Licence III :

PROGRAMMATION WEB

Dr Konaté

https://www.researchgate.net/profile/Konate-Ngolo

Table des matières

CHAPITRE 1 : INTRODUCTION A LA PROGRAMMATION WEB

Définition et historique

Le World Wide Web, littéralement la « toile (d'araignée) mondiale », communément appelé le web parfois la Toile ou le WWW, est un système hypertexte public fonctionnant sur Internet qui permet de consulter, avec un navigateur, des pages accessibles sur des sites.
Le Web n'est qu'une des applications d'Internet. D'autres applications d'Internet sont le courrier électronique, la messagerie instantanée, …etc.

Historique

1989 : Tim Berners-Lee lance l'idée de
la toile
En tant qu'utilisateur de CERNET, le réseau du **CERN**, le chercheur **Tim Berners-Lee** conçoit l'idée de naviguer simplement d'un espace à un autre d'Internet à l'aide de liens hypertextes et grâce à un navigateur. **Tim Berners-Lee** parle de la création d'une toile, tout internaute pouvant aller d'un contenu à l'autre suivant des voies multiples. Il présentera son projet au **CERN** en Novembre 1990. Pendant les trois années suivantes, il travaillera à l'apparition du World Wide **Web**, « toile d'araignée mondiale ».

1990 : L'Université de l'Illinois présente
Mosaic
L'université de l'Illinois présente son navigateur **Web** graphique, reposant sur les principes de la Toile tels qu'ils ont été formulés par l'équipe du **CERN** de **Tim Berners-Lee**, notamment le HTTP. Nommée **Mosaic**, l'application fonctionnant sur **Windows** simplifie considérablement la navigation. Elle annonce le développement ultérieur de **Netscape** et autres navigateurs.

1994 : Naissance du W3C
Tim Berners-Lee fonde le World Wide Web Consortium, également appelé W3C. Cet organisme a pour objectif et fonction d'émettre des recommandations afin de promouvoir et d'assurer la compatibilité des technologies utilisées sur le Web. Toutefois les standards proposés ne sont pas des normes absolues. L'organisme, essentiel pour assurer l'efficacité des applications tels que les navigateurs, est géré

conjointement par des universités et centres de recherche américains, européens et japonais.

LES CONCEPTS DE BASE

Le **World Wide Web**, littéralement la « toile (d'araignée) mondiale », communément appelé le Web, le web parfois la Toile ou le WWW, est un système hypertexte public fonctionnant sur Internet qui permet de consulter, avec un navigateur, des pages accessibles sur des sites.

Une ressource du web est une entité informatique (texte, image, forum Usenet, boîte aux lettres électronique, etc.) accessible indépendamment d'autres ressources. Une ressource en accès public est librement accessible depuis Internet. Une ressource locale est présente sur l'ordinateur utilisé, par opposition à une ressource distante (ou en ligne), accessible à travers un réseau.

On ne peut accéder à une ressource distante qu'en respectant un protocole de communication. Les fonctionnalités de chaque protocole varient : réception, envoi, voire échange continu d'informations.

HTTP (pour HyperText Transfer Protocol) est le protocole de communication communément utilisé pour transférer les ressources du Web. HTTPS est la variante sécurisée de ce protocole.

Une **URL** (pour Uniform Resource Locator) pointe sur une ressource. C'est une chaîne de caractères permettant d'indiquer un protocole de communication et un emplacement pour toute ressource du Web. https://abidjan.net/

Un hyperlien (ou lien) est un élément dans une ressource associé à une URL. Les hyperliens du Web sont orientés : ils permettent d'aller d'une source à une destination.

HTML (pour HyperText Markup Language) et XHTML (Extensible HyperText Markup Language) sont les langages informatiques permettant de décrire le contenu d'un document (titres, paragraphes, disposition des images, etc.) et d'y inclure des hyperliens. Un document HTML est un document décrit avec le langage HTML. Les documents HTML sont les ressources les plus consultées du Web.

Dans un mode de communication client-serveur, un serveur est un hôte sur lequel fonctionne un logiciel serveur auquel peuvent se connecter des logiciels clients fonctionnant sur des hôtes clients.

Un serveur Web est un hôte sur lequel fonctionne un serveur HTTP (ou serveur Web). Un serveur Web héberge les ressources qu'il dessert.

Un navigateur Web est un logiciel client HTTP conçu pour accéder aux ressources du Web. Sa fonction de base est de permettre la consultation des documents HTML disponibles sur les serveurs HTTP. Le support d'autres types de ressource et d'autres protocoles de communication dépend du navigateur considéré.

Une page Web (ou page) est un document destiné à être consulté avec un navigateur Web. Une page Web est toujours constituée d'une ressource centrale (généralement un document HTML) et d'éventuelles ressources liées automatiquement accédées (typiquement des images).

Un éditeur HTML (ou éditeur Web) est un logiciel conçu pour faciliter l'écriture de documents HTML et de pages Web en général.

Un site Web (ou site) est un ensemble de pages Web et d'éventuelles autres ressources, liées dans une structure cohérente, publiées par un propriétaire (une entreprise, une administration, une association, un particulier, etc.) et hébergées sur un ou plusieurs serveurs Web.

Une adresse Web est une URL de page Web, généralement écrite sous une forme simplifiée limitée à un nom d'hôte. Une adresse de site Web est en fait l'adresse d'une page du site prévue pour accueillir les visiteurs.

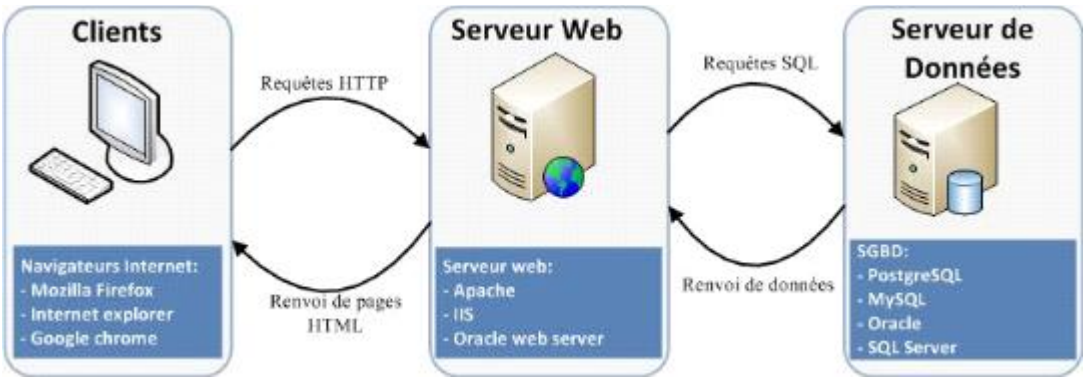
Un hébergeur Web est une entreprise de services informatiques hébergeant (mettant en ligne) sur ses serveurs Web les ressources constituant les sites Web de ses clients.

Un annuaire Web est un site Web répertoriant des sites Web.

Un portail Web est un site Web tentant de regrouper la plus large palette d'informations et de services possibles dans un site Web. Certains portails sont thématiques.

Un service Web est une technologie client-serveur basée sur les protocoles du Web.

ARCHITECTURE DU WEB



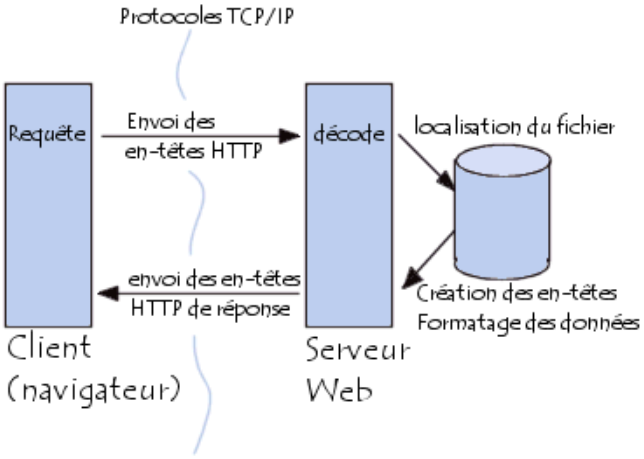
L'architecture du web se base sur les modèles de Serveur/Client. Le client envoie des requêtes au serveur, comme :

- transfert de fichiers
- exécution de programmes sur le serveur
- mise à jour de fichiers

Les objets manipulés sont repérés par leur URL.

Le transfert se fait en utilisant le protocole http. Il définit le langage utilisé pour les échanges entre client et serveur Web. Ce protocole n'exige pas de session permanente entre client/serveur

DEROULEMENT D'UNE REQUETE



1. Demande d'une connexion
2. Attente de la réponse du serveur
3. Etablissement de la connexion
4. Envoi d'une requête URL
5. Réponse du serveur
6. Affichage de la réponse
7. Fermeture de la connexion

LE PROTOCOLE HTTP

Le HTTP est un protocole simple, basé sur l'émission d'une requête vers un **serveur HTTP**, en vue de l'obtention d'une **ressource**. Cette ressource est identifiée par un URL = une position "logique" dans le réseau. L'objectif initial du HTTP est la requête sur des pages hypertexte (HTML), lesquelles servent de base à la construction d'un document composite, par combinaison de divers types de ressources annexes :

- Des images
- Des feuilles de styles
- Des scripts clients attaches

L'un des plus importants avantages du protocole http est de « découper » les documents à transférer en blocs. Ces blocs peuvent être acheminés de manière indépendante du serveur vers le client. . Ils seront ensuite assemblés au niveau du récepteur.

Gestion des cas d'erreur

Comme tout protocole se doit d'être complet, HTTP doit permettre de renseigner le client sur les multiples cas d'erreur qui peuvent se rencontrer au moment de la résolution de l'adresse "logique". Les codes d'erreur du HTTP sont le résultat du catalogage des erreurs possibles (ressource manquante, URL non conforme, domaine inexistant, accès interdit, etc...)

