Poly du TP n° 1 : Révisions de HTML, CSS, PHP et MySQL

I) La programmation Web

Un site Web fonctionne sur deux niveaux : sur le serveur et sur la machine de l'utilisateur (le client). La connexion entre les deux se fait par le protocole HTTP. Nous réviserons un certain nombre de langages pour le développement d'un site Web, chacun ayant un but spécifique.
`html` est un langage pour décrire le contenu des pages Web.
`css` est associé à `html` pour le mettre en forme. Ces deux langages sont interprétés par les navigateurs Web sur la machine client. Nous feront référence ici à la dernière version de `html`, la version 5 datant de 2014 et ayant plusieurs changements significatifs par rapport aux précédentes versions.

`php` est un langage de programmation interprété sur le serveur. Il est un des langages les plus utilisés aujourd'hui pour le développement serveur. Le serveur gère la construction dynamique des pages Web demandées par le client. À cet effet il gère les liens avec les bases de données, les sessions utilisateurs, et tout autre calcul concernant le fonctionnement du site.
Le client (navigateur) envoie des informations au serveur, qui traite le code `php` et renvoie une page Web (`html`). Ainsi l'utilisateur ne voit pas l'exécution de `php`, juste son résultat.

Plus loin dans le cours nous traiterons aussi `Javascript` et sa bibliothèque `jQuery`. `Javascript` est un langage de programmation très utilisé pour dynamiser les pages Web coté client. En effet chaque navigateur est capable d'exécuter du `Javascript` (inclus par le serveur dans la page Web renvoyée). Cela permet de déléguer une partie de la logique du site au navigateur, sans avoir besoin de communiquer avec le serveur (par exemple pour vérifier les données saisies dans un formulaire, pour dynamiser l'affichage, etc.).
`Javascript` peut également être utilisé pour le développement côté serveur comme on verra plus loin (`node.js`).

Enfin `mySQL` est un système de gestion de bases données (SGBD). Une base de données contient les informations que le site doit stocker de façon permanente (comptes utilisateurs, catalogues de vente en ligne, etc.). On y accède par exemple grâce à `php`, ou tout autre langage muni d'une bibliothèque d'interface. SQL est le langage utilisé pour la communication avec le SGBD.

II) Révisions de HTML

a) Document HTML

Un document (ou page) `html` prend l'extension `.html` ou sert de base à des pages `.php` par exemple. Voici un document `html` minimal, avec son entête (`<head></head>`) et son corps (`<body></body>`).

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <meta name="description" content="courte description">
        <title>Titre de la page</title>
        <link href="style.css" rel="stylesheet">
    </head>
    <body>
    </body>
</html>
```

Les balises (*tags*) peuvent aller par paire (`<head></head>`) et entourent un contenu, ou être seules (`<link>`). Dans les deux cas, elles peuvent avoir des *attributs* indiqués avec la syntaxe `attribut="valeur"`.
Pour une liste des balises et des attributs existants, on pourra se référer à http://www.w3schools.com/tags.

b) Head

Les balises dans l'entête servent à donner des méta-informations sur la votre page. Voici quelques balises importantes pouvant être mises dans `<head></head>` :

- `<title></title>` Le titre de la page. C'est le petit texte qui apparaîtra dans l'onglet de votre navigateur ou dans le titre d'un résultat d'un moteur de recherche. C'est un élément obligatoire en `html`.
- `<meta charset="UTF-8">` Donne l'encodage de la page, c'est une information non obligatoire mais essentielle. Avant HTML5, la balise ressemblait à :
  `<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />`.
- `<meta name="description" content="Brève description de votre table" />`. Cette brève description peut être utilisée par les moteurs de recherche ou les réseaux sociaux.

D'autres balises existent mais certaines sont obsolètes comme `keywords` qui servait aux moteurs de recherches à référencer la page. D'autres comme `author` et `copyright` ont une faible utilité et ne sont visibles que pour les développeurs regardant le code source. Afin de protéger un site, il vaut mieux utiliser une vraie licence, par exemple *creative commons*.

c) Éléments de base d'une page html

Texte. L'élément de base d'une page `html` est souvent le texte. Des balises peuvent être utilisées pour le mettre en forme (par exemple `<em></em>` pour mettre en évidence — souvent en italique par défaut). Elles ne doivent être utilisées que ponctuellement. Si un motif de mise en forme se répète dans le texte, il est conseillé d'utiliser du `css`, plus modulable et aidant la page à garder une unité.

Mise en page. Pour la mise en page du texte on peut utiliser les balises préexistantes (`<p></p>` pour les paragraphes, `<ul><li>element1</li><li>element2</li></ul>` pour les listes non numérotées, `<ol><li>element1</li><li>element2</li></ol>` pour les listes numérotées, etc.) ou en créer des nouvelles, en utilisant les balises génériques `<div>` et `<span>` et en les personnalisant grâce au code `css`.
`<br />` permet de sauter une ligne.

Liens et images. Pour introduire un lien :

