TD N°I. Création de pages

1 Le langage HTML (et XHTML)

1.1 Principes

Une page Web standard est un simple fichier texte suffixé par htm ou html. Les informations de structure sont fournies à l'aide de balises qui seront interprétées par le navigateur pour donner au texte l'aspect prévu. Les principales (XHTML) sont :

```
<html>...</html>                Début et fin d'un fichier Html
<head>...</head>                Zone d'en-tête d'un fichier Html
<title>...</title>              Titre affiché par le navigateur (élément de head)
<body>...</body>                Début et fin du corps d'un fichier Html
<p>...</p>                      Nouveau paragraphe
<img src="xyz.gif" />           Insertion d'une image
<a href="http://...">...</a>    Lien vers une page Web
<a href="mailto:...">...</a>    Lien vers une adresse eMail
<a href="fichier.htm">...</a>   Lien vers la page locale fichier.
<!--...-->                      Commentaire ignoré par le navigateur
<br />                          A la ligne
```

Une liste détaillée des principales balises se trouve sur le serveur du cours (http://dormeur.info.prive/coursweb ou http://dormeur.info.iut.u-bordeaux1.fr/coursweb).

REMARQUE : les éléments de cours disponibles sur le serveur font essentiellement référence au HTML4.01, en particulier pour les syntaxes des balises. Nous vous encourageons vivement à utiliser la syntaxe de XHTML1.0, une norme plus récente et plus structurée. Les principales différences sur la syntaxe sont :

- balises et attributs uniquement en minuscule,
- balises de fermeture nécessaires (… , ,
),
- valeur de tous les attributs entre guillemets (),
- les balises doivent s'imbriquer correctement.

1.2 Création d'une page simple

La création d'une page simple peut se faire facilement à l'aide d'un éditeur de texte aussi rudimentaire que le Bloc-Notes de Windows. Pour la sauvegarder, on pourra créer un nouveau répertoire destiné à contenir l'ensemble des exercices de ce cours. Un double-clic sur l'icône associée au fichier permettra de lancer le navigateur par défaut.

Exercice :

1. Créer une page comportant un titre, des sous-titres, une liste à puce, une liste numérotée, des liens et un tableau. On pourra par exemple créer une page personnelle, présentant son état civil, son CV et ses hobbies (en inventant si nécessaire) sous forme de liens renvoyant sur des pages plus détaillées.

2 Chargement de la page sur un serveur

L'approche la plus ancienne consiste à créer les pages dans un répertoire situé sous le compte de l'utilisateur et qui est publié directement par le serveur. C'est l'approche "public_html" utilisée par le serveur Apache qui la publie sous le nom ~<login>. Sympathique dans les débuts du Web, cette approche comporte un inconvénient majeur : la publication directe d'un répertoire utilisateur engendre un trafic interne à l'organisation qui peut être pénalisant en terme d'efficacité et de sécurité.

Le serveur utilisé pour les exercices se trouve à l'adresse dormeur.info.prive. Chaque étudiant possède un répertoire personnel et donc une url http://dormeur.info.prive/<login> accessible à tout le monde. La page affichée par défaut au chargement du site peut s'appeler index.htm ou default.htm. Une telle page est en principe obligatoire.

La publication de pages sur le serveur se fait par ftp sur la même machine, sur le port standard. La connexion est possible en utilisant son nom de login et son mot de passe, le répertoire personnel <login> n'étant accessible en lecture et écriture qu'à l'utilisateur <login>.

Pour publier une page sur le serveur, on peut donc utiliser un client ftp quelconque (ligne de commande, graphique comme FileZilla ou WS_FTP, ...), ou, comme nous le verrons juste après, se servir d'un gestionnaire de site plus élaboré qui accepte ce protocole (Microsoft Frontpage, Macromedia Dreamweaver, Adobe GoLive, Nvu (logiciel libre), ...).

Exercice :

2. Publier sur le serveur la page créée à l'exercice précédent à l'aide du client ftp FileZilla.

3 Création d'un site et utilisation d'un gestionnaire de site

La gestion "à la main" d'un site complexe peut se révéler assez délicate. L'utilisation d'un gestionnaire de site permettra d'en maîtriser la structure plus facilement et de réaliser une publication "intelligente", en ne publiant que les pages modifiées.

En outre, un gestionnaire sera usuellement couplé à un éditeur de pages qui permettra, dans tous les cas simples d'éviter de taper à la main les noms des diverses balises.

Parmi les logiciels de gestion de site cités plus haut, seul Nvu est installé sur les machines du département. Nvu 1.0 peut être configuré pour générer du HTML 4.01 ou XHTML 1.0, strict ou transitional. Il ne supporte pas les cadres (frame) dont l'utilisation est actuellement fortement déconseillée par les moteurs de recherche.

Exercices :

3. Reprendre l'exemple précédent et le reproduire en utilisant l'éditeur Nvu pour éditer la page.
4. Utiliser le gestionnaire de site de Nvu pour créer un site local et un site distant, et gérer la publication des fichiers sur le serveur.

4 Mise en forme des pages

Les balises présentées plus haut permettent de préciser la structure du texte. Grâce à elles, le navigateur pourra distinguer les titres, les sous-titres et autres éléments du document. Il les affichera alors selon ses paramètres par défaut.