GESTION DE L'ACHEMINEMENT

Le HTTP prend en charge, pour la gestion de l'acheminement des données, certains aspects techniques :

- Gestion des caches : Le transport de documents en HTTP s'effectue à travers un réseau informatique qui peut compter un certain nombre de relais. Ces relais sont des organes physiques (hors câbles) qui peuvent faire preuve d'une certaine intelligence vis à vis du contenu (filtres, proxies, routeurs). Le protocole HTTP prend en charge un certain nombre d'aspects de cette chaîne d'acheminement, en donnant des indications sur les réactions souhaitées de cette chaîne.
- Gestion des redirections : Les ressources sont des document "propriété" de l'émetteur. Le problème du Web aujourd'hui est autant dans les technologies clientes (accroissement des possibilités) que dans celle de l'organisation de la masse de documents qu'il représente. Les ressources sont donc souvent réorganisées, mobiles et parfois fugitives. C'est de plus en plus le cas avec les applications dynamiques. Le protocole HTTP contient des primitives qui permettent de donner des informations pertinentes sur ces déplacements.

GESTION D'UNE PERSISTANCE D'INFORMATION

Le protocole HTTP, à travers les Cookies, fournit un moyen pour "marquer" l'agent client de façon suffisamment précise :

- L'application est marquée, puisque c'est dans le container de cette application que l'information de marquage est stockée.
- L'utilisateur d'une machine multi-utilisateur est marqué, puisque les informations de marquage sont stockées dans le profil particulier de cet utilisateur.

STRUCTURE DU HTTP

Notion d'entité : Le HTTP est un protocole destiné à transporter des entités document. Il est constitué de messages-requêtes (du client au serveur) et de messages-réponse (du serveur au client). Il implémente donc parfaitement un paradigme client-serveur en ce sens que :

- Le client (agent utilisateur) est toujours l'initiateur de la requête (PULL).
- Le serveur est toujours en attente d'un client

Les messages transportent des entités. Le cas le plus fréquent est celui d'une entité unique. L'entité est toujours définie par :

- Une en-tête d'entité (méta informations sur le contenu et la transmission).
- Un corps d'entité (le contenu).

Le protocole admet cependant que plusieurs entités puissent être transportées par le même message. C'est le cas du téléchargement de fichiers accompagnant des données de formulaire, ou de téléchargements multiples. (format **multipart**).

Dans ce cas, le message doit permettre de séparer proprement les différentes entités qu'il transporte.

CONSTRUCTION DE LA REQUETE

Le libellé de requête :

La requête HTTP, comme son nom l'indique est une demande à un serveur pour qu'il exécute un ordre (principalement d'obtenir un document). Ces ordres sont symbolisés par des VERBES.

Les verbes les plus connus sont :

- HEAD : "donne moi les méta informations conernant un document"
- GET : "donne moi le contenu du document (et ses méta informations par la même occasion)"
- POST : "fais ce que tu veux (ou tu peux avec les données que je t'envoie)"
- PUT : "mets le document que je t'envoie où je te le dis"

La cible :

Elle est représentée par une URL absolue ou relative. Le serveur est chargé de "consommer" cette cible, la traduire en une réalité physique tangible (un fichier physique, un exécutable, un répertoire). Les champs d'en-tête

La requête peut informer le serveur d'un certain nombre de prédisposition de l'agent utilisateur. Des mécanismes de gestion de préférences, convenues entre le client et le serveur peuvent permettre d'améliorer le service rendu, sans faire appel à des choix explicites de l'utilisateur. Ces préférences sont, par exemple :

- Le choix de la langue la plus appropriée.
- Le choix d'un encodage acceptable du côté du client.
- Le choix d'une version suffisamment récente, ou au contraire de versions plus anciennes.
- Les champs de requête sont placés à la suite de la "ligne VERBE" sous la syntaxe :

NomChamp:[ValeurChamp]LF

L'en-tête doit se terminer par une ligne vide, pour indiquer que ce qui suit est un corps d'entité (ou que le message est fini).

Un corps de requête

Certains verbes (PUT, POST) supposent très fortement une transmission de contenu du client vers le serveur. Ce contenu pouvant être à priori d'une taille quelconque, il s'agit bien d'un corps d'entité.

Dans ce cas, on doit alors considérer potentiellement une requête comme étant une transmission d'un document à part entière. Le client pousse un document local vers le serveur.

Détermination de la longueur du corps d'entité

Il n'existe pas en HTTP de bloc de fin de message identifiable. D'autre part, virtuellement, le corps d'entité peut contenir n'importe quelle séquence binaire, y compris une séquence qui pourrait se confondre avec un tel bloc. La question de prévoir la fin du corps d'entité DOIT être réglée dès l'en- tête. Toute requête portant un contenu d'une certaine longueur doit informer le serveur de cette longueur. On utilise le champ :

`Content-Length`

Résumé

Une requête mono-entité, dans le cas général se contruit ainsi :

```
[verbe] [Url] [version HTTP] LF
Content-Length:[longueur du corps]
[Nom Champ]:[Valeur Champ]LF
...
LF (sépare l'en-tête et le corps d'entité)
[Octets de l'entité]
```

Construction de la réponse

De cette dernière conclusion, on déduit que le schéma de réponse est symétrique à celui de la requête, à la différence près que le corps d'entité est le plus souvent non vide.

Les outils du web

L'IDE

En 2021, les outils de développement web ne se contentent pas d'enregistrer notre texte – ils nous aident à créer des projets et dynamisent l'ensemble du processus. En outre, il faut tenir compte de bien d'autres éléments que le HTML et le CSS.

Dans cette section, on se bornera a la notion d'éditeur en parlant d'IDE.

Le choix d'un IDE dépend de plusieurs sensibilités qui se développent au cours de la carrière d'un développeur web

Le navigateur

Tester une application web en développement sur plusieurs navigateurs répondait par le passé aux soucis de vérifier la compatibilité des balises sur les plateformes. Aujourd'hui une notion importante nous affranchit peu a peu de cette difficulté, la notion de **sites responsives.**

Le serveur

Avant tout déploiement de site, un travail en local peut se faire. Pour cela nous disposons de plusieurs serveurs locaux en fonction de notre SE

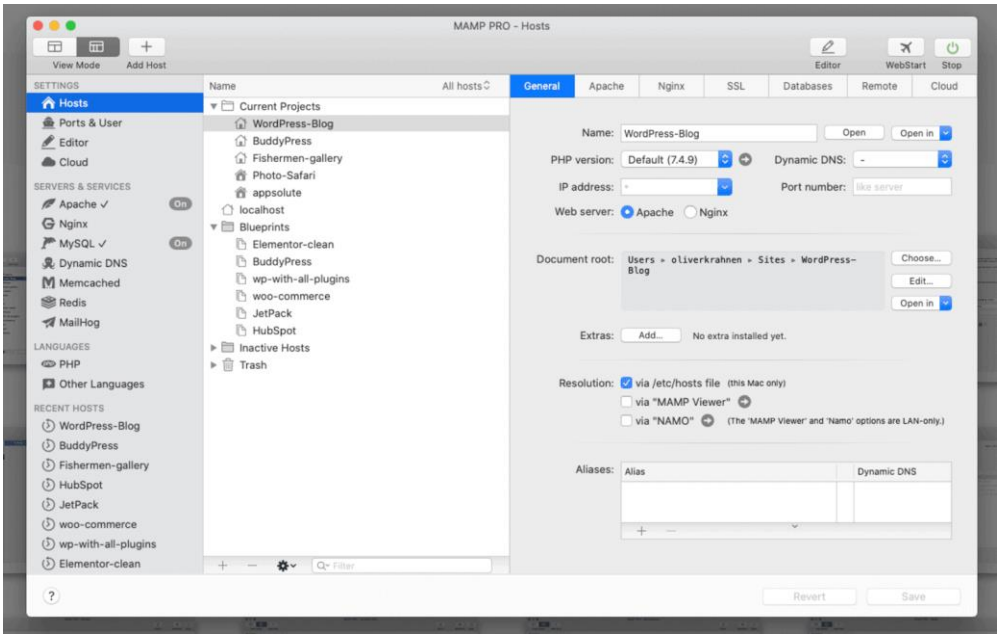
On pourrait penser que les outils classiques de création et de déploiement de pages web sont morts et enterrés, compte tenu de l'arrivée d'outils sandbox plus rapides. Pourtant, les piles de services web traditionnelles, telles que LAMP, MAMP et XAMPP, ont toujours le vent en poupe

En général, elles combinent en une seule pile un système d'exploitation (OS) – Linux, macOS ou Windows – avec un serveur web Apache, une base de données MySQL et

les langages de programmation Python, PHP et Perl. En tant que telle, une pile de services web comme celle-ci sera toujours utilisée en 2021.

MAMP

MAMP est la version de l'outil spécifique à macOS. Cette approche vous permet d'installer une pile et de travailler sur la conception et le déploiement. Bien que le processus puisse être plus fastidieux que les configurations plus modernes, il existe un niveau similaire de flexibilité sous le capot – ou du moins, le potentiel est là.



Xamp

XAMPP est une autre pile de services web qui reçoit beaucoup d'amour de la part des développeurs PHP, y compris ceux qui créent des produits WordPress. Le « X » dans le nom représente la nature multi-plateforme de l'outil. Il propose des installateurs pour les machines Windows, macOS et Linux

Alors qu'il y avait autrefois une différence entre les diverses piles de services web, les mises à jour et améliorations constantes ont égalisé le score. Néanmoins, XAMPP possède quelques atouts uniques dans sa manche.

Par exemple, MySQL n'est plus le système de gestion de base de données relationnelle (SGBDR) par défaut. À la place, XAMPP utilise MariaDB. Il s'agit probablement d'une représentation plus fidèle d'un serveur de production, étant donné le passage à d'autres solutions après l'acquisition d'Oracle.

De plus, il y a un installateur d'applications web dans le paquetage XAMPP. Bitnami est similaire à des solutions telles que Softaculous, mais Bitnami est spécifique à XAMPP :

Les langages de la programmation web

Les langages back end

Le développement backend se concentre principalement sur le côté serveur du site web. En général, le backend se compose de trois parties principales :

- Base de données
- Application
- Serveur

Du côté du serveur, le backend alimente le fonctionnement du site Web. En outre, les programmes écrits ici par les développeurs backend sont utilisés pour communiquer les informations de la base de données au navigateur.