```
<a href="http://www.exemple.com">Ce lien est un exemple</a>
```

et une image :

```
<img src="logoP-7.jpg" alt="Le logo de Paris 7" />
```

`href` étant l'adresse cible, `src` l'adresse de l'image et `alt` le texte affiché si l'image ne s'affiche pas.

d) Structure d'une page

Les propriétés des différentes balises permettent de positionner des éléments dans une page. Il existe deux grands types de balise du point de vue de la mise en page : les *blocs* et les *inlines*.
Les balises blocs sont positionnées en introduisant un retour à la ligne avant et après. Les balises paragraphes `<p></p>` sont un exemple de balise *bloc*. La balise *bloc* générique est `<div></div>`.
Les balises *inline* en revanche n'introduisent aucun retour à la ligne. En font partie les balises liens `<a></a>`. La balise *inline* générique est `<span></span>`.
Depuis HTML5 existent des balises sémantiques qu'il est recommandé d'utiliser. Elles ne sont pas définies par la façon dont elles doivent apparaître (fond bleu, texte rouge, bloc centré dans la page, etc.) mais par ce qu'elles doivent contenir. Les balises sémantiques les plus importantes sont :

- `<header></header>` contiendra l'entête de la page, Il s'agit de l'entête affiché. Il ne doit pas être confondu avec `<head></head>` qui contient les informations (non-affichées) sur la page ;
- `<footer></footer>` contiendra le pied de page, éventuellement le pied de chaque section ;
- `<nav></nav>` introduit une section de la page dédiée aux liens de navigation ;
- `<section></section>` Introduit une section générique de la page. À la tête de chaque section il y aura habituellement un titre (`<h1></h1>` étant le plus grand, `<h6></h6>` le plus petit) qui définit le contenu de la section ;
- `<article></article>` servira à présenter du contenu pouvant exister indépendamment du reste de la page. C'est par exemple un article de blog, un message, une description de produit etc.

Toutes les balises sémantiques s'affichent de la même façon au premier abord. Les différences entre elles seront apportées par le code `css` associé à la page. Elles sont un moyen de clarifier le rôle des différentes partie de la page page.

e) Formulaires

Un formulaire est introduit par les balises : `<form></form>`. Elles ne servent pas (nécessairement) à faire de la mise en page, mais à délimiter les éléments de saisie dont le contenu sera soumis au serveur. La balise `<form></form>` précisera la méthode d'envoi des données (attribut `method`) : `POST` ou `GET`. Ces deux méthodes sont détaillées plus bas. La balise `<form></form>` précisera également la page à laquelle doivent être envoyées les données saisies dans le formulaire (attribut `action`).
Les différentes balises de saisie sont les suivantes.

- Champ de texte (mono-ligne) :
  `<input type="text" name="NomDuChamp" value="Préremplissage">`.
  L'attribut `name` permettra d'identifier le champ chez le serveur. L'attribut `value` est un petit texte pré-rempli dans la zone de saisie et disparaissant dès qu'on sélectionne cette zone.
- Champ mot de passe : `<input type = "password" name="Nom">`. Le mot de passe ne s'affiche pas, des étoiles remplacent les caractères saisis.
- Zone de texte (multiligne) : `<textarea name="Nom"></textarea>`. Sa taille pourra être réglée en css.
- Case à cocher : `<input type="checkbox" name="Nom" />`. Un carré que l'on peut cocher ou non.
- Champs d'option :

```
<input type="radio" name="Nom" value="1" />
<input type="radio" name="Nom" value="2" />
<input type="radio" name="Nom" value="3" />
```

Plusieurs balises ont le même nom mais pas la même valeur, l'utilisateur ne pourra en cocher qu'une.

- Liste déroulantes :

```
<select name="Nom">
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
</select>
```

Les valeurs affichés sont ceux contenus entre les balises `<option><option/>` (et non pas les valeurs des attributs `value`).

- Bouton d'envoi : `<input type="submit" value="Envoyer" >` pour envoyer votre formulaire à la page `action` par la méthode `method`.

Tous ces champs de saisie ne sont accompagnés d'aucun texte. Par exemple `<input type="checkbox" name="Nom" />` affichera une simple case à cocher. Pour y associer un texte, on peut utiliser la balise `label`.

