigant pour l'utilisateur si ce dernier doit multiplier l'opération.

Si vous testez le script sur votre serveur local, vous ne vous rendrez peut-être pas compte du coût de ce rechargement. Si vous l'essayez sur un serveur distant, en particulier avec un réseau de faible capacité, vous vous convaincrez du problème que cela pose.

4.5.1.1 Méthode en javascript

> Plutôt que de recharger la page, nous allons utiliser javascript pour modifier le menu des membres. Pour cela, les informations sur chaque membre seront rangées comme élément d'un tableau javascript. Javascript lira ce tableau pour générer un nouveau menu select.

Le tableau ressemblera à ceci :



Pour changer le menu des membres en javascript, nous allons :

- détruire les options du menu existant
- reconstruire les options correspondant au club sélectionné

Voici le code, tel qu'il doit se présenter sur le poste client :

```
<h2>Saisir le président d'un club : </h2>
<form action="admin2.php" method=POST>
  <input type=hidden name=operation value="saisie Président">
```

```
Nom du club :
<script>
tabM = new Array() ;
tabM[0]=[1,3, "Conda, Anna"];
tabM[1]=[4,3, "Thiedgut, Justine"];
tabM[2]=[5,5, "Menvussa, Gérard"];
//etc.

function changeSelectMembre(idClub) {
//alert(1) ;
  selMembre = document.getElementById("selMembreSaisiePresident") ;
  selOpt = selMembre.options ;
  // selOpt, c'est le tableau qui contient
  // toutes les options du menu des membres

  // on vide le select de ses options
  for (i=selOpt.length-1 ; i>-1 ; i--)
    selOpt[i]=null ;

  // et on le remplit de nouveau
  selOpt[0]= new Option("-- choisir un président --","");
  for (cpt=0 ; cpt<tabM.length ; cpt++ ) if (tabM[cpt][1]==idClub)
  {
    var o=new Option(tabM[cpt][2],tabM[cpt][0]);
    selOpt[selOpt.length]=o;
  }
}
</script>
<select name=club onChange="changeSelectMembre(this.value)" >
  <option value=""> -- choisir un club --
  <!-- ici, on met les balises <option>, une pour chaque club, -->
  <option value="1"> nom du club 1
  <option value="2"> nom du club 2
</select>
<br>
Nom du président :
<select name=membre id=selMembreSaisiePresident >
  <option disabled>
      -- choisir un club avant de choisir un membre --
</select>
<br>
<input type=submit value="SAISIR">
</form>
```

Que fait ce code ? Décrivons pas à pas son activité :

- lorsque la valeur du menu des clubs change, le gestionnaire d'événement **`onChange`** lance la fonction **`changeSelectMembre()`**.Cette fonction porte un paramètre, qui est la valeur de l'options nouvellement sélectionnée : **`this.value`**. Cette valeur, nous savons que c'est l'identifiant du club (si vous en doutez, relisez les pages où nous avons construit le menu des clubs).
- la fonction **`changeSelectMembre()`** commence par récupérer le menu des membres, puisque c'est lui que nous devons modifier. Pour cela, nous avons donné un identifiant (**`selMembreSaisiePresident`**) dans le code HTML, et dans la fonction javascript, nous récupérons l'objet qui porte cet identifiant grâce à la fonction **`getElementById()`**.
- Nous allons manipuler les options de ce menu des membres, nous donnons un nom au tableau dans lequel ces options sont rangées : nous l'appelons **`selOpt`**.

- Nous vidons **selOpt**, au moyen d'une boucle en décrémentation.
- Nous construisons les nouvelles options, et nous les rangeons dans le tableau. Nous ne construisons une nouvelle option que pour les membres dont l'identifiant du club (**tabM[i][1]**) est égal à l'identifiant du club sélectionné dans le menu des clubs (**idClub**, le paramètre de la fonction, c'est à dire **this.value** dans l'appel de la fonction).