Il existe une pléthore de langage back end. Dans le cadre de ce cours, nous utiliserons le langages PhP

Les langages Front end

Ces langages les affichages de l'application. Pendant longtemps JavaScript a été le pionnier en termes de langages front end. Même si son extension avec les bibliothèques jQuery lui a donné un vent de jouvence, certains langages plus dynamiques ont le vent en poupe ces dernières années

CHAPITRE II : HTML/CSS

Pour créer un site web, on doit donner des instructions à l'ordinateur. Il ne suffit pas simplement de taper le texte qui devra figurer dans le site (comme on le ferait dans un traitement de texte Word, par exemple), il faut aussi indiquer où placer ce texte, insérer des images, faire des liens entre les pages, etc.

I. Rôles de HTML

HTML (*HyperText Mark up Language*) : il a fait son apparition dès 1991 lors du lancement du Web. Son rôle est de gérer et organisé le contenu. C'est donc en HTML que vous écrirez ce qui doit être affiché sur la page : du texte, des liens, des images … Vous direz par exemple : « Ceci est mon titre, ceci est mon menu, voici le texte principal de la page, voici une image à afficher, etc. ».
XHTML Il s 'agit d'une variante du HTML qui se veut plus rigoureuse e et qui est donc un peu plus délicate à manipuler.

II. Différentes du HTML

HTML 1: c'est la toute première vers ion créée par Tim Berners -Lee en 1991.
HTML 2 : la deuxième vers ion du HTML apparaît en 1994 et prend fin en 1996 avec l'apparition du HTML 3.0. C'est cette version qui posera en fait les bases des versions suivantes du HTML. Les règles et le fonctionnement de cette version sont donnés par le W3C (tandis que la première vers ion a été créée par un seul homme).
HTML 3: apparue en 1996, cette nouvelle version du HTML rajoute de nombreuses possibilités au langage comme les tableaux, les applets, les scripts, le positionnement du texte autour des images, etc.
HTML 4 : il s 'agit de la vers ion la plus répandue du HTML (plus précisément, il s'agit de HTML 4.01). Elle apparaît pour la première fois en 1998 et propose l'utilisation de frames (qui découpent une page web en plusieurs parties), des tableaux plus complexes, des améliorations sur les formulaires, etc. Mais surtout, cette version permet pour la première fois d'exploiter des feuilles de style, notre fameux CSS !
HTML 5: c'est la dernière version. Elle fait beaucoup parler d'elle car elle apporte de nombreuses améliorations comme la possibilité d'inclure facilement des vidéos, un meilleur agencement du contenu, de nouvelles fonctionnalités pour les formulaires, etc. C'est cette version que nous allons découvrir ensemble.

III. Premiers pas avec html

III.1 Les balises

Les balises sont des bouts de code permettant de donner une structure bien particulière a une partie de notre site. Les balises se repèrent facilement. Elles sont entourées de « chevrons », c'est-à-dire des symboles < et >, comme ceci : **<balise>.** On distingue deux types de balises: les balises en paires et les balises orphelines.

III.1.1 Les balises paires

Elles s'ouvrent, contiennent du texte, et se ferment plus loin. Voici à quoi elles ressemblent :

```
10 <body>
11
12
13
14 </body>
```

On distingue une balise ouvrante *(<body>)* et une balise fermante *(</body>)* qui indique que le corps se termine. Cela signifie pour l'ordinateur que tout ce qui n'est pas entre ces deux balises

III.1.2 Les balises orphelines

Ce sont des balises qui servent le plus souvent à insérer un élément à un endroit précis (par exemple une image). Il n'est pas nécessaire de délimiter le début et la fin de l'image, on veut juste dire à l'ordinateur « Insère une image ici ».
Une balise orpheline s'écrit comme ceci :

```
<image />
```

II.2 Les attributs

Les attributs sont un peu les options des balises. Ils viennent les compléter pour donner des informations supplémentaires.
L'attribut se place après le nom de la balise ouvrante et a le plus souvent une valeur, comme ceci :

```
<balise attribut="valeur">
```

De manière plus pratique, on a :

```
<image nom="photo.jpg" />
```

On peut ajouter plusieurs attributs à une balise html en fonction du résultat qu'on veut obtenir.

Il existe des attributs particuliers :

- Id : Permet de différencier un élément de tous les autres de la mêmecatégories
- class

II.3 Structure d'une page html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>

    <body>

    </body>
</html>
```

La toute première ligne s 'appelle le **doctype**. Elle est indispensable car c'est elle qui indique qu'il s'agit bien d'une page web HTML.
Ce n'est pas vraiment une balise comme les autres car elle commence par un point d'exclamation.

Une page web est constituée de deux parties:

- **L'en-tête <head>< /head>**: cette section donne quelques informations générales sur la page comme son titre, l'encodage (pour la gestion des caractères spéciaux), etc. Cette section est généralement assez courte. Les informations que contient l'en-tête ne sont pas affichées sur la page, ce sont simplement des

informations générales à destination de l'ordinateur. Elles sont cependant très importantes!

- **Le corps <body></body>**: c'est là que se trouve la partie principale de la page. Tout ce que nous écrirons ici sera affiché à l'écran. C'est à l'intérieur du corps que nous écrirons la majeure partie de notre code.

IV. Mise en place de la page html

IV. 1 L'entête

La structure de l'en-tête est décrite dans le RFC 2616[i]. Il est délimité par les balises <head></head>. Les éléments ajoutés dans cette section n'apparaissent pas sur le site web. La structure interne de l'entête est :

```
<head>
  <meta charset="utf-8" />
  <title>Inserer le site</title>
  <link rel="stylesheet" href="" />
</head>
```

- **L'encodage** : IL est symbolisé par <meta charset="utf-8" />. Il y a plusieurs techniques d'encodage en fonction des langues: ISO-8859-1, OEM 775, Windows -1253… Une seule cependant devrait être utilisée aujourd'hui autant que possible: UTF-8. Cette méthode d'encodage permet d'afficher sans aucun problème pratiquement tous les symboles de toutes les langues de notre planète! C'est pour cela que j'ai indiqué utf-8 dans cette balise.

```
<meta charset="utf-8" />
```

Cette information est très fortement recommandée.

- **Le titre** est compris entre les balises <title>Insérer le site</title>
Habituellement, le titre de la page est affiché dans la barre de titre du navigateur (tout en haut), et lorsque le navigateur gère les onglets, dans l'étiquette des onglets. **Cet élément est obligatoire.**
- **Feuille de style :** Lorsque la page utilise une feuille de style située dans un autre fichier, on utilise la balise.

```
- <link rel= "stylesheet" type = "text/css" href="nom_du_fichier.css">
```

- **Description de la page :** La description du contenu de la page est indiquée avec la balise suivante

```
<meta name = "description"
   content = "phrase de description" >
```

- **Mots-clefs :** Les mots-clefs servent à référencer la page. Cependant, de nombreux moteurs de recherche n'ont plus recours aux mots-clefs car des auteurs utilisaient des mots-clefs sans rapport avec le contenu afin d'augmenter la fréquentation de leur page. Les mots-clefs sont indiqués avec la balise suivante

```
<meta name = "keywords" content = "liste de mots-clefs" >
```

- **Robots :** Le robot sert à gérer le référencement de la page. Il prend comme argument follow (permet au robot de suivre les liens de la page), index (permet au robot d'indexer la page), les arguments nofollow et noindex sont les contraires (ne pas suivre et ne pas indexer). Les deux derniers arguments sont all et none qui, comme leur noms l'indique, active ou désactive les deux fonctions. Les instructions pour robots sont indiquées avec la balise suivante

```
<meta name = "robots" content = "all|(no)follow|(no)index|none" >
```

- **La balise Link**: Elle sert à relier le html à sa feuille de style css ou à d'autres documents externes qui seront utilisé en général dans le rendu de la page.

<link href="/dossier/fichier.css" rel="stylesheet" type="text/css" media="screen"/>

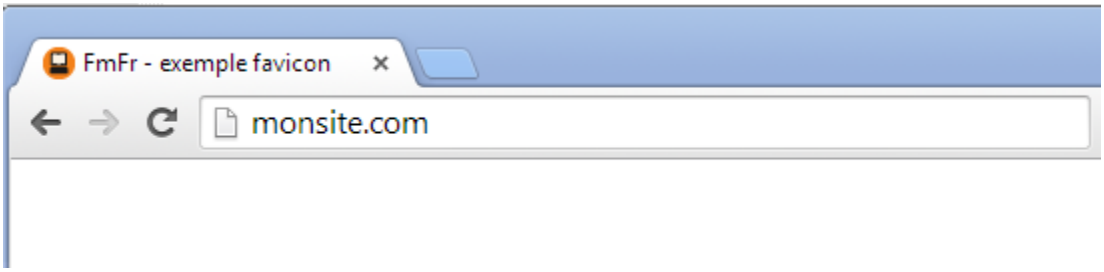
✓ **Ajouter le logo de son site dans la barre de titre et les onglets :**

La favicon est la petite icône qui apparaît dans la barre des titres ou dans les onglets d'un navigateur, voire éventuellement les historiques. Elle est souvent basée sur le logo de l'entreprise ou de la structure qui possède le site et peut être de taille variée allant de 16 pixels à 128.

<link rel="shortcut icon" href="dossier/fichier.ico" type="image/x-icon"/>

Le format de l'icône pourra en général être Gif ou PNG car JPEG n'acceptant les transparences, il ne conviendra pas pour les icones non rectangulaires.

Notre code d'entête devient alors :



- ✓ **Faciliter l'abonnement au site en liant un flux rss**

<link rel="alternate" type="application/rss+xml"
href="http://www.votreur/fichier.xml" title="Flux Rss de votre site />

Les valeurs d'attributs pour les balises link sont très nombreuses. Elles sont systématiquement accompagnées de :

- type : définissant le type mime du fichier lié.
- href : définissant l'adresse du fichier à lier.
- rel : mentionne le type de relation.

IV.2 Le corps

Le corps <*body*></*body*> est la partie de la structure du site contenant tous les éléments qui doivent s'afficher sur l'interface du client.

Les éléments relatifs au contenu du body se verront dans le chapitre IV. Dans cette section nous nous appesantirons sur quelques balises nécessaires a la conception d'un site web.

IV.2. Autres balises utiles

- **Les paragraphes** : Bonjour et bienvenue sur mon site !
- **Saut de ligne** : Le retour à la ligne peut se faire en utilisant la balise < /p> ou la balise orpheline
- **Les titres** : il en existe trois niveau en fonction du niveau d'importance dudit titre :
  - ✓ <h1> </h1> : signifie « titre très important ». En général, on s'en sert pour afficher le titre de la page au début de celle-ci.
  - ✓ <h2></h2> : Signifie « titre important ».
  - ✓ <h3> </h3> : pareil, c'est un titre un peu moins important (on peut dire un « sous -titre » si vous voulez).
  - ✓ <h4> </h4> : titre encore moins important.
  - ✓ <h5> </h5> : titre pas important.
  - ✓ <h6> </h6> : titre vraiment, mais alors là vraiment pas important du tout
- La mise en valeur : Il existe plusieurs balises de mise en valeur :
  - ✓ Mise un peu en valeur
  - ✓ Mise en valeur plus corsée :
- **Les listes** : Il en existe de deux types
  - ✓ Les listes non-ordonnées: C'est un système qui nous permet de créer une liste d'éléments sans notion d'ordre (il n'y a pas de « premier » ni de « dernier »). Créer une liste non ordonnée est très simple. Il suffit d'utiliser la balise <*ul*> que l'on referme un peu plus loin avec </*ul*>. Commencez donc à taper ceci :