Pour maîtriser la présentation, le créateur de la page peut être tenté de modifier certaines de ses caractéristiques : couleur de fond, taille des caractères, police, ... Le faire directement en utilisant des balises de formatage (comme FONT, B, BIG, ...) qui ont été rajoutées à la norme initiale, est un contre-sens absolu. Comme avec un traitement de texte, la police, la graisse, la couleur, ... font partie de la nature d'un paragraphe et ne doivent pas être modifiées directement. Tout comme Word et tous les traitements de texte évolués, HTML 4.01 ou XHTML 1.0 possèdent une notion de style et de feuille de style.

La définition d'un élément de style se fait sous la forme :

```
balise {propriété de style: Valeur; propriété de style: Valeur ...}
```

comme par exemple :

```
A {font-family:Verdana; font-size:18px; font-weight:bold; font-color:yellow}
```

L'information de style peut-être contenue dans le document :

```
<style type="text/css">
      <!--
      balise1 {propriété de style: Valeur; propriété de style: Valeur ...}
      balise2 {propriété de style: Valeur; propriété de style: Valeur ...}
      -->
</style>
```

ou décrite par une feuille de style que l'on pourra appeler dans l'en-tête du fichier par une ligne du type :

```
<head>
…
<link rel="stylesheet" type="text/css" href="monstyle.css">
…
</head>
```

La première approche garantit l'uniformité de style à l'intérieur d'une page Web, la seconde, de loin préférable, permet d'uniformiser l'apparence d'un site entier.

On peut affecter des styles différents à des balises grâce au concept de classe (ce qui permet de distinguer plusieurs types de paragraphes, de titres, ... La définition des classes est aussi simple que celles des styles :

```
.maclasse {propriété de style: Valeur; propriété de style: Valeur ...}
```

On pourra ainsi tester l'effet de la notion de classe sur l'exemple suivant :

```
<style type="text/css">
      <!--
.mclass {margin:0cm; margin-bottom:.0001pt; font-size:12.0pt}
      -->
</style>
<h1>Bienvenue dans mon site Web</h1>
<h1 class=mclass>Bienvenue dans mon site Web</h1>
<p class=mclass>Bienvenue dans mon site Web</p>
<p>Bienvenue dans mon site Web</p>
```

Exercice :

5. Reprendre le premier document, en imposant des tailles et des couleurs particulières aux titres, en mettant les listes à puces en italique et en changeant la couleur du fond. On pourra aussi modifier la présentation des liens hypertextes.

On trouvera sur le serveur du cours (qui utilise bien évidemment les feuilles de styles) de plus amples détails.

TD N°II. Feuille de style : positionnement du contenu

CSS2 prévoit des feuilles de style liées à un média spécifique ce qui autorise les auteurs à présenter des documents sur mesure pour les navigateurs visuels, les synthétiseurs de parole, les imprimantes, les lecteurs en Braille, les appareils portatifs, etc. Cette spécification introduit aussi les notions de positionnement du contenu, de téléchargement des polices, de mise en forme des tables, de fonctions d'internationalisation, de compteurs et de numérotage automatiques et quelques propriétés concernant l'interface utilisateur.

1 Balise de bloc et balise en ligne

Il existe deux groupes de balises HTML :

- Les balises de type bloc comme les balises <p>, <ul>, <li>, , <form>, <input>, <blockquote>, <h1>,< h2>, …,<h6>, …

Une balise bloc peut contenir une ou plusieurs balises « bloc » et/ou « en ligne » et peut avoir une dimension (largeur, hauteur définies). Les blocs se placent par défaut l'un en dessous de l'autre (exemple : une suite de paragraphe), à droite ou à gauche dans son conteneur (le bloc qui le contient) (*propriété float*), en position absolue ou relative – (*propriété position*).

La balise est une balise neutre permettant de définir un conteneur rectangulaire que l'on pourra configurer à souhait.

- Les balises de type « en ligne » comme <a>, , , , <img>, …

Une balise « en ligne » ne peut contenir qu'une ou plusieurs balises « en ligne ». Elle n'a pas de dimension, elle occupera la place nécessaire à son contenu.

La balise est utilisée pour modifier l'aspect d'une zone limitée de données, en général une portion de paragraphe.

Exercices :

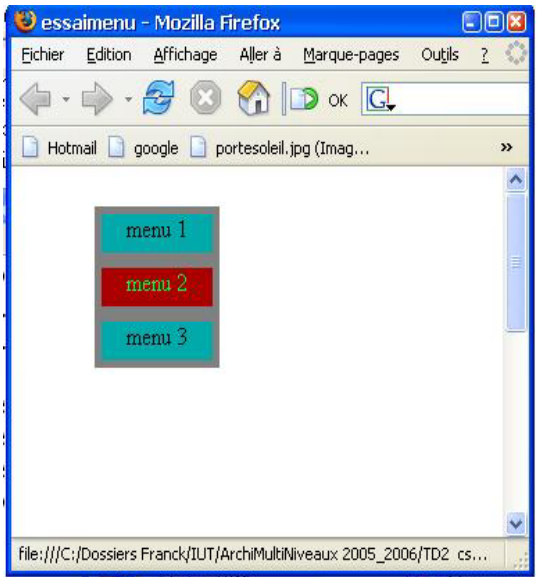
6. Placer un bloc jaune de taille 100x150 pixels à distance 15 de la gauche et 100 du haut de son conteneur.
7. Dans un paragraphe, changer la taille ou la couleur de quelques mots.
8. Réaliser une présentation à l'aide de deux colonnes, chaque colonne comportant un texte et une image à sa droite (en utilisant la propriété *float*). Que se passe-t-il si l'on redimensionne la fenêtre ? Modifier afin d'éviter le repositionnement lors d'une diminution de taille.
9. En utilisant la propriété *position* réaliser une superposition de texte.
10. En utilisant encore la propriété *position*, placer un bloc qui reste fixe (non lié au défilement de la page). Remarque importante : cette propriété ne fonctionne pas avec Internet Explorer…