Exercice 15 :

a. identifiez, pour chacun des points de la description ci-dessus, les lignes de code concernées.
b. sauvez le fichier et testez-le dans votre navigateur.

Naturellement, cela marche, mais nous n'avons pas tout à fait fini : pour le moment, le tableau **tabM**, où sont rangées les informations sur les différents membres de chaque club, est un tableau d'exercice. Il est purement statique, et ne contient pas la totalité des membres de la base de données.

Nous allons donc transformer notre page statique en page dynamique, en remplaçant les affectations du tableau :

```
tabM[0]=[1,3,'Conda, Anna'];
tabM[1]=[4,3,'Thiedgut, Justine'];
tabM[2]=[5,5,'Menvussa, Gérard'];
```

par un code PHP qui va extraire les informations de la table membre pour les écrire sous la forme de tableau javascript. Voici ce code PHP :

```
<?php
include("connexion.php") ;
$rep=mysql_query("SELECT                    FROM membre") ;
$i=0 ;
while( $mb=mysql_fetch_object(      ) )
{
  $nomEtPrenom = "$mb->nom, $mb->prenom" ;
  echo "tabM[$i]=[$mb->id,$mb->idClub,'$nomEtPrenom'];" ;
  $i++;
}
?>
```

Exercice 16 :

a. dans le code PHP ci-dessus, remplacer les blancs pour que le programme fonctionne.
b. modifier le code pour que le menu des membres ne propose comme président que les membres majeurs.
c. modifier le code pour que le menu des membres, lorsqu'il est construit dans la fonction **changeSelectMembre()**, affiche par défaut le président actuel.
   Indication : il faut procéder en deux temps – d'une part extraire l'information sur le président de chaque club et l'intégrer dans la page, d'autre part à l'aide de cette information forcer l'option du président du club à apparaître sélectionnée.

Attention, lorsque vous passez du statique au dynamique, n'oubliez pas de mettre votre fichier PHP dans la racine web de votre serveur, faute de quoi le code PHP ne sera pas interprété par le serveur.

> L'inconvénient de cette seconde méthode, c'est qu'elle oblige à faire une extraction complète de la table membre, et à tout transmettre au client. A partir de 1000 items, la page commence à devenir lourde pour les navigateurs un peu anciens, et à partir de 10.000 items, elle est carrément dangereuse.

4.5.1.2 Méthode en Ajax

Chacune des deux méthodes que nous avons vues précédemment présente ses avantages et ses inconvénients :

- avec javascript, nous avons un meilleur confort dans la manipulation de la page, puisqu'on évite le rafraîchissement complet de la page. Mais le chargement de tous les objets et leur manipulation dans javascript est dangereux.
- avec PHP, nous n'avons extrait de la base que les objets qui nous intéressaient : les membres appartenant au club sélectionné. Mais nous avons un rafraîchissement de page peu ergonomique.

Ajax va permettre de prendre les avantages de chacune des deux méthodes :

- avec javascript, nous allons faire une requête HTTP vers un script PHP sur le serveur. Ce script PHP va lire dans la base les informations qui nous intéressent (uniquement les membres du club sélectionné).
- javascript, avec ces informations, va modifier le menu des membres, et seulement ce menu : la page ne subit pas de rafraîchissement.

> Si vous êtes intéressé par AJAX, reportez vous au TP sur Ajax.

5 Cookies et sessions

Vous trouverez très facilement des documentations sur internet qui vous permettront de programmer en PHP l'utilisation de cookies et de sessions.

Ce que nous allons expliquer ici, ce n'est pas comment on fait, mais comment cela fonctionne, pour vous permettre

1- de maîtriser ce que vous faites.
2- d'accélerer votre compréhension

Nous commençons avec les cookies. Les sessions sont une ergonomie particulière bâtie autour d'un cookie particulier.