```
<ul>
    <li>Fraises</li>
    <li>Framboises</li>
    <li>Cerises</li>
</ul>
```

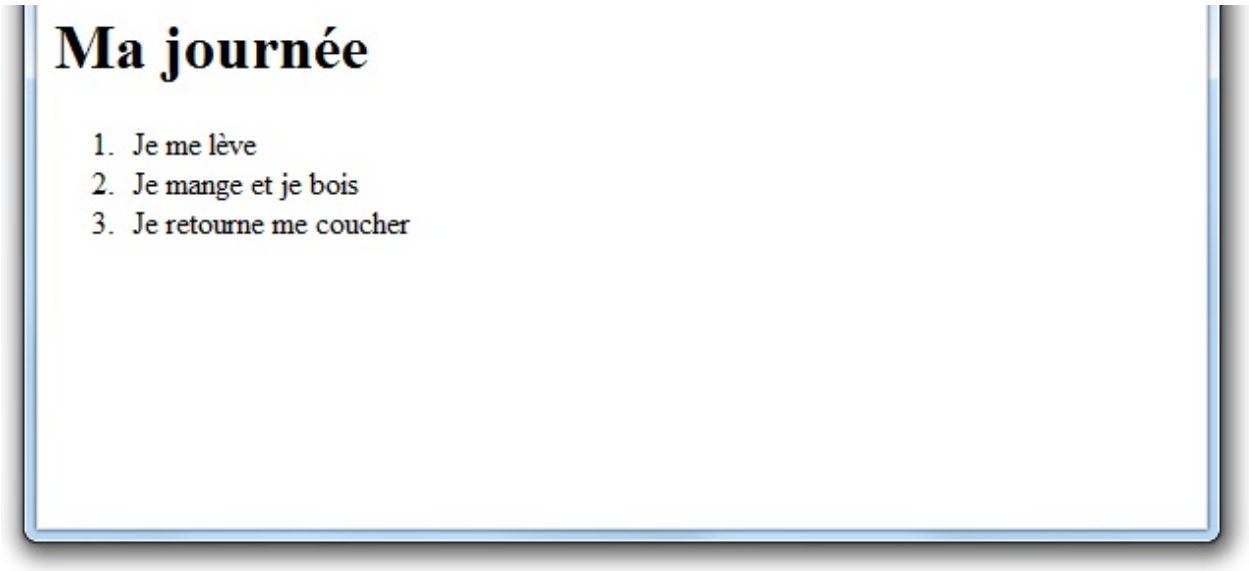
- Fraises
- Framboises
- Cerises

 délimite toute la liste
< /li> délimite un élément d'une liste à puce

- ✓ Les listes ordonnées :

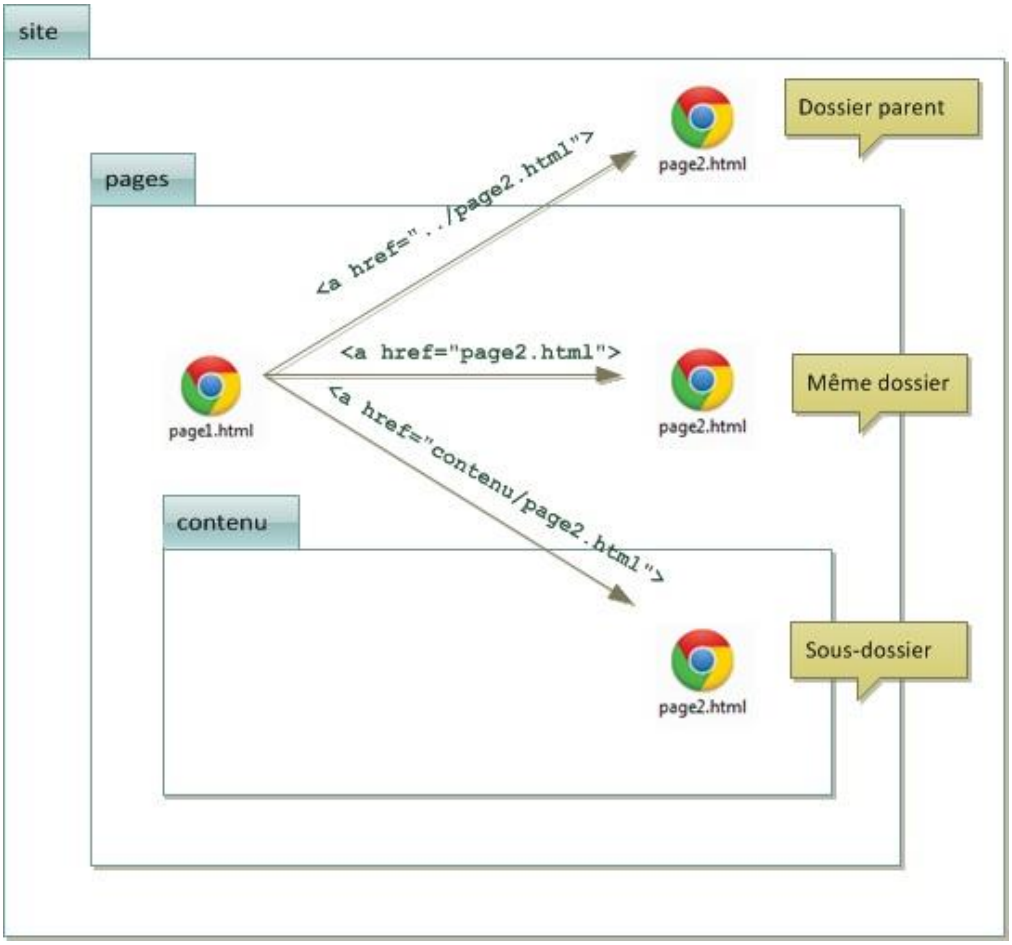
```
<h1>Ma journée</h1>

<ol>
    <li>Je me lève.</li>
    <li>Je mange et je bois.</li>
    <li>Je retourne me coucher.</li>
</ol>
```



V. Les liens

Nous avons vu dans les sections précédentes que le html seul ne peut nous permettre de créer un site web. Les liens permettent de pointer vers une autre ressource en vue d'obtenir des fonctionnalités de celle-ci. On peut schématiquement avoir les liens suivants :



- Même niveau d'arborescence : <a href="page.html"><a/>
- Un cran en dessous : <a href="contenu/page.html"><a/>
- Un cran au-dessus : <a href=../page.html"><a/>

Il est également possible de mettre en place un lien vers une ressource en ligne. C'est le cas des liens qu'on peut effectuer pour se connecter a JQuery ou a bootstrap.

IV.2 Les ancres

Les ancres sont utilisées pour une section bien précise d'un site internet. C'est un outil très utile lors de la mise en place d'un site sur une seule page web

En effet, il peut alors être utile de faire un lien amenant plus bas dans la même page pour que le visiteur puisse sauter directement à la partie qui l'intéresse.

Pour créer une ancre, il suffit de rajouter **l'attribut id** à une balise qui va alors servir de repère. Ce peut être n'importe quelle balise, un titre par exemple.

Utilisez l'attribut id pour donner un nom à l'ancre. Cela nous servira ensuite pour faire un lien vers cette ancre. Par exemple :

```
<h2 id="mon_ancre">Titre</h2>
```

On pourra ensuite accéder grâce à la balise