```
<input type="checkbox" name="Nom" /><label>Case à cocher</label>
```

Les différents éléments d'un formulaire peuvent être organisés soit utilisant des blocs (`<div>`, `<span>` , etc.), soit avec les balises `<fieldset></fieldset>`.
Voici un exemple de formulaire :

```
<form method="post" action="traitement.php">

   <fieldset>
       <legend>Vos coordonnées</legend> <!-- Titre du fieldset -->

       <label for="nom">Quel est votre nom ?</label>
       <input type="text" name="nom" id="nom" />

       <label for="prenom">Quel est votre prénom ?</label>
       <input type="text" name="prenom" id="prenom" />
    </fieldset>

   <fieldset>
       <legend>Votre souhait</legend> <!-- Titre du fieldset -->

       <p>
           Faites un souhait que vous voudriez voir exaucé :

           <input type="radio" name="souhait" value="riche" id="riche" />
            <label for="riche">Etre riche</label>
           <input type="radio" name="souhait" value="celebre" id="celebre" />
             <label for="celebre">Etre célèbre</label>
           <input type="radio" name="souhait" value="intelligent" id="intelligent" />
             <label for="intelligent">Etre <strong>encore</strong> plus intelligent</label>
           <input type="radio" name="souhait" value="autre" id="autre" />
             <label for="autre">Autre...</label>
       </p>

       <p>
           <label for="precisions">Si "Autre", veuillez préciser :</label>
           <textarea name="precisions" id="precisions" cols="40" rows="4"></textarea>
       </p>
```

```
   </fieldset>
   <input type="submit" value="Envoyer" >
</form>
```

f) Validateur et débogage

f).1 Validation

Le W3C fournit un validateur de pages `html` que l'on trouve à l'adresse http://validator.w3.org/. On peut y entrer une URL. Une page est valide si le validateur affiche "This document was successfully checked as HTML5 !". Il est fréquent qu'il y ait des warnings.

f).2 DOM

Pour l'instant nous avons traité les documents `html` de deux façons : comme un fichier texte (le code source de la page, qu'on écrit par un éditeur de texte). Et comme une page web lue par un navigateur.
Une troisième façon de représenter le document `html` est par la structure arborescente de la page, le DOM (Document Object Model). On peut visualiser et naviguer à travers le DOM via les outils de débogage du navigateur. Les langages de script coté client (`Javascript`) ont accès et peuvent manipuler le DOM pour modifier la page.

III) Révisions de CSS

Les références pourront être trouvées sur le site du W3schools.

a) Formatage de texte

Les éléments de mise en forme (style) d'une page peuvent soit être insérés directement dans le document `html`, soit être relégués dans un fichier `.css` (une *feuille de style*) qui sera lié à la page par une balise `<link rel="stylesheet" href="maFeuillle.css" />` dans le `head`.
L'insertion directe de `css` se fait via l'attribut `style`, par exemple :

```
<ul style="color:red; list-style-type:square;">
<li> Un élément </li>
</ul>
```

Cette syntaxe est vite pénible pour de nombreux attributs ou pour modifier plusieurs éléments. Pour modifier tous les éléments d'un certain type, on utilisera la syntaxe suivante, placée dans la feuille de style `css`.