2 Création de menu

La possibilité de positionner des éléments les uns sous les autres ou les uns à coté des autres permet la réalisation de menu simple (sans menu déroulant).

Exercice :

11. Réaliser un menu vertical en utilisant la balise <ul>. Chaque élément de la liste utilisera une balise <a> qui permet une réaction au passage de la souris (a:hover). Le choix de couleur permettra de faire apparaître des éléments rectangulaires. Par exemple on pourra obtenir un menu de la forme :



12. Réaliser le même menu mais horizontal.

3 Réalisation d'un « design type »

A l'heure actuelle, beaucoup de sites web sont construits grâce à l'utilisation de tableaux. Les tableaux, permettent en effet de structurer la page en plusieurs parties, et chaque partie peut contenir un menu, un entête, un contenu,... Cependant, cette méthode quasi-universelle présente de nombreux désavantages :

- l'imbrication multiple de tableaux est souvent nécessaire, même pour des design simples.
- le nombre de balises (table, tr, td, colspan, rowspan...) devient vite considérable et alourdit le code, la lisibilité et donc la mise à jour.
- ce code très lourd augmente souvent inutilement le poids de la page et du chargement.

D'autres sites sont construits à l'aide des frames (le site du cours par exemple). C'est une solution simple mais qui pose quelques problèmes (outil ne les gérant pas : Nvu par exemple, référencement plus difficile, problème d'impression des pages, …).

Une troisième solution consiste à créer un site en utilisant CSS sans tableau ni frame, uniquement en utilisant le positionnement des blocs.

Exercices :

13. Créer une page d'accueil d'un site comportant une zone entête (image, logo, texte), un menu horizontal, un menu à gauche, une zone d'affichage au centre et un pied de page.

Par exemple on pourra obtenir un menu de la forme :

TD N°III. Pages dynamiques côté client

1 Langages de script

Plusieurs approches ont été proposées pour la création de pages capables d'interagir avec l'utilisateur sans nécessiter un aller-retour avec le serveur. Les seules qui soient acceptées de façon à peu près universelle reposent sur l'utilisation des langages Java ou Javascript. Le premier étant traité dans un autre cours, nous ne nous intéresserons qu'au second. Un script commence par : <script type ="text/javaScript"> et se termine par </script>

Le bref exemple qui suit montre le code d'une fonction qui affiche une chaîne de caractères reçus en paramètre.

```
<script type="text/javaScript">
<!--function affiche(texte)
{
    alert(texte)
}
--></script>
```

L'exécution de code dans une page peut être réalisé en l'insérant directement à l'endroit voulu :

```
<script type ="text/javaScript">
<!--
    document.write('Bonjour');
    for(i=1;i<10;i++) document.write(i);
--></script>
```

Ou en provoquant l'appel à une fonction lorsque surviennent certains événements.

Exercices :

14. Informer l'utilisateur des caractéristiques de son écran, de son navigateur ...
15. Afficher l'heure et la date

Aux divers constituants de la page sont associés les événements auxquels ils peuvent répondre

onload = *script*	A la fin du chargement. Peut être utilisé dans des éléments BODY et FRAMESET.
onunload = *script*	au déchargement. Peut être utilisé dans des éléments BODY et FRAMESET.
onclick = *script*	lorsque le bouton de la souris est pressé puis relâché alors que ce dernier est situé sur l'objet
ondblclick = *script*	lorsque d'un double click. Cet attribut est reconnu par la plupart des éléments.
onmousedown = *script*	bouton enfoncé mais pas immédiatement relâché. Reconnu par la plupart des éléments.
onmouseup = *script*	Relâchement du bouton de la souris
onmouseover = *script*	La souris arrive sur l'élément. Cet attribut est reconnu par la plupart des éléments.
onmousemove = *script*	Mouvement de la souris. Cet attribut est reconnu par la plupart des éléments.
onmouseout = *script*	La souris quitte l'élément. Cet attribut est reconnu par la plupart des éléments.
onfocus = *script*	L'élément reçoit le "focus". Reconnu par LABEL, INPUT, SELECT, TEXTAREA, BUTTON.
onblur = *script*	L'élément perd le "focus". Mêmes éléments que l'événement onfocus.
onkeypress = *script*	Une touche du clavier est enfoncée puis relâchée immédiatement alors que l'élément a le "focus". Cet attribut est reconnu par la plupart des éléments.
onkeydown = *script*	Une touche du clavier est enfoncée alors que l'élément a le "focus".
onkeyup = *script*	Une touche du clavier est relâchée alors que l'élément a le "focus".
onsubmit = *script*	intervient lorsque le formulaire est soumis. Seulement dans le contexte d'un élément FORM.
onreset = *script*	lorsqu'un formulaire est réinitialisé. Seulement dans le contexte d'un élément FORM.
onselect = *script*	lorsque l'utilisateur sélectionne du texte dans un champ de texte. Cet attribut est exploitable pour des éléments INPUT et TEXTAREA.
onchange = *script*	L'élément perd le "focus" *ET* sa valeur a été modifiée depuis le dernier instant où il a été activé.