```
<a href="#mon_ancre">Aller vers l'ancre</a>
```

Lien vers une ancre situé dans une page :

L'idée, c'est de faire un lien qui ouvre une autre page ET qui amène directement à une ancre située plus bas sur cette page.
En pratique c'est assez simple à faire : il suffit de taper le nom de la page, suivi d'un dièse (#), suivi du nom de l'ancre.

<a href="ancres.html#lic2x"><a/>

CHAPITRE II : CSS

I. Rôles de CSS

CSS (*Cascading Style Sheets*, aussi appelées Feuilles *de style*) : le rôle du CSS est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleurs, taille du texte…). Ce langage est venu compléter le HTML en 1996.

II. Evolution du CSS

CSS 1 : dès 1996, on dispose de la première version du CSS. Elle pose les bases de ce langage qui permet de présenter sa page web, comme les couleurs, les marges, les polices de caractères, etc.
CSS 2 : apparue en 1999 puis complétée par CSS 2.1, cette nouvelle version de CSS rajoute de nombreuses options. On peut désormais utiliser de techniques de positionnement très précises, qui nous permettent d'afficher des éléments où on le souhaite sur la page.
CSS 3 : c'est la dernière version, qui apporte des fonctionnalités particulièrement attendues comme les bordures arrondies, les dégradés, les ombres, etc

III. Mise en place du CSS

Le CSS peut s'agréger de trois manière dans un site.

III.1 Insertion du css

Le CSS peut être insérer à partir d'un fichier CSS. Dans ce cas on crée un fichier un fichier par exemple style.css qu'on enregistre. Ensuite on accède à ce fichier en fonction de son emplacement. C'est la méthode la plus recommande.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Premiers tests du CSS</title>
    </head>

    <body>
        <h1>Mon super site</h1>

        <p>Bonjour et bienvenue sur mon site !</p>
        <p>Pour le moment, mon site est un peu <em>vide</em>.
Patientez encore un peu !</p>
    </body>
</html>
```

Mise en forme css dans les balises html : Cette méthode est fortement déconseillée
Mise en forme dans l'entête : Cette méthode est également déconseillée

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <style>
            p
            {
                color: blue;
            }
        </style>
```

III.2 Appliquer un style

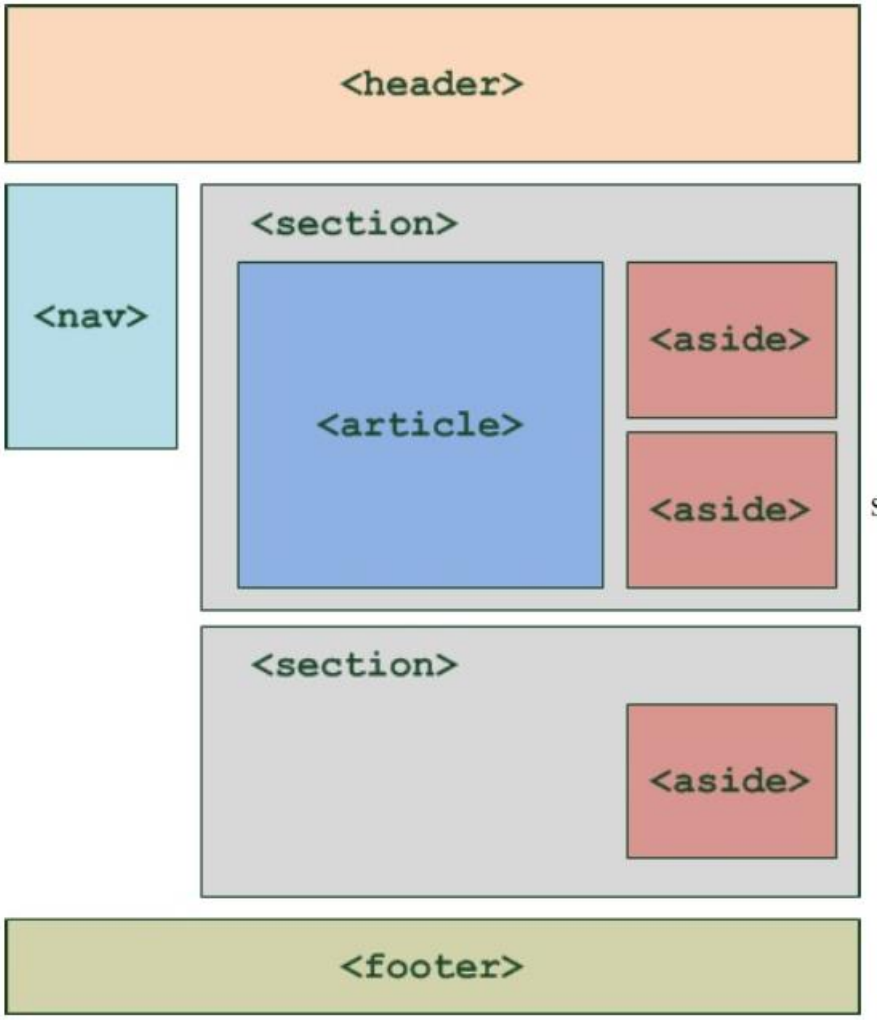
Il est possible d'appliquer un style a plusieurs éléments : une balise , un élément particulier, un groupe d'élément.

```
balise1
{
   propriete1: valeur1;
   propriete2: valeur2;
   propriete3: valeur3;
}
```

CHAPITRE III : MISE EN PLACE D'UN SITE

La structure d'une page html est effectué grâces aux balises introduites dans le html5. Les différentes structures était mise en place par le passe grâce à la balise générique < /div>. Elle peut l'être toujours mais risque dans un futur propre de ne pas être optimal.

Si les effets de ces balises html5 ne sont pas très visible pour le moment, elles permettent de dire au navigateur la section dans laquelle nous nous trouvons.



Structure basique d'une page web

L'entête du site

Il est symbolisé par la balise paire <header></header>.

Dans cette section on peut mettre le logo, le menu de navigation, la barre de recherche grâce a des balises html spécifiques.

```
<header>
    <div id="titre_principal">
        <img src="login_logo.png" id="logo"/>
        <h1>Nom du site</h1>
        <h2>Slogan</h2>
    </div>
    <nav>
        <ul>
            <li>Accueil</li>
            <li>Nos services</li>
            <li>Notre &eacute;quipe</li>
            <li>Contact</li>
        </ul>
    </nav>
</header>
```

Le CSS associe

```
#titre_principal{
    display: inline-block;
}
#logo{
```

```
}
#logo, header h1{
    display: inline-block;
}
nav{
    display: inline-block;
    /*width: 740 px;*/
    position: relative;
    text-align: right;
}
nav ul{
    list-style-type: none;
}
nav li{
    display: inline-block;
    margin-right: 15px;
}
```

Le corps du site

Le corps du site est l'élément le plus parlant du site web. IL est constitué de bannière, d'article, de formulaire…

L'élément section

L'élément section est utilisé pour regrouper du contenu de manière thématique. Cela rappelle l'élément div, souvent utilisé comme un conteneur de contenu générique. La différence, c'est que div n'a aucune signification sémantique ; il ne nous dit rien de son

contenu. L'élément section, en revanche, est utilisé explicitement pour regrouper du contenu apparenté.

On pourra remplacer quelques-uns des éléments div par des éléments section, mais il faudrait toujours que tout le contenu soit apparenté.

```
<section>
<h1>UFR Math Info</h1>
<p>Faculté de Mathématique et informatique »
Pas de littéraire. </p>
<p>Le doyen</p>
</section>
```

L'élément header

header est un conteneur pour « un groupe d'outils d'introduction ou de navigation». Il y a une différence cruciale entre l'élément header de l'HTML5 et l'usage répandu du mot « en-tête » head. Il n'y a généralement qu'une seule <entete> head sur une page, mais un document peut comporter plusieurs éléments header. On pourra utiliser un élément header dans un élément section.

La spécification décrit l'élément section comme « un regroupement de contenu thématique, comprenant généralement un en-tête ».

```
<section>
  <header>
    <h1> UFR Math Info </h1>
  </header>
  <p> Faculté de Mathématique et informatique »
Pas de littéraire </p>
  <p> Le doyen </p>
</section>
```

Un header apparaîtra normalement en haut d'un document ou d'une section, mais ce n'est pas obligatoire. Il est défini par son contenu (présentation ou aide à la navigation) plutôt que par sa position.

L'élément footer

Comme l'élément header, footer semble décrire une position, mais comme pour header, ce n'est pas le cas. L'élément footer contient plutôt des informations au sujet de l'élément qui le contient : auteur, informations de copyright, liens vers des contenus apparentés, etc.
L'HTML5 nous permet également d'en placer à l'intérieur des sections.

```
<section>
  <header>
    <h1> UFR Math Info </h1>
  </header>
  <p> Faculté de Mathématique et informatique »
Pas de littéraire </p>
<footer>
  <p> Le doyen </p>
</footer>
</section>
```

L'élément aside

L'élément aside est une sorte d'encadré. « encadré », ne correspond pas à la position. Il ne suffit pas que le contenu apparaisse à gauche ou à droite du contenu principal pour utiliser l'élément aside.
L'élément aside doit être utilisé pour du contenu indirectement apparenté. Si vous avez un morceau de contenu que vous considérez comme différent du contenu principal, l'élément aside est probablement le bon conteneur.

L'élément nav

L'élément nav contient des informations de navigation, généralement une liste de liens.
L'élément nav est conçu pour les informations de navigation principales. Une simple liste de liens ne justifie pas l'utilisation de l'élément nav. En revanche, les liens permettant de naviguer sur un site doivent presque obligatoirement utiliser l'élément nav.
La plupart du temps un élément nav apparaîtra dans un élément header. Cela paraît logique si l'on considère que l'élément header peut être utilisé pour les « aides à la navigation ».

L'élément article

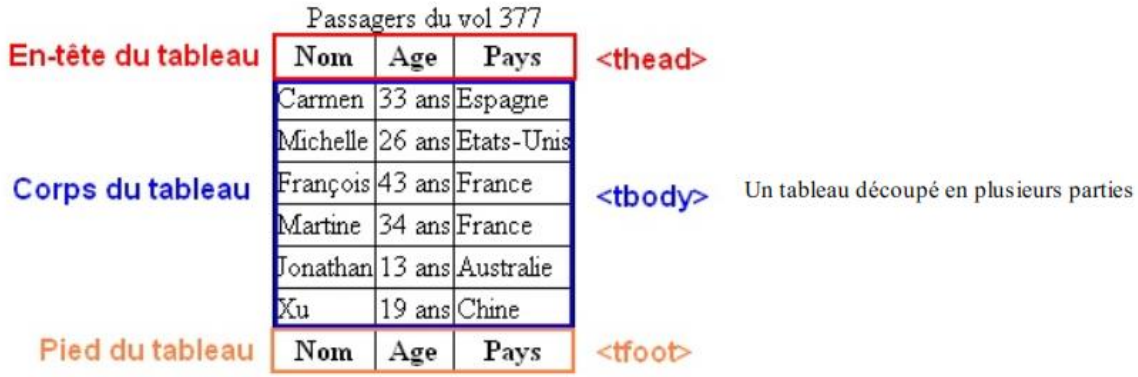
Il peut être utile de considérer header, footer, nav et aside comme des formes spécialisées de l'élément section. Une section est un morceau générique de contenu apparenté, tandis que les éléments header, footer, nav et aside sont des morceaux spécifiques de contenu apparenté.
L'élément article est un autre type de section spécialisé. On l'utilise pour le contenu apparenté autonome. Le problème étant de déterminer ce qui constitue un contenu « autonome ».

Les tableaux

La première balise à connaître est <table> </table>. C'est cette balise qui permet d'indiquer le début et la fin d'un tableau.

Cette balise est de type bloc, il faut donc la placer en dehors d'un paragraphe. En HTML, un tableau se construit ligne par ligne. Dans chaque ligne (<tr>), on indique le contenu des différentes cellules (<td>).

- **<tr> </tr>** : indique le début et la fin d'une ligne du tableau ;
- **<td> </td>** : indique le début et la fin du contenu d'une cellule