```
p{
color:red;
font-size:small;
}
*{
font-family: Impact, "Arial Black", Arial, Verdana, sans-serif;
text-align: justify;
}
```

Pour une liste des différentes propriétés `css`, vous pouvez vous référer à http://www.w3schools.com/cssref/.

b) Les sélecteurs

On a trois moyens de spécifier les balises à décorer par du `css`. On peut sélectionner toutes les balises d'un type donné, par exemple toutes les balises `<p></p>` :

```
p{

}
```

On peut sélectionner une balise unique grâce à son `id`. On indiquera en `css` l'id précédé d'un dièse. Ainsi pour décorer la balise `<img src="images/logo.png" alt="Logo du site" id="logo" >`, on écrira dans la page `css` :

```
#logo{

}
```

Enfin on peut sélectionner un ensemble de balises (pas nécessairement du même type) grâce à leur classe. En `css` la classe sera précédée d'un point. Ainsi pour décorer les balises `<span class="liens"><img src="images/logo.png" alt="Logo du site" class="liens" ></span>` on ajoutera dans le `css` :

```
.liens{

}
```

Ces différents sélecteurs peuvent se combiner (`p.liens` concernera les balises `p` de classe `liens`) et être spécialisés. Une liste exhaustive des sélecteurs est disponible sur le site du W3C https://www.w3.org/TR/css3-selectors/#selectors. Nous reverrons ces sélecteurs lors de l'étude de `jQuery`.

c) Positionnement

En `css` les blocs possèdent par défaut la propriété de positionnement `position:static`, qui les affiches les uns à la suite des autres, dans l'ordre de déclaration dans le `html`. Cette suite de blocs positionnés « naturellement » s'appelle le *flux* des éléments. Il est possible de modifier ce flux.

- **Positionnement relatif**. La directive `position:relative` permet de positionner un bloc relativement à sa position naturelle et ce sans modifier le flux. Les autres blocs se comportent donc comme si le bloc positionné relativement se situait à sa place « normale ». Afin de spécifier la position relative on utilise les directives `top, left, right, buttom` qui permettent de choisir la distance relative – qui peut être négative – par rapport aux différents bords du bloc.
- **Positionnement absolu** La directive `position:absolute` permet de positionner un bloc de manière absolue dans le système de coordonnées de son premier ancêtre n'ayant pas un positionnement `static`. S'il n'en existe pas, le positionnement se fait dans le système de coordonnées de la page. Le bloc est alors retiré du flux et n'aura plus d'impact sur les autres blocs.
- **z-index** Lorsque le flux des blocs est altéré, il arrive souvent qu'il y ait de l'empiètement entre les blocs. Il est alors utile de pouvoir spécifier quel bloc doit être affiché devant ou derrière un autre. On utilise alors la propriété `z-index` qui prend pour valeur un entier. Plus cet entier sera élevé plus le bloc sera mis en avant.
- **Positionnement fixe**. La directive `position:fixed` permet de positionner un bloc de manière absolue dans le système de coordonnées de la fenêtre. Le bloc est alors retiré

du flux : il n'impactera plus les autres. Le positionnement du bloc dans le système de coordonnées de la fenêtre et non du document ou de l'un des blocs fait que la position est indépendante du défilement de la page. Ainsi le bloc apparaîtra toujours à la même distance des bords de la fenêtre.
On sera amené à préciser sa position dans la fenêtre grâce aux directives : `top, left, right, bottom`. Par exemple `position fixed; left:0px; top:50%;`.

– **Positionnement flottant** La propriété `float` permet de modifier le flux d'une manière plus complexe que la propriété `position`. En fonction de sa valeur (`left ou right`), celle-ci a pour effet de positionner les bloc dans le flux à sa position naturelle puis de le retirer et de le réinsérer le plus à droite (ou à gauche) possible, jusqu'au bord de son bloc parent ou du premier élément flottant rencontré.
La propriété `clear` permet de spécifier qu'aucun élément flottant n'est autorisé d'un côté (gauche ou droite) ou des deux côtés de l'élément. Ses valeurs sont `left, right et both`.

IV) Révisions de PHP

Le manuel de `php` peut être trouvé en français sur http://php.net/manual/fr.

a) Les bases du langage `php`

La base d'une page `.php` est un document `html`. Le code `php` sera inséré dans les balises `<?php ?>`. Par exemple :

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Ceci est une page de test avec des balises PHP</title>
            <meta charset="utf-8" />
        </head>
        <body>
            <p>
            Cette page contient du code HTML avec des balises PHP.<br />
            <?php /* Insérer du code PHP ici */ ?>
            </p>
        </body>
    </html>
```

Le langage `php` est un langage de programmation standard : il possède des variables, des fonctions, des conditionnelles, des boucles, etc. Il possède une partie orientée objet. Chaque instruction termine par un point virgule. Voici un bref rappel de ses caractéristiques de base :