L'exemple suivant montre comment réaliser un aiguillage vers un nouveau lien choisi dans une liste déroulante :

```
Choisissez le lien <br/>
<select onChange="aiguille(this);">
     <option selected="selected">Choisissez</option>
     <option>Premier lien</option>
     <option>Deuxième</option>
</select>
```

Le code de la fonction aiguille est :

```
<script type ="text/javaScript">
```

```
<!--
  urls = new Array('http://www','http://www');
  function aiguille(liste) {
      url = urls[liste.selectedIndex-1];
      window.open(url,"decouvrir","toolbar=yes,status=yes,menubar=yes,scrollbars=yes,
resizable=yes,width=400,height=300");
    }
--></script>
```

Exercices :

16. Ouvrir une fenêtre d'information, sans barre de navigation au lancement d'une nouvelle page Web.
17. Insérer un champ de saisie et vérifier que la valeur qu'il contient est un nombre entier.
18. Réaliser une calculette à l'aide de champs de saisie et de boutons.

2 DHTML

Le document de travail est muni d'une structure arborescente (le modèle de document ou Document Objet Model) qui permet de retrouver l'ensemble de ses composants et dont d'utiliser leurs attributs pour réaliser diverses actions.

Les balises créent dans le document de nouveaux objets que la propriété id permet de nommer. Le code suivant permet de créer un objet constitué d'une liste de deux éléments.

```
<table id="indecis"><tr> <td>  Elément 1</td></tr><tr> <td>  Elément 2</td></tr> </table>
      avec pour style "border: 0; position: absolute; left: 120px; top: 30px; width: 100px;
height: 100px; z-index: 1; visibility: hidden"
```

Possédant un nom pour désigner l'objet, on sera alors en mesure de le modifier, de le faire apparaître ou disparaître, de le déplacer,... Il suffira de le rechercher en utilisant la fonction getElementById avec du code de la forme :

```
Var node = document.getElementById("indecis");
node.style.visibility = 'visible''; //pour le rendre visible
```

Exercice :

19. Ecrire un script qui permet à l'utilisateur de modifier la couleur de fond d'une balise <h1> (*attention background-color du css devient backgroundColor dans le code javascript*).
20. Réaliser un menu déroulant.
21. Permettre la sélection et le déplacement d'objets à la souris.

TD N°IV. Pages dynamiques côté serveur

1 Principes

La première technique utilisée pour créer des pages dynamiquement consistait à écrire des programmes externes. Lorsque le serveur reçoit une demande, il lance le programme avec les paramètres voulus et celui-ci lui retourne la page générée. Il peut alors la renvoyer au demandeur. Une programmation complète est nécessaire et la création d'un processus à chaque demande peut être une charge assez lourde (malgré les multiples techniques d'optimisation que l'on peut imaginer).

Une amélioration a consisté à permettre au serveur d'exécuter des programmes dans son propre contexte d'exécution (.dll de Windows ou modules chargeables d'autres environnements). Cette approche allie avantages (en termes de performances) et inconvénients (en terme de fiabilité, un module mal écrit pouvant mettre en cause le fonctionnement global du serveur).

Une approche consiste à proposer au créateur de page de réduire sa charge de travail en lui permettant d'insérer directement dans la page html des séquences d'instructions en langage interprété. Plusieurs solutions principales ont été proposées :

- PHP : (à l'origine signifiant Personal Home Page Tools, officiellement aujourd'hui, ce sigle est un acronyme récursif pour *PHP: Hypertext Preprocessor*). C'est un langage de scripts généraliste et Open Source spécialement conçu pour le développement d'applications web ; il est inspiré du langage C, les scripts sont insérés au sein du code html. Evolutions de PHP : php 1 – 1994, php 2 – 1995, php 3 – 1998, php 4- 1999, php 5 - 2004.
- ASP (Technologie de pages web dynamiques de Microsoft) : un environnement de développement basé sur un ensemble d'objets standard que l'on peut programmer à l'aide de langages comme VBScript, JavaScript ou Perl. Cet environnement est aujourd'hui remplacé par la plateforme .NET.
- ASP.NET : Constituant l'évolution logique de ASP , cette solution repose sur le « Framework » .NET. Celui-ci offre un modèle de programmation unifié permettant de développer aussi bien des applications clients Web-HTML, Windows, PocketPC que des composants serveurs (SGBD, Services Web, XML, Entreprises Services, Transactions…). La plate-forme Microsoft .NET offre actuellement une prise en charge intégrée pour trois langages : C#, Visual Basic et JScript.
- SERVLET : Programme Java qui s'exécute dynamiquement sur le serveur Web. Les JSP (*Java Server Pages*) permettent d'écrire des « servlets », en incluant dans des balises spécifiques le code java au sein du fichier HTML. Enfin, lié à la notion de portail, est apparu le terme portlet que l'on peut définir comme un servlet fonctionnant à l'intérieur d'un portail.
- Nous nous limiterons à l'utilisation de php et d'asp pour le moment.
- Le serveur reconnaît qu'une page contient du code à interpréter grâce au suffixe de la page demandée (php ou asp). Il la transmet alors à l'interpréteur qui utilise le fait qu'une balise spéciale (<?php pour php ou <% pour asp en général) indique le début d'une séquence de code, tandis que la balise inverse en marque(?> pour php ou %> pour asp) la fin.
- Exemple de code php :

```
…
<body>
<p>
<?php
echo "Hello world";
?>
</p>
</body>
</html>
```