5.1 Les cookies – principe de fonctionnement

Nous avons dit dans le cours n°1 que les cookies sont des variables qui sont postées du serveur vers le client, et que le client renvoie au serveur à chaque nouvelle requête. La figure suivante illustre ce fonctionnement :

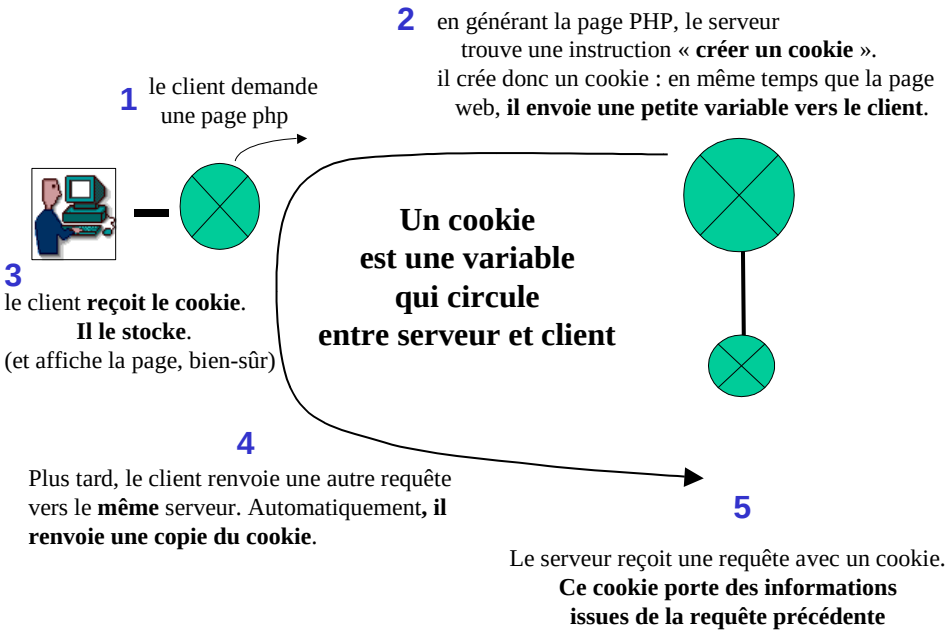


figure 4. La circulation du cookie.

A quoi cela sert-il ?

Nous devons nous rappeler que toutes les requêtes vers un serveur sont indépendantes les unes des autres : le serveur reçoit une requête, il y répond, et lorsqu'il a finit de répondre, il passe à une autre tâche (répondre à la requête d'un autre client) en effaçant tout ce qu'il vient de calculer pour le premier.

Ainsi, plus tard, lorsque le même client renverra une nouvelle requête, le serveur ne possèdera pas de trace de la première. Il n'a pas les moyens de garder cette mémoire. Le seul moyen

pour lui est de faire transiter cette mémoire par le client. Le cookie est cette mémoire des choses importantes que le serveur stocke chez le client pour qu'elle soit disponible pour toutes les requêtes futures (dans la limite de durée de validité du cookie, et sous réserve que le client accepte les cookie).

Prenons un exemple : l'utilisateur remplit un formulaire avec son login et son mot de passe. Le serveur lui répond : « bravo, votre mot de passe est reconnu, voulez-vous voir vos mail ? ». Si l'utilisateur cherche à voir ses mails, il doit transmettre en même temps une preuve de son identité. Sinon, n'importe qui pourrait consulter les mails de n'importe qui.

Transmettre une preuve de son identité : il vient pourtant de le faire à la première requête, en entrant son login et son mot de passe. Le cookie est donc le moyen pour le navigateur d'authentifier l'identité de l'utilisateur sans avoir à lui demander son mot de passe à chaque chargement de.