```
<table>
  <caption>Passagers du vol 377</caption>
  <thead> <!-- En-tête du tableau -->
    <tr>
      <th>Nom</th>
      <th>Âge</th>
```

```
        <th>Pays</th>
    </tr>
</thead>
<tfoot> <!-- Pied de tableau -->
    <tr>
        <th>Nom</th>
        <th>Âge</th>
        <th>Pays</th>
    </tr>
</tfoot>
<tbody> <!-- Corps du tableau -->
    <tr>
        <td>Carmen</td>
        <td>33 ans</td>
        <td>Espagne</td>
    </tr>
    <tr>
        <td>Michelle</td>
        <td>26 ans</td>
        <td>États-Unis</td>
    </tr>
    <tr>
        <td>François</td>
        <td>43 ans</td>
        <td>France</td>
    </tr>
    <tr>
        <td>Martine</td>
        <td>34 ans</td>
        <td>France</td>
    </tr>
    <tr>
        <td>Jonathan</td>
        <td>13 ans</td>
        <td>Australie</td>
    </tr>
    <tr>
        <td>Xu</td>
        <td>19 ans</td>
        <td>Chine</td>
```

```
    </tr>
  </tbody>
</table>
```

Les formulaires

Tout comme les tableaux, les formulaires sont une fonctionnalité évolué du html. Ils permettent de gérer les interactions entre le client et la base de donnée. Un formulaire est créé à partir de la balise **<form> </form>**.

- Les méthodes de transfert de donnée sont :
  - **method="get"** : C'est une méthode en général assez peu adaptée car elle est limitée à 255 caractères. La particularité vient du fait que les informations seront envoyées dans l'adresse de la page (http://…).
  - **method="post"** : C'est la méthode la plus utilisée pour les formulaires car elle permet d'envoyer un grand nombre d'informations . Les données saisies dans le formulaire ne transitent pas par la barre d'adresse.
- **action**: c'est l'adresse de la page ou du programme qui va traiter les informations

```
<form  method="post" action=" ">
  <p>
    <label for="login">Login :</label>
    <input type="text" name="login" id="login" />

    <br />
    <label for="pass">Mot de passe :</label>
    <input type="password" name="pass" id="pass" />