- Exemple de code asp :

```
…
<body>
<p>
<% FOR i = 1 to 10 %>
     Bienvenue
<% Next %>
</p>
</body>
</html>
```

Exercices (les deux exercices seront réalisés dans les deux environnements (ASP+VBSCRIPT et php).

22. Ecrire une page affichant une liste (<ul>) des entiers de 1 à 10.
23. Ecrire une page affichant les valeurs de divers paramètres CGI standard (Url, nom du serveur, nom et ip du client). On pourra utiliser les fonctions php *getenv* et/ou *phpinfo*, la méthode *servervariables* disponible pour l'objet *request* d'asp.

2 Traitement de formulaires

2.1 Construction d'un formulaire

La construction d'un formulaire se fait très facilement, en mode texte ou en mode graphique. Un bouton « envoyer » (de type submit) transmettra alors les données du formulaire à la page spécifiée dans le paramètre « action », de la façon spécifiée dans le paramètre « method » :

```
<form method="post" action="exemple2.php">
  <p> Nom :   <input type="text" size="20" name="nom"/></p>
  <p>Pr&eacute;nom : <input type="text" size="20" name="prenom"/></p>
  <p><input type="submit" value="Envoyer"/>
       <input type="reset" value="Clear"/></p>
</form>
```

Les deux méthodes, POST et GET différent en ce que GET transmet les paramètres dans l'url elle-même en les concaténant aux noms de fichiers, tandis que POST les transmet de façon transparente (et n'a donc pas les limitations de longueur de la méthode GET).

Dans les deux cas, la communication se fait en respectant le protocole standard CGI (Common Gateway Interface). La deuxième forme est particulièrement intéressante en ce qu'elle permet de simuler un formulaire en construisant directement la chaîne d'appel, ie en appelant une URL de la forme :

```
http://serveur/page.php?nom=dupont&prenom=pierre
```

Nous verrons plus loin que c'est elle qui permet de réaliser facilement des liens paramétrés lorsque nous construirons des pages à l'aide d'accès à des bases de données.

2.2 Traitement des données

Les données transmises à la page de code par l'une ou l'autre des méthodes POST et GET sont facilement récupérées par le serveur. Les modèles actuels (asp aussi bien que php) décodent automatiquement les paramètres reçus et les fournissent à la page sous le nom qui leur était attribué dans le formulaire ou dans l'url.

2.2.1 ASP

Le modèle asp distingue selon que la méthode d'appel est POST :

```
<p>Vous êtes <%= Request.Form("nom")%> <%= Request.Form("prenom")%> </p>
```

Ou GET, auquel cas l'accès se fait par Request.Querystring("nom") au lieu de Request.Form("nom").

2.2.2 PHP

Le code php est un peu plus simple et peut ne pas distinguer la méthode d'appel :

```
<p>Vous &ecirc;tes <?php echo $_REQUEST["nom"]." ".$_REQUEST["prenom"]; ?></p>
```

24. Réaliser en php un formulaire comportant les éléments standard (champ de saisie, liste déroulante, case à cocher), puis la page permettant de récupérer les informations transmises (par exemple en les affichant)

3 Accès aux bases de données en PHP

L'accès à une base de données nécessite tout d'abord la création d'une connexion, qui peut être native ou se faire par le biais d'un lien ODBC (Open Data Base Connectivity). Il n'y a qu'un seul langage SQL, mais chaque éditeur de SGBD implémente son propre *dialecte*. Le "dictionnaire" qui permet de passer d'un dialecte à l'autre s'appelle ODBC. Il a été imaginé par Microsoft (1993). Depuis une plate-forme Win32 , des drivers ODBC existent pour pratiquement toutes les bases de données.

```
…
<body>
<?php// Connexion odbc
    $host = "Bibli";
    $user = "etd";
    $password = "etd";
    $conn = odbc_connect($host,$user,$password) or die ("raté");
    // Accès à la table
    $query = "Select Nom from auteur order by nom";
    $result = odbc_exec($conn,$query);
    // boucle de lecture
    while (odbc_fetch_row($result)) {
        $nom = odbc_result($result,1);
        echo "<p> Nom : ";
        echo htmlentities($nom); // htmlentities pour convertir les caractères éligibles en
                                 // entités HTML
       echo "</p>";
    }
    // Deconnexion de la base de donnees
    odbc_close($conn);
?>
</body>
</html>
```

Les exercices demandés font appel à la base bibli à laquelle on peut accéder par la connexion etd/etd. Les tables sont en lecture seule (select) à l'exception des tables prêter et lecteur qui sont accessible en modification (insert et update).