5.2 Les cookies dans l'organisation de la page web

Le cookie est véhiculé dans l'entête de la réponse HTTP, et la page web est véhiculée dans le corps de cette réponse. Attention, nous parlons ici de l'entête et du corps du message HTTP. Il ne faut pas confondre avec l'entête et le corps de la page web. Prière de ne pas mélanger ! la figure suivante illustre cette emboîtement :

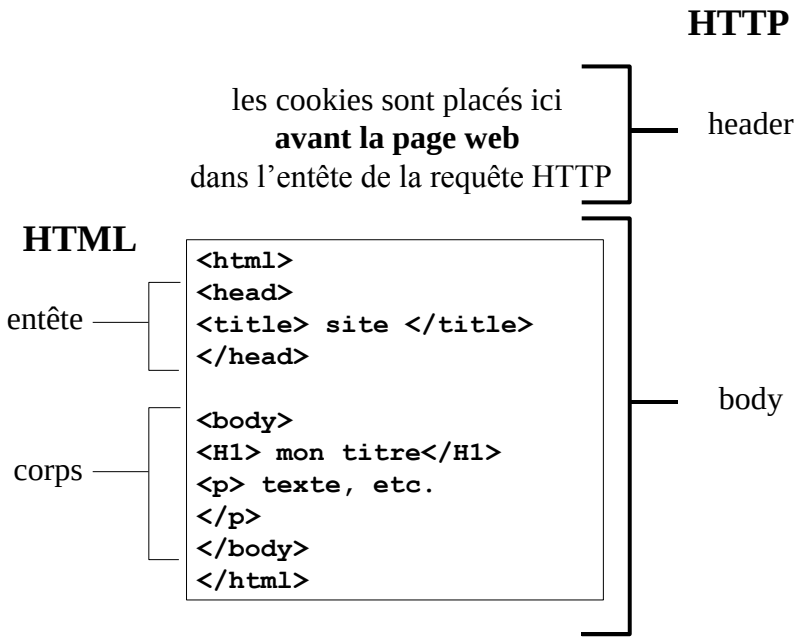


figure 5. La page web se trouve toute entière dans le corps de la réponse HTTP.
Les cookies sont stockés dans l'entête de cette réponse.

Quelle est l'importance de cette architecture pour la conception d'une page PHP ?

> Dès que vous commencez à écrire du texte HTML, vous ne pouvez plus écrire un cookie : au moment où vous commencez à écrire du texte HTML, le serveur ferme l'entête et ouvre le corps de la réponse, pour y mettre la page web.

Cela signifie en particulier que si vous utilisez des cookies, vous ne devez même pas mettre une ligne blanche ou un espace avant l'ouverture de la balise du code PHP (`<?php`). Ceci est une source d'erreurs fréquente et difficile à tracer si on n'y pense pas.

La bonne architecture
de la
pour mettre en ligne une
Mettre en premier lieu tous le code PHP.

> Effectuer en premier lieu tous les traitements nécessaires par le calcul de la page, et ne lancer l'affichage que quand tout est prêt.

Idéalement, les calculs sont dans une page php séparée, appelée par un `include()` au début de la page principale. La page principale est constituée du code HTML, avec quelques insertions PHP appelées par la fonction `echo`.

5.3 Les sessions – principe de fonctionnement

Plus utiles que les cookies, les sessions permettent de stocker un grand nombre d'informations sur le serveur, et de n'envoyer qu'un seul cookie qui permet de retrouver toutes les informations.

La figure suivante illustre ce fonctionnement des sessions.