– **Affichage**. Pour afficher dans la page `html` du texte (ou des balises `html`), on utilise la fonction `echo` :
`<?php echo "Ceci est du <strong>texte</strong>"; ?>`
– Les **commentaires** en `php` sont encadré par `/*` et `*/`
– **Variables**. Le nom des variables doit commencer par un symbole $. Il n'y a pas besoin de préciser le type des variables. De plus `php` est un langage polymorphe : une variable peut prendre successivement plusieurs types.

```
<?php
   $age = 17;
   echo 'Le visiteur a '. $age . ' ans<br />';
   $age = $age*2;
```

```
   echo "Le visiteur a maintenant $age ans<br />";
?>
```

- **Chaînes de caractères.** Les chaînes de caractères sont délimitées par des guillemets. Il en existe deux types en `php` : les simples `' '` qui n'évaluent pas les variables, et les doubles `" "` qui évaluent les variables. Ainsi `<?php echo "Bonjour $nom";?>` affichera `Bonjour Michel` alors que `<?php echo 'Bonjour $nom';?>` affichera `Bonjour $nom`.
  La concaténation de chaînes de caractères (ou d'entiers) est le point .
- **Conditionnelles**. La syntaxe de l'instruction conditionnelle est illustrée par l'exemple suivant :

```
<?php
   if($autorisation == true)
      {
      echo "<p>C'est vrai</p>";
      }
   elseif($autorisation == false || $autorisation == 3)
      {
      ?>
      <p>C'est faux !</p>
      <?php
      }
   else
      {
      echo '<p>C\'est un nombre.</p>';
      }
?>
```

  Remarquer qu'un bloc `php` peut être interrompu au milieu d'une instruction conditionnelle, et reouvert ensuite.
- **Boucles**. Il existe en `php` des boucles `for` et des boucles `while` similaires à celles de java.

```
<?php
    for($i = 0; $i < 100; $i++)
        {
        echo $i;
        }
    while($i > 0)
       {
       echo $i;
       i--;
       }
?>
```

- **Fonctions.** La déclaration des fonctions est réduite à sa plus simple expression : il n'y a pas de mention de type.

```
<?php
    function DireBonjour($nom)
        {
        echo "Bonjour $nom<br />";
        return true;
        }
     DireBonjour('Patrice');
 ?>
```

- **Tableaux**. Les tableaux sont aussi appelé `array` pour les différencier des tableaux `html`.

— Ils se définissent soit exhaustivement, soit valeur par valeur :

```
$tab = array(1,4);
OU
$tab[] = 1;
$tab[] = 4;
OU
$tab[0] = 1;
$tab[1] = 4;
```

Pour accéder à une valeur stockée dans un tableau, il suffit de connaître son indice `$a = $tab[0];`. Les indices entiers (qui démarrent toujours à 0) peuvent être remplacés par des chaînes de caractères, on parle alors de tableaux associatifs :

```
$tab = array("Premier" =>1,"Deuxieme"=>4);
OU
$tab["Premier"] = 1;
$tab["Deuxieme"] = 4;
```

Pour accéder à un élément d'un tableau associatif, on peut utiliser soit son indice (chaîne de caractères) , soit sa position : `$a = $tab["Premier"];` ou `$a = $tab[0];`

— Pour parcourir un tableau, on peut utiliser une boucle standard ou une boucle `foreach` :

```
foreach($tab as $element)
    {
    echo $element . '<br />';
    }
OU
for($i = 0; $i < sizeof($tab); $i++)
    {
    echo $tab[$i];
    }
OU pour un tableau associatif:
foreach($tab as $clef => $element)
    {
    echo 'À la clef ' . $clef . ' est l\'élément : ' + $element . '<br />';
    }
```

– **Fonctions de base.** `php` dispose d'une riche bibliothèque standard de fonctions. Très utiles sont les fonctions de gestion des variables, par exemple `isset($a)` est une fonction qui renvoie `true` si la variable `$a` existe, `false` sinon. Il existe aussi plusieurs fonctions sur les tableaux. À titre d'exemple `in_array($tab,$val)` renvoie `true` si `$val` est une valeur dans l'array `$tab`.
Pour d'autres fonctions consulter la référence en ligne `php` http://php.net/manual/fr/funcref.php

b) Communication entre plusieurs pages d'un site

Le client peut transmettre des données au serveur dans le cycle requête/réponse HTTP. Ces données peuvent être récupérées en `php` permettant une forme de communication à deux sens entre client et serveur.
Les données récupérées peuvent ensuite être transmises à d'autres pages, ce qui peut garantir un mécanisme de "persistance" de certaines données (comme par exemple les informations d'authentification) entre plusieurs pages d'un site. Nous verrons trois méthodes.