25. Page affichant la liste des auteurs dont le nom commence par la lettre A.
26. Page affichant la liste des auteurs dont l'initiale a été fournie par un formulaire.
27. Même page, en ajoutant pour chaque auteur un lien sur la liste de ses oeuvres (calculée par une deuxième page).
28. Insertion d'un nouveau lecteur. Gestion des erreurs potentielles (homonymes par exemple), mise à jour des données
29. Insertion d'un prêt

TD N°V. Suivi des utilisateurs (cookies et sessions)

Le protocole HTTP étant un protocole sans état, le serveur ne garde pas trace de l'identité de ses visiteurs. Il n'est donc pas capable lorsqu'il reçoit une nouvelle requête, de savoir si elle provient de quelqu'un qui s'est déjà connecté.

1 Les cookies

Les cookies ont été conçus par la société Netscape afin d'étendre les fonctionnalités du protocole http et de lui ajouter la possibilité de faire un lien entre les différentes requêtes. Ils ont été ensuite intégrés au protocole ; tous les navigateurs actuels les prennent en charge. La solution consiste à stocker chez le client une information qu'on pourra lui demander de renvoyer lorsqu'il fera une nouvelle demande de page. C'est ces informations qui sont appelés cookies. Ils peuvent être de deux types :

- Cookies sans durée d'expiration : ils disparaissent lorsque le navigateur est fermé. Ils permettent essentiellement d'assurer le suivi de la connexion de l'utilisateur puisqu'ils disparaissent à la fin de la session ;
- Cookies permanents, stockés sur le disque de l'utilisateur et qui peuvent être réutilisés lors d'une nouvelle connexion sur le site. Ce sont de simples fichiers « texte » que vous pouvez lire. De tels cookies permettent d'identifier chaque utilisateur de manière unique.

Le principe de fonctionnement est simple ; lorsque le serveur envoie un cookie, il demande au navigateur de le renvoyer dans ses prochaines requêtes. Il faut remarquer que les cookies sont gérés dans les en-têtes envoyés avant la page web donc avant tout contenu html.

Toute la gestion des cookies en php se fait avec une unique fonction : *setcookie()* et à l'aide de la variable *$_COOKIE[]* .

Exemple d'envoi d'un cookie :

```
<?php
	setcookie('nom','valeur du cookie') ;
?>
<html ...>
...
<body>
	<p> Un cookie a &eacute;t&eacute; envoy&eacute;</p>
	<p> Son nom est : "nom" </p>
	<p> Son contenu est  "valeur du cookie"</p>
<p><a href="lectureCookie.php" >Suite </a></p>
</body>
</html>
```

Exemple de lecture d'un cookie :

```
<html ...>
...
<body>
<?php
if (isset($_COOKIE["nom"])) {
	echo "<p> Un cookie a &eacute;t&eacute; envoy&eacute;</p>" ;
	echo "<p> Son nom est : nom </p>" ;
	$val=$_COOKIE["nom"];
	echo "<p> Son contenu est:\" $val \" </p>" ;
}
?>
</body>
</html>
```

Pour effacer un cookie, il suffit d'envoyer un cookie de même nom sans contenu. Pour rendre un cookie permanent, il suffit de définir une date d'expiration (voir documentation de *setcookie*).

Exercice :

30. En utilisant l'exemple ci-dessus, rendre le cookie permanent en plaçant comme contenu le nombre de visite (incrémenté à chaque visite).

Quelques limitations des cookies :

- Limités en nombre et en taille (20 par domaine, 4ko par cookie) ;
- Aucune sécurité : l'utilisateur peut le créer, le modifier, …
- De nombreuses personnes désactivent dans leur navigateur l'utilisation de cookies, empêchant ainsi d'utiliser cette technique.

2 Les sessions

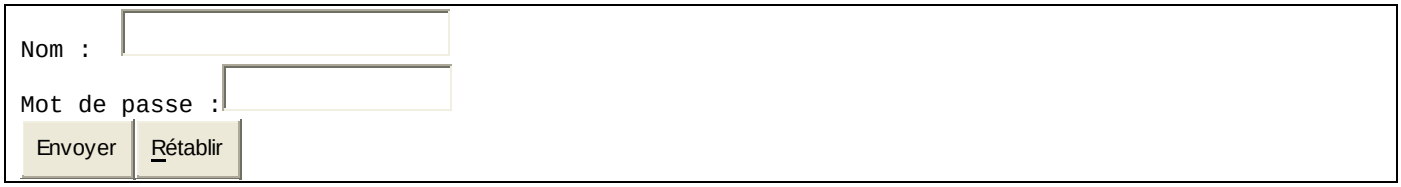
Les cookies ne sont donc pas une solution pour gérer des données sensibles comme des données d'authentification. Durant une session, pour que seul un utilisateur authentifié puisse accéder aux pages protégées, nous allons étudier deux techniques

nécessitant la saisie par l'utilisateur d'un login et d'un mot de passe. Bien évidemment, nous devons disposer de la liste des usagers autorisés (logins et mots de passe) par exemple dans une table d'une base de données.

2.1 Suivi manuel de session

Dans cette page, nous proposons une solution simple de suivi de connexion sans utiliser les méthodes proposées par l'un ou l'autre des langages de script. Le code proposé est en php, mais pourrait sans peine être écrit dans le modèle asp.