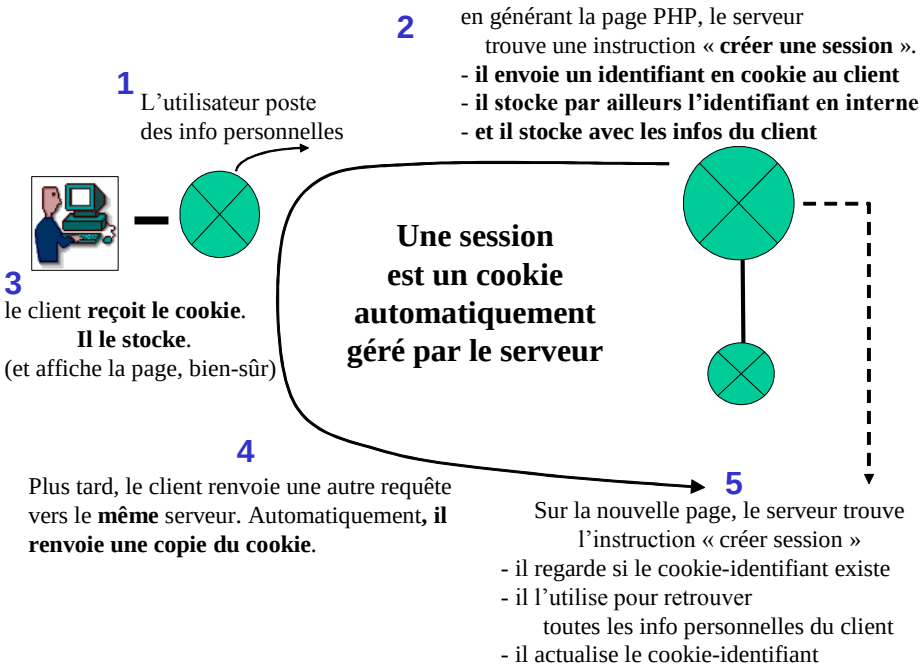


figure 6. Les sessions permettent de n'envoyer qu'un seul cookie, un identifiant. Les autres informations sont stockées sur le serveur, et c'est l'identifiant qui permet de les récupérer de manière sécurisée.

L'utilisation des sessions encapsule totalement la création du cookie identifiant : lorsqu'on crée une session, on teste l'existence du cookie, s'il existe on récupère les variables de la session précédente dans le tableau `$_SESSION`. Si le cookie n'existait pas, on le crée. récupère les variables de la session précédente.

Comme pour les cookies, la création de la session doit se faire avant l'écriture du code HTML. En revanche, une fois que la session est crée, on peut stocker des informations sur le serveur comme on le souhaite

Exercice 17 :

- sur internet, trouvez une documentation portant sur les sessions en PHP
- protégez avec un mot de passe le formulaire d'administration de la base des clubs et des membres, et programmez l'accès à cette page avec des sessions.

Pour aller plus loin, vous pouvez consulter la page suivante :
http://php.developpez.com/cours/sessions/?page=plus#L3
Vous y apprendrez, entre autre, qu'on peut aussi faire passer l'identifiant de session par la barre d'adresse du navigateur (méthode trans_sid) plutôt qu'avec un cookie. Cette opération, un peu plus risquée que le cookie en terme de sécurité, est déconseillée.

6 Conclusion

6.1 Bilan

A l'issue de ce cours, vous devez être en mesure de coder une application simple mettant en œuvre un formulaire pour extraire de l'information d'une base de données mySQL, en insérer et la modifier.

Vous devez être capable de coder une application mettant en œuvre à la fois PHP et javascript, en interaction avec une base de données.

Si l'un des points ci-dessus ne vous semble pas acquis :

- refaites les exercices
- complétez votre information sur les sites spécialisés
- posez des questions au professeur

6.2 Divers – pour aller plus loin en PHP

Si vous souhaitez aller plus loin, voici quelques pistes à explorer :

- à partir de PHP5, le langage est orienté objet.
- Outre les outils de manipulation de chaîne classiques, PHP intègre entre autre les outils basés sur les expressions régulières.

Exercice 18 :

- ajouter l'outil d'administration qui permet de supprimer un membre
- de supprimer un club (il faut vérifier que le club n'a plus de membres avant de le supprimer)
- de mettre à jour un club (modifier son nom et son adresse)
- de mettre à jour un membre (modifier son club d'appartenance, sa date de naissance).