  </p>
</form>
```

Il existe plusieurs autres types de zone de texte : email, url, numero de telephone, date….

Balise	Description
`<form>`	Formulaire
`<fieldset>`	Groupe de champs
`<legend>`	Titre d'un groupe de champs
`<label>`	Libellé d'un champ
`<input />`	Champ de formulaire (texte, mot de passe, case à cocher, bouton, etc.)
`<textarea>`	Zone de saisie multiligne
`<select>`	Liste déroulante
`<option>`	Élément d'une liste déroulante
`<optgroup>`	Groupe d'éléments d'une liste déroulante

Le pied de page

À l'inverse de l'en-tête, le pied de page se trouve en général tout en bas du document. On y trouve des informations comme des liens de contact, le nom de l'auteur, les mentions légales, etc.

Le model des boites

block : une balise de type crée automatiquement un retour à la ligne avant et après. Il suffit d'imaginer tout simplement un bloc. La page web sera en fait constituée d'une série de blocs les uns à la suite des autres. Mais, il est possible de mettre un bloc à l'intérieur d'un autre, ce qui va augmenter considérablement nos possibilités pour créer le design du site !
inline : une balise de type inline se trouve obligatoirement à l'intérieur d'une balise block. Une balise inline ne crée pas de retour à la ligne, le texte qui se trouve à l'intérieur s'écrit donc à la suite du texte précédent, sur la même ligne (c'est pour cela que l'on parle de balise « en ligne »)

Balises block	Balises inline
<p>	
<footer>	
<h1>	<mark>
<h2>	<a>
<article>	<img />
…	…

Positionnement

La propriete css display permet de transformer n'importe quel élément inline en block et vice versa

Display prends les valeurs suivantes :

Valeur	Exemples	Description
`inline`	**<a>**, ****, ****…	Eléments d'une ligne. Se placent les uns à côté des autres.
`block`	**<p>**, ** **, **<section>**…	Eléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
`inline-block`	**<select>**, **<input>**	Eléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être redimensionnés (comme les blocs).
`none`	**<head>**	Eléments non affichés.

CHAPITRE IV : PHP/MYSQL

L'intérêt majeur de PHP est son interfaçage avec un grand nombre de bases de données. Pour la visualisation de vos tables et bases de données MySQL nous vous conseillons l'emploi de l'excellent phpMyAdmin qui est une interface en ligne très simple d'utilisation et redoutable dans ses possibilités.

Nous ne passerons pas en revue les différentes commandes SQL mais nous privilégierons leurs interactions avec PHP. Une bonne connaissance préalable des SGBD et des types de données que l'on y trouve est donc nécessaire.

Enfin, il vous appartient de savoir comment on remplit une base de données manuellement pour son exploitation avec PHP.

Présentation des bases de données

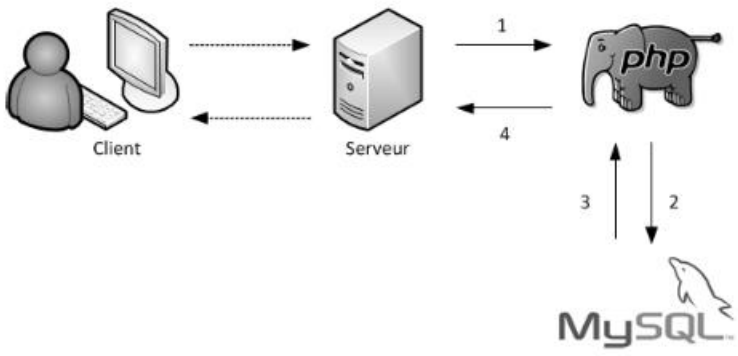
La base de données (BDD) est un système qui enregistre des informations. Les SGBD sont les programmes qui se chargent du stockage de vos données.

Les plus connus sont, pour rappel :

- MySQL : libre et gratuit, c'est probablement le SGBD le plus connu. Nous l'utiliserons dans cette partie ;
- PostgreSQL : libre et gratuit comme MySQL, avec plus de fonctionnalités mais un peu moins connu ;
- SQLite : libre et gratuit, très léger mais très limité en fonctionnalités ;
- Oracle : utilisé par les très grosses entreprises ; sans aucun doute un des SGBD les plus complets, mais il n'est pas libre et on le paie le plus souvent très cher ;
- Microsoft SQL Server : le SGBD de Microsoft.

Il faut donc choisir le SGBD que vous allez utiliser pour stocker les données.

MySQL ne communique pas directement avec l'interface utilisateur. Pour cela, on uilisera le langage SQL



Structure d'une base de données

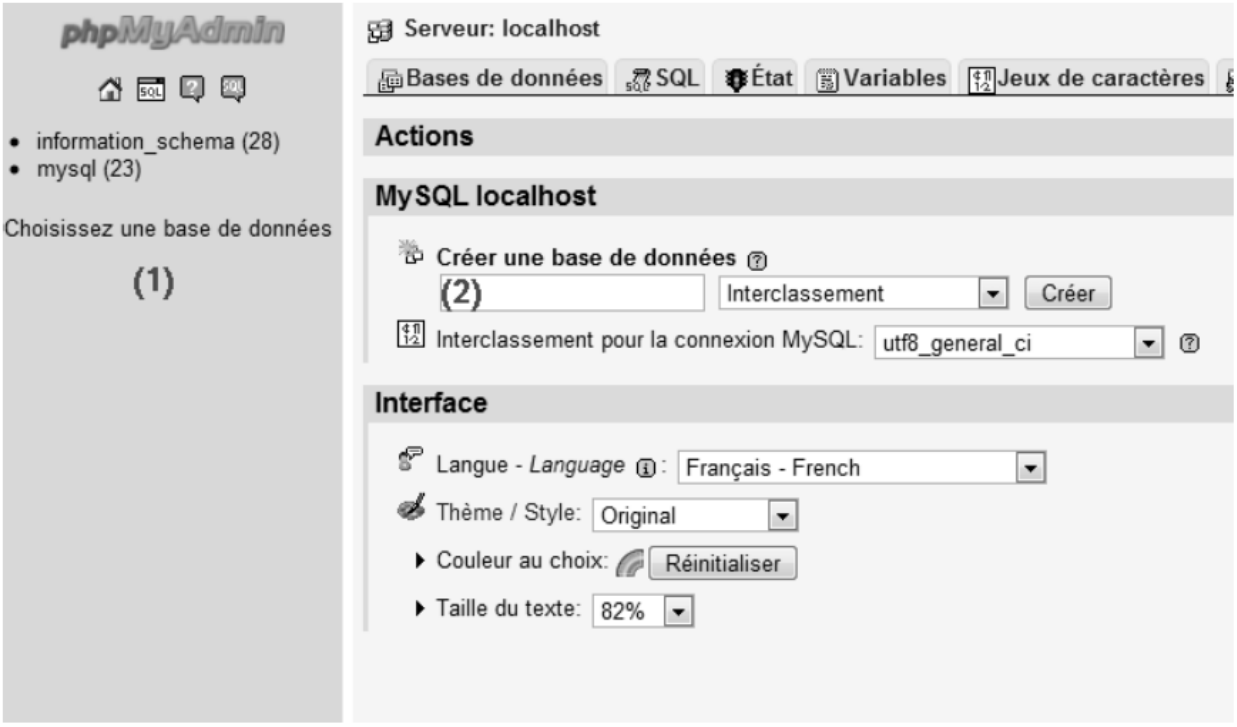
Avec les bases de données, il faut utiliser un vocabulaire précis.

- **La base de donnée** est l'espace de stockage de toute l'information.
- **Les tables** représentent les différentes entités.
- Chaque table est un tableau où les colonnes sont appelées « champs » et les lignes « entrées ».

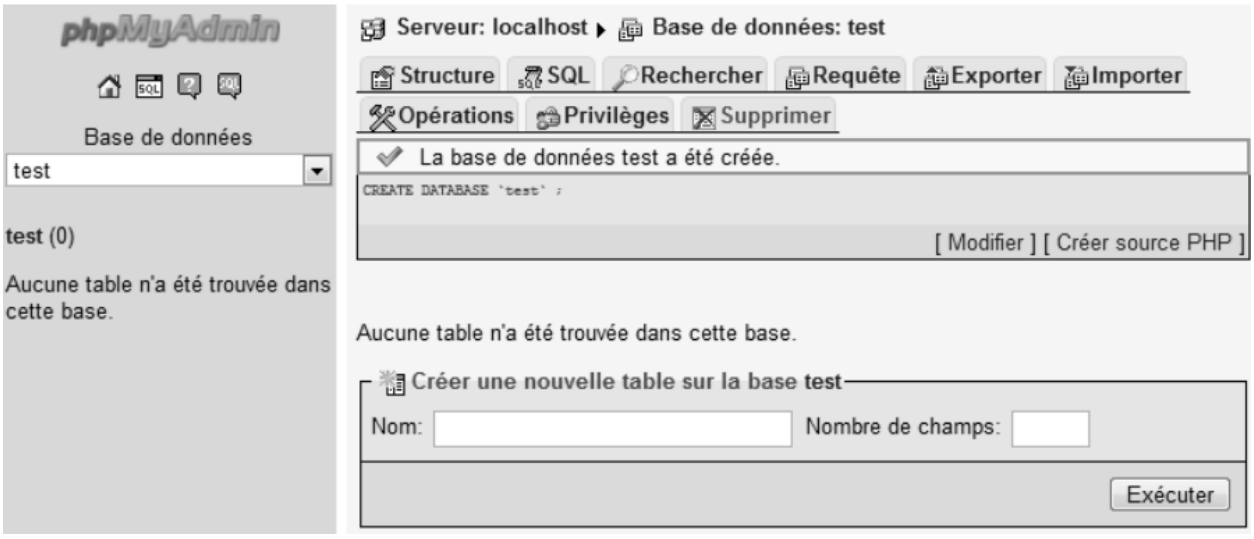
Opération sur les bases de donnée

Le SQL comprend plusieurs requêtes de manipulation de données. Cependant dans le cadre de cours, nous nous restreindrons aux opérations suivantes :

- Création : création de bases de données, de tables…
- Sélection : Il s'agit de la sélection globale ou certaines contraintes.
- Insertion : Il s'agit d'ajouter des lignes de données aux différentes tables.
- Suppression : Il s'agit de supprimer une/plusieurs lignes de codes
- Modification : Il s'agit de la modification d'une ou plusieurs lignes de code

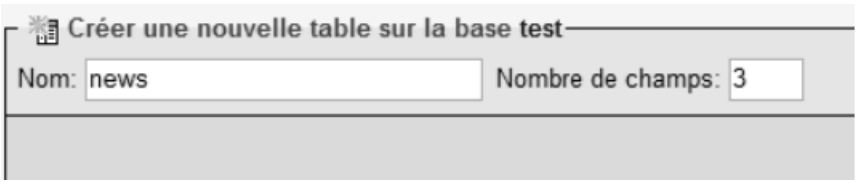


1. **Liste des bases** : c'est la liste de vos bases de données. Le nombre entre parenthèses est le nombre de tables qu'il y a dans la base. Sur ma capture d'écran, on a donc deux bases : information_schema qui contient 28 tables n'y touchez pas, elles servent au fonctionnement interne de MySQL., et mysql, qui en contient 23.
2. **Créer une base** : pour créer une nouvelle base de données, entrez un nom dans le champ de formulaire à droite, cliquez sur « Créer »



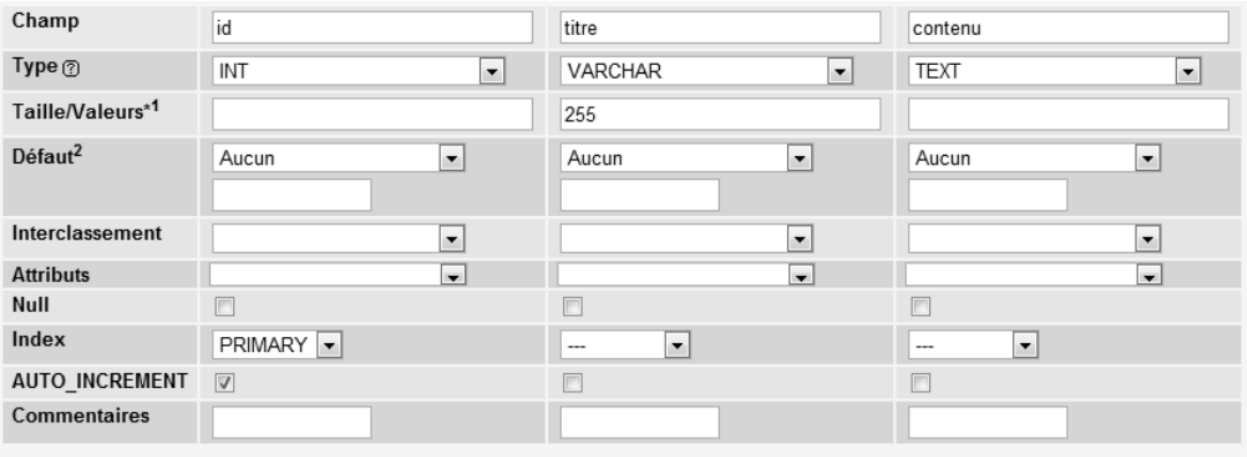
La base de test a été créée, vide

Dans le champ « Créer une nouvelle table sur la base test », entrez le nom news et le nombre de champs 3, comme vous le montre la figure ci-dessus



Créer une table

La table n'est pas immédiatement créée : il faut maintenant indiquer le nom des champs et les données qu'ils peuvent contenir.



Création d'une table MySQL

Chaque colonne représente un champ. Nous avons demandé trois champs, il y a donc trois colonnes.
phpMyAdmin vous demande beaucoup d'informations mais rassurez-vous, il n'est pas nécessaire de tout remplir. La plupart du temps, les sections les plus intéressantes seront :

- **Champ** : permet de définir le nom du champ (très important !) ;
- **Type** : le type de données que va stocker le champ (nombre entier, texte, date. . .) ;
- **Taille/Valeurs** : permet d'indiquer la taille maximale du champ, utile pour le type VARCHAR notamment, afin de limiter le nombre de caractères autorisés ;
- **Index** : active l'indexation du champ. Ce mot barbare signifie dans les grandes lignes que votre champ sera adapté aux recherches. Le plus souvent, on utilise l'index PRIMARY sur les champs de type id ;

- **AUTO_INCREMENT** : permet au champ de s'incrémenter tout seul à chaque nouvelle entrée. On l'utilise fréquemment sur les champs de type id.

Toute table doit posséder un champ qui joue le rôle de clé primaire. La clé primaire permet d'identifier de manière unique une entrée dans la table. En général, on utilise le champ id comme clé primaire, comme on vient de le faire.

Présentation de l'interface de PhP MyAdmin

Utiliser

Pour se connecter à MySQL via PHP on utilise une API (Application Programming Interface, ou interface de programmation d'application) qui définit les classes, méthodes, fonctions et variables dont votre application va faire usage pour exécuter différentes tâches.

Les API peuvent être procédurale ou orientée objet. Avec une API procédurale, vous pouvez appeler les fonctions pour exécuter des commandes ; avec une API orientée objet, vous devez créer des objets et appeler les méthodes de ces objets.

Il existe trois API principales pour se connecter à MySQL :

- L'extension mysql (pour les versions PHP < 4.1.3) : ce sont des fonctions qui permettent d'accéder à une base de données MySQL et donc de communiquer avec MySQL. Leur nom commence toujours par mysql_. Toutefois, ces fonctions sont vieilles et on recommande de ne plus les utiliser aujourd'hui.
- L'extension mysqli (mysql ameliorer pour les versions PHP > 5). Ce sont des fonctions améliorées d'accès à MySQL. Elles proposent plus de fonctionnalités et sont plus à jour.

- PHP Data Objects (PDO) qui a des fonctionnalités moins avancées que mysqli mais permet de travailler avec différentes bases de données en conservant le même code. C'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle.

Etapes pour se connecter à une base de données

L'exploitation de MySQL avec PHP s'effectue en plusieurs étapes :

- Ouverture d'une connexion à MySQL et sélection de la base de données
- Requête sur la base de données
- Exploitation des résultats de la requête
- Fermeture de la connexion à MySQL

Ouvrir une connexion la base de données MySQL

```
< ?php
$source = 'mysql::host=localhost ;dbname=lic2Miage';
$utilisateur = 'root';
$pwd ="
?>
```

Ou