La connexion se fera naturellement à travers un formulaire du type



qui appelle une page de code qui va successivement :

- valider le nom et le mot de passe (et rejeter l'utilisateur si l'information est incorrecte)
- créer une session et rediriger l'utilisateur vers la page voulue.

Le site du cours présente le code php correspondant.

Exercice:

31. Page affichant la liste des auteurs dont l'initiale a été fournie par un formulaire avec suivi manuel de session.

2.2 Suivi de session en php (>=4)

Pour répondre aux limitations des cookies, php met à disposition un concept plus évolué : les sessions. Au lieu de stocker les informations chez le visiteur, elles sont stockées sur le serveur. Techniquement, un identifiant est attribué au visiteur. A chaque fois qu'il revient en annonçant cet identifiant, php peut récupérer les informations sauvegardées pour ce visiteur.

Pratiquement, une session s'initialise avec session_start(). Php essaie alors de lire l'identifiant fourni par l'utilisateur, va chercher le fichier correspondant et met à disposition les informations sauvegardées dans la variable $_SESSION[]. Si aucun identifiant de session n'est reçu, php en crée un unique aléatoirement, l'envoie au visiteur et crée le fichier de données correspondant.

Pour lire, modifier, supprimer ou créer des informations, il suffit de lire, modifier ou créer des entrées dans $_SESSION[].

Exemple de code pour vérifier si une session est active :

```
<?php
session_start();
if (!isset($_SESSION["Valide"]))header("Location:
/rubi/TD5/SolutionSessionPHP/Connexion.php");
?>
```

Exemple de code pour initier une session:

```
<?php
ouvreBD($conn);
$Nom=$_REQUEST["Nom"];
$Password=$_REQUEST["Password"];
// recherche si l'utilisateur est enregistré et possède le bon mot de passe
$query = "Select * from Lecteur where Nom ='".$Nom."' and Password ='".$Password."'";
$succ = odbc_exec($conn,$query);
if (odbc_fetch_row($succ))
{
  $Code = odbc_result($succ,1);
  $Loc = "Location: /rubi/TD5/SolutionSessionPHP/SelectAuteur.php";
  session_start();
  $_SESSION["Valide"] = "true";
  $_SESSION["NOM_USER"]= "$Nom";
  fermeBD($conn);
  header($Loc);
}
else
{
  // Rejet d'un utilisateur incorrect
  fermeBD($conn);
  header("Location: /… ");
}
?>
```

session_destroy() permet de terminer une session.

32. Page affichant la liste des auteurs dont l'initiale a été fournie par un formulaire avec suivi de session php.

TD N°VI. XML

1 Présentation XML

Si le langage de présentation de page HTML permet de préciser la structure d'une page pour que le navigateur la présente de façon agréable à l'utilisateur, il ne permet pas de gérer la structure interne des données contenues dans la page (liste de personnes, de produits, ...). Elles peuvent être au mieux être organisées au sein de listes ou de tableaux, mais toute information de structure est en fait perdue.

Le langage XML (eXtensible Markup Language) a été défini dans le but de pallier cette déficience. Comme HTML, XML est issu d'un langage beaucoup plus général mais beaucoup trop complexe pour être utilisé en pratique, le langage SGML créé dans le but de faciliter la gestion électronique de documents.

Comme HTML, XML est un langage à balises, mais deux caractéristiques l'en distinguent :

- les balises ne sont pas prédéfinies, mais peuvent être créées librement par l'utilisateur,
- à chaque balise ouvrante doit correspondre une balise fermante

Les données du document sont alors organisées sous forme d'imbrication de balises.

Quelques exemples d'avantages de XML :

- Compatibilité et interopérabilité : XML est un standard W3C, il permet la communication entre applications et est aujourd'hui considéré comme le standard d'échange entre systèmes hétérogènes ;
- Pérennité : XML est compris parla majorité des applications et logiciels, les méthodes de manipulation sont standardisées ;
- Simplicité d'utilisation : le traitement XML est relativement simple et les API sont standardisées et similaires dans tous les langages ce qui facilité l'apprentissage.

XML permet de plus d'utiliser un fichier afin de vérifier qu'un document XML est conforme à une syntaxe donnée. La norme XML définit ainsi une définition de document type appelée DTD (*Document Type Definition*), c'est-à-dire une grammaire permettant de vérifier la conformité du document XML. La norme XML n'impose pas l'utilisation d'une DTD pour un document XML, mais elle impose par contre le respect exact des règles de base de la norme XML.

Ainsi on parlera de:

- **document valide** pour un document XML comportant une DTD
- **document bien formé** pour un document XML ne comportant pas de DTD mais répondant aux règles de base du XML

Une DTD peut être définie de 2 façons:

- sous **forme interne**, c'est-à-dire en incluant la grammaire au sein même du document
- sous **forme externe**, soit en appelant un fichier contenant la grammaire à partir d'un fichier local ou bien en y accédant par son URL.