A chaque fois, il faut ajouter un formulaire dans la page **`admin.php`**, et traiter ce formulaire dans la page **`admin2.php`**.

Exercice 19 :

Programmer un site intranet permettant de faire du co-voiturage les jours de grève. Les fonctions à assurer sont :

- inscription avec choix de mot de passe, numéro de téléphone et adresse mail.
- identification (identification avec des sessions)
- enregistrement d'un trajet en voiture, proposé aux autres usagers. On renseigne : date, lieu de départ (le lieu d'arrivée est le lieu de travail), heure de départ le matin, heure de retour le soir, et nombre de place dans la voiture.
- Réservation d'une place : le candidat passager affiche les annonces pour lesquelles il reste des places, et sélectionne celle qui l'intéresse. Dans la base, sa réservation est inscrite.

Exercice 20 :

Amélioration du site précédent – on ajoute les fonctions suivantes :

- Le candidat passager peut sélectionner les annonces par date, de manière ergonomique (=> au moyen d'un calendrier javascript, à récupérer sur internet).
- Le conducteur peut consulter son annonce, récupérer la liste des passagers déjà inscrits, avec leurs emails et leurs numéros de téléphone.
- Le conducteur peut supprimer son annonce : un mail est automatiquement envoyé à ceux qui était déjà inscrits.

7 Sommaire

1 Introduction 2
1.1 Objectif 2
1.2 prérequis 2
1.3 Méthode 2
1.4 Matériel 3
2 Premiers pas en PHP 4
2.1 L'affichage en PHP 4
2.2 Les bases du langage 5
2.3 PHP et les formulaires HTML 8
2.3.1 Le transfert de l'information 9
2.3.2 Récupération des variables dans PHP 10
2.3.3 Méthode GET et méthode POST 11
2.4 Conclusion 12
3 Premiers pas en SQL 13
3.1 Présentation 13
3.1.1 Qu'est-ce que SQL ? 13
3.1.2 Les tables 13
3.2 Modèle physique de données 14
3.2.1 Les identifiants 14
3.2.2 Modèle conceptuel de données 15
3.2.3 Conversion vers le modèle physique de données 16
3.2.4 Création des tables en SQL 17
3.3 Manipulation des données 17
3.3.1 Consulter les données : la requête SELECT..FROM 17
3.3.1.1 Restriction sur les colonnes (projection) 17
3.3.1.2 Restriction sur les lignes 18
3.3.1.3 Alias de colonne 18
3.3.2 Produit cartésien 18
3.3.3 Insertion, suppression et mise à jour 19
3.4 Les index 20
3.4.1 La recherche séquentiel 20
3.4.2 La recherche indexée 20
3.4.3 Index et identifiant : les clefs 20
3.4.4 Où dois-je créer un index ? 21
3.5 Pour aller plus loin 21
4 PHP et MySQL 23
4.1 Connexion à la base de données 23
4.2 Fonction include() et partage de code 23
4.3 Afficher le contenu de la base 24
4.3.1 Extraction d'une table 24
4.3.2 Restriction sur une table 24
4.3.3 Extraction d'une jointure 25
4.4 Agir sur les données 26
4.4.1 Ajouter un club 26
4.4.2 Ajouter un membre 27
4.4.3 Saisie du président 29
4.5 Interaction SQL / PHP / javascript 30

4.5.1 Méthode sur le serveur *30*
4.5.1.1 Méthode en javascript 32
4.5.1.2 Méthode en Ajax 35
5 Cookies et sessions 36
5.1 Les cookies – principe de fonctionnement 36
5.2 Les cookies dans l'organisation de la page web 37
5.3 Les sessions – principe de fonctionnement 38
6 Conclusion 40
6.1 Bilan 40
6.2 Divers – pour aller plus loin en PHP 40
7 Sommaire 41