– `$_GET`. La première méthode consiste à utiliser les url pour transmettre des informations. Cette méthode n'est absolument pas sécurisée. Pour faire en sorte que la page `html1.php` transmette des données à la page html2.php on peut y ajouter un lien de cette forme (paramètres séparées par `&`).
`<a href="html2.php?nom=Dupont&prenom=Jean">Dis-moi bonjour !</a>`
Dans la page `html2.php`, on peut récupérer un tableau associatif nommé `$_GET` dont les indices sont les noms des paramètres et les valeurs sont les valeurs des paramètres :

```
<?php
    $nom = $_GET['nom'];
    $prenom = $_GET[1];
    echo "Vous vous appelez $prenom $nom<br />";
?>
```

– `$_POST`. Cette méthode requiert un formulaire. Le formulaire (écrit en `html`) enverra des informations dans un tableau associatif appelé `$_POST`. Ainsi dans la page `html1.php` on a par exemple :

```
<?php
    <form action="html2.php" method="post">
    <p>
         <input type="text" name="prenom" />
         <input type="submit" value="Valider" />
    </p>
    </form>
?>
```

Dans `html2.php` on peut récupérer les valeurs envoyées par le formulaire, en utilisant les noms des champs comme indices :

```
<?php
    $prenom = $_POST['prenom'];
    echo "Vous vous prénommez $prenom <br />";
?>
```

– `$_SESSION`. Les sessions permettent de transmettre à toutes les pages d'un site certaines variables (par exemple le login de l'utilisateur après authentification) . De la même façon que `$_POST` ou `$_GET`, `$_SESSION` est un tableau associatif.
On peut ajouter des valeurs au tableau `$_SESSION` dans n'importe quelle page, il sera transmis automatiquement aux autres pages. Il suffit de démarrer chaque page `php` par `session_start();`

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

<!DOCTYPE html>
<html>
    <head>
        <meta charset-"uft-8" />
        <title>Titre</title>
    </head>
    <body>
        <p>Bonjour <?php echo $_SESSION['prenom'] ?> !</p>
     </body>
 </html>
```

Dans une autre page on aura :

```
<?php
session_start();
$_SESSION['prenom'] = 'Jacques';
?>
```

V) Révisions de MySQL

Le langage `php` permet d'interagir avec une base de données gérée par un SGBD sur le serveur. `mySQL` est l'un des SGBD les plus utilisées par les applications Web. On supposera ici connus les rudiments des bases de données : forme d'une base de données relationnelle, requêtes SQL `SELECT/FROM/WHERE` et jointures simples.
L'interaction entre `php` et une base de données se fait en quatre temps.

- La connexion à la base de données grâce à l'instruction :
  `$bdd = new PDO("mysql:host=VotreHote;dbname=test;charset=uft8","root","");`
  Le premier argument contient l'hôte, le nom et l'encodage de la bdd. Le deuxième et le troisième sont le login et mot de passe `mySQL`.
- L'envoi de la requête, grâce à la méthode `query` de l'objet `$bdd` :
  ```
  $reponse = $bdd->query('SELECT *
                          FROM table1
                          WHERE col1 = ' . $var1 . ' AND col2 >= 4');
  ```
- L'exploitation de la réponse. La réponse peut avoir plusieurs lignes. Elle se présente sous la forme d'une pile. Une ligne de la réponse peut être dépilée par la méthode `fetch()`. La ligne est renvoyée par `fetch()` sous la forme d'un tableau associatif dont les indices sont le noms des colonnes de la requête.
  ```
  while($ligne = $reponse->fetch())
      {
      echo $ligne['col1'];
      }
  ```
- On termine le traitement de la requête grâce à l'instruction
  `<?php $reponse->closeCursor(); ?>`.

Une fois terminées toutes les requêtes, la connexion à la bdd sera fermée en affectant NULL à la variable `$bdd`. Si cela n'est pas fait explicitement, PHP fermera automatiquement la connexion lorsque le script arrivera à la fin.