Exemple de fichier XML (gens.xml) :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<GENS>
  <PERSONNE>
    <NOM> Flora Bonhomme </NOM>
    <ADRESSE>
      <RUE> 5 rue Galliéni </RUE>
      <CODE> 75012 </CODE>
      <VILLE> Paris </VILLE>
    </ADRESSE>
    <TEL> 01 23 45 67 89 </TEL>
  </PERSONNE>
  <PERSONNE>
    <NOM> Anne Gentile </NOM>
    <ADRESSE>
      <RUE> 2 rue des Dames </RUE>
      <CODE> 75007 </CODE>
      <VILLE> Paris </VILLE>
    </ADRESSE>
    <TEL> 01 45 23 67 89 </TEL>
    </PERSONNE>
    <PERSONNE>
    <NOM> Linda Legay </NOM>
    <ADRESSE>
      <RUE> 5 avenue Foch </RUE>
      <CODE> 35000 </CODE>
      <VILLE> Rennes </VILLE>
    </ADRESSE>
    <TEL> 01 23 67 45 89 </TEL>
  </PERSONNE>
</GENS>
```

Exercice :

33. En vous inspirant de l'exemple *gens.xml*, créer un document *étudiants.xml* comportant les données : noms, adresses (rue, ville), téléphone, scolarité (établissement, année d'études). Ce fichier comportera au moins 5 étudiants. Visualiser ce fichier à l'aide d'Internet explorer et de Mozilla Firefox.

2 Langage XSL

Le langage XSL permet de présenter visuellement des éléments définis dans un document XML alors que le langage XML (eXtended Markup Language) définit plutôt la sémantique (le sens) des données.

Le langage XSL se divise en trois parties principales :

- **Le formatage** : application de règles de style sur des éléments XML à l'instar du langage CSS.
- **La transformation** : substitution d'un marquage XML en un balisage HTML ou un autre marquage XML.

La partie *formatage* du langage XSL (eXtensible Stylesheet Language) a une fonction semblable à celle du langage CSS (Cascading StyleSheet). Un ensemble de propriétés de style permet de contrôler la présentation des données à l'écran. Ainsi, un document XML pourra être mis en forme pour un affichage sur un moniteur informatique, un écran de télévision ou pour l'impression ou encore pour une version auditive.

Le langage XSL par une série de règles de transformation, remplace les éléments XML et leurs attributs en balisage HTML ou en d'autres marqueurs XML. Cette section du langage XSL s'appelle XSLT soit Langage des feuilles de Style de Transformation dont les spécifications sont mises au point par le W3C (Worl Wide Web Consortium).

Par des règles de transformation, les données textuelles contenues dans les éléments XML ou dans leurs attributs sont présentées selon le résultat de la génération d'un balisage HTML. Une feuille de style XSL peut entièrement réorganiser les éléments XML en sélectionnant des éléments et leurs attributs puis en les transformant en d'autres éléments.

Le fonctionnement du langage XSL s'effectue selon des règles de style applicables à différent motif d'un document XML.

Ainsi, tous les motifs énoncés par la feuille de style et trouvés dans le document XML par un processeur XSL, sont soit formatés, soit transformés en une combinaison textuelle spécifique.

Pour plus d'informations, vous pouvez par exemple consulter le site :

http://www.laltruiste.com/document.php?page=1&rep=5

Exemple de style xsl :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html"
       indent="yes"
       encoding="ISO-8859-1"/>
<xsl:template match="/">
<html>
<title>Annuaire telephonique</title>
<link rel="stylesheet" type="text/css" href="monstyle.css" />
</head>
<xsl:apply-templates/>
</html>
</xsl:template>
<xsl:template match="GENS">
  <body>
  <h1>Annuaire téléphonique</h1>
    <table class=”table”
      <th>Nom</th>
      <th>Tel</th>
      <th>Ville</th>
      <xsl:apply-templates/>
    </table>
  </body>
</xsl:template>
<xsl:template match="PERSONNE">
  <tr>
    <td><xsl:value-of select="child::NOM"/></td>
    <td><xsl:value-of select="child::TEL"/></td>
    <td><xsl:value-of select="descendant::VILLE"/></td>
   </tr>
</xsl:template>
</xsl:stylesheet>
```

Le document xml comportant alors une référence à la feuille xsl de la forme :

```
<?xml-stylesheet href="exemple2.xsl" type="text/xsl"?>
```

Exercise:

34. En vous inspirant de la feuille de style *exemple2.xsl*, créer une feuille de style étudiants1.xsl permettant une visualisation complète des données contenues dans *étudiants.xml* sous forme d'un tableau.

XML permet d'associer des attributs à une balise permettant ainsi d'apporter des informations complémentaires. La syntaxe est : <*balise attribut1="valeur1" attribut2="valeur2" …*>. L'élément <*xsl:if*> permet d'appliquer un test conditionnel par exemple sur la valeur d'un attribut.

Exercice :

35. Ajouter un attribut à la donnée adresse indiquant s'il s'agit de l'adresse personnelle de l'étudiant ou l'adresse de ses parents. Créer une seconde feuille de style ne faisant apparaître que les étudiants de l'IUT comportant les données nom, année d'études et adresse s'il s'agit de leur adresse personnelle.

Autres exercices possibles …..A VOIR

Exercice :

Récupérer sur le site du département le document ppn.xml.

Créer une feuille de style présentant uniquement le sommaire sur les trois premiers niveaux.

Créer une seconde feuille de style comportant la liste des Unités de Formation ne nécessitant aucun pré requis.

Exercice :

A l'aide de visual studio, créer la DTD xml (.*xsd*) associé à étudiants.xml.

Exercice XML et php

Voir par exemple RSS