```
< ?php
$source = new PDO('mysql:host=localhost;dbname=test', 'root', '');
?>
```

- **Host="** : le nom de l'hôte : c'est l'adresse de l'ordinateur où MySQL est installé (comme une adresse IP). Le plus souvent, MySQL est installé sur le même ordinateur que PHP : dans ce cas, mettez la valeur localhost.

Néanmoins, il est possible que votre hébergeur web vous indique une autre valeur à renseigner (qui ressemblerait à ceci : sql.hebergeur.com). Dans ce cas, il faudra modifier cette valeur lorsque vous enverrez votre site sur le Web ;

- **Dbname=" "** : c'est le nom de la base de données à laquelle vous voulez vous connecter. Dans notre cas, la base s'appelle test. Nous l'avons créée avec phpMyAdmin dans
- **Le login 'root'** : il permet de vous identifier. Renseignez-vous auprès de votre hébergeur pour le connaître. Le plus souvent (chez un hébergeur gratuit), c'est le même login que vous utilisez pour le FTP;
- **le mot de passe** : il y a des chances pour que le mot de passe soit le même que celui que vous utilisez pour accéder au FTP. Renseignez-vous auprès de votre hébergeur.

Tester la présence d'erreurs

```
<?php
try
{
$bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '');
}
catch (Exception $e)
{
die('Erreur : ' . $e->getMessage());
}
?>
```

PHP

Le code PHP vient s'insérer au milieu du code HTML. On va progressivement placer dans nos pages web des morceaux de code PHP à l'intérieur du HTML. Ces bouts de code PHP seront les parties dynamiques de la page, c'est-à-dire les parties qui peuvent changer toutes seules

La forme d'une balise PhP est < ?php ?>

Il existe d'autres balises pour utiliser du PHP, par exemple < ? ?>, <% %>…

Un commentaire est un texte que vous mettez pour vous dans le code PHP. Ce texte est ignoré, c'est-à-dire qu'il disparaît complètement lors de la génération de la page. Il n'y a que vous qui voyez ce texte. Il en existe deux types :

- Mono lignes : Pour indiquer que vous écrivez un commentaire sur une seule ligne, vous devez taper deux slashs : « // ».
- Multi lignes : Ce sont les plus pratiques si vous pensez écrire un commentaire sur plusieurs lignes, mais on peut aussi s'en servir pour écrire des commentaires d'une seule ligne. Il faut commencer par écrire /* puis refermer par */
-

Insertion de page

En PHP, nous pouvons facilement insérer d'autres pages à l'intérieur d'une page.

Le principe de fonctionnement des inclusions en PHP est plutôt simple à comprendre. Il consiste à découper différentes parties réutilisables de notre site et de les rappeler pour éviter de réécrire le code grace a la syntaxe :

```
<?php include("nomPage"); ?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.
↪ DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Mon super site</title>
       <meta http-equiv="Content-Type" content="text/html; charse
↪ >
   </head>

        <body>

        <?php include("entete.php"); ?>

        <?php include("menus.php"); ?>

        <div id="corps">
           <h1>Mon super site</h1>

                        <p>
               Bienvenue sur mon super site !<br />
               Vous allez adorer ici, c'est un site génial qui va
↪ euh... Je cherche encore un peu le thème de mon site. :-D
                        </p>
        </div>

        <?php include("pied_de_page.php"); ?>

   </body>
</html>
```

Les structures de contrôles

- **Les symboles à connaître**

Symbole	Signification
==	Est égal à
>	Est supérieur à
<	Est inférieur à
>=	Est supérieur ou égal à
<=	Est inférieur ou égal à
!=	Est différent de

- **Structure de contrôle simple**

```
<?php
$age = 8;
if ($age <= 12)
{
echo "Salut gamin !";
}
?>
```

- **Structure de contrôle complexe**

```
<?php
$age = 8;
if ($age <= 12) // SI l'âge est inférieur ou égal à 12
{
echo "Salut gamin ! Bienvenue sur mon site !<br />";
$autorisation_entrer = "Oui";
}
else // SINON
{
echo "Ceci est un site pour enfants, vous êtes trop vieux pour pouvoir
,! entrer. Au revoir !<br />";
$autorisation_entrer = "Non";
}
echo "Avez-vous l'autorisation d'entrer ? La réponse est : $autorisation_entrer"
,! ;
?>
```

Il y a un symbole qui permet de vérifier si la variable vaut false : le point d'exclamation (!). On écrit :

```
<?php
if (! $autorisation_entrer)
{
Instruction
}
?>
```

- **Des conditions multiples**

Mot-clé	Signification	Symbole équivalent
AND	Et	&&
OR	Ou	||

```
<?php
if ($age <= 12 AND $sexe == "garçon")
{
echo "Bienvenue sur le site de Captain Mégakill !";
}
elseif ($age <= 12 AND $sexe == "fille")
{
echo "C'est pas un site pour les filles ici, retourne jouer à la Barbie !";
}
?>
```

En règle générale, les structures de contrôle if… else… if suffisent à traiter n'importe quelle situation. Cependant il existe une structure de contrôle pour éviter la lourdeur et la répétition des if…else…if.

On pourra par exemple remplacer :

```
<?php
if ($note == 0)
{
```

```
echo "Tu es vraiment un gros Zéro !!!";
}
elseif ($note == 5)
{
echo "Tu es très mauvais";
}
elseif ($note == 7)
{
echo "Tu es mauvais";
}
elseif ($note == 10)
{
echo "Tu as pile poil la moyenne, c'est un peu juste...";
}
elseif ($note == 12)
{
echo "Tu es assez bon";
}
elseif ($note == 16)
}
?>
```

Par un switch qui est plus synthetique

```
<?php
$note = 10;
switch ($note) // on indique sur quelle variable on travaille
{
case 0: // dans le cas où $note vaut 0
echo "Tu es vraiment un gros Zér0 !!!";
break;
case 5: // dans le cas où $note vaut 5
echo "Tu es très mauvais";
break;
case 7: // dans le cas où $note vaut 7
echo "Tu es mauvais";
break;
case 10: // etc. etc.
echo "Tu as pile poil la moyenne, c'est un peu juste...";
break;
case 12:
echo "Tu es assez bon";
break;
case 16:
echo "Tu te débrouilles très bien !";
break;
```

```
case 20:
echo "Excellent travail, c'est parfait !";
break;
default:
echo "Désolé, je n'ai pas de message à afficher pour cette note";
}
?>
```

Les ternaires

Les ternaires sont des conditions condensées qui font un test sur une variable, et en fonction des résultats de ce test donnent une valeur à une autre variable. Elles sont cependant plus rarement utilisées.

```
<?php
$age = 24;
if ($age >= 18)
{
$majeur = true;
}
else
{
$majeur = false;
}
?>
```

Qui deviendra :

```
<?php
$age = 24;
$majeur = ($age >= 18) ? true : false;
?>
```

Les boucles

Les boucles simples(While)

C'est une structure qui fonctionne sur le même principe que les conditions (if... else). Une boucle permet de répéter de instructions plusieurs fois. Dans son mode de fonctionnement les instructions sont d'abord exécutées dans l'ordre, à la fin des instructions, on retourne à la première, on recommence à lire les instructions dans l'ordre ainsi de suite.

Tel qu'expose, il faudrait indiquer une condition de sortie au risque la boucle soit exécuté à l'infini tant que la condition n'est plus remplie, on sort de la boucle.

```
$apprenants = array( "Thamer", "Mohamed", "Mounira", "Samira", "Tarek" );
$i=0;
while(isset($apprenants[$i]) )
{
echo "L'élément".$i. " est $apprenants[$i]<br />";
$i++;
}
```

La boucle while, permet de parcourir un tableau d'une manière efficace dans le cas d'un tableau retourné après une requête sur une base de données :

- Dans le cas d'affichage d'une requête sur une base de donnée avec affichage grâce à la boucle while l'expression booléenne contenue dans l'instruction while est **isset($tab[$i]).**
- Cette expression prend la valeur **TRUE** tant que l'élément désigné par **$tab[$i]** existe. Sinon, elle prend la valeur **FALSE**, ce qui est le cas en fin de tableau.

Les boucles for

« for » est un autre type de boucle, dans une forme un peu plus condensée et plus commode à écrire, ce qui fait que for est assez fréquemment utilisé. Cependant, sachez que for et while donnent le même résultat et servent à la même chose : répéter des instructions en boucle. L'une peut paraître plus adaptée que l'autre dans certains cas. En reprenant l'exemple précèdent avec une boucle while, on obtient :

```
<?php
$apprenants = array( "Thamer", "Mohamed", "Mounira", "Samira", "Tarek" );
for($i = 0, $size = count($apprenants); $i < $size; ++$i) {
   echo $apprenants[$i].'<br>';
}
?>
```

La boucle while est plus simple et plus flexible : on peut faire tous les types de boucles avec mais on peut oublier de faire certaines étapes comme l'incrémentation de la variable.
En revanche, for est bien adapté quand on doit compter le nombre de fois que l'on répète les instructions et il permet de ne pas oublier de faire l'incrémentation pour augmenter la valeur de la variable !

La boucle foreach

La boucle foreach() n'est utilisable qu'à partir des versions 4 de PHP. Elle est efficace pour les tableaux associatifs mais fonctionne également pour les tableaux de valeur ou d'indice. Contrairement à la boucle for, la boucle foreach() ne nécessite pas de connaître par avance le nombre d'éléments du tableau à lire, ce qui fait qu'elle est l'une des boucles les plus utilise lors de l'affichage du résultat d'une requête sur une base de donnée. Sa syntaxe varie en fonction du type de tableau.

```
<?php
$apprenants = array( "Koffi", "Kone", "Traore", "Gueu", "Cisse", "Yao"  );
foreach( $apprenants as $valeur) {
   echo $ valeur.'<br>';
}
?>
```

Dans les versions modernes de PHP (5.5 et plus récentes), on peut déclarer un tableau directement dans le foreach.

Pour aller plus loin dans la syntaxe, nous allons afficher l'ensemble des carrés et des cubes de 1 à 10 à partir du tableau [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], de l'opérateur "puissance" ****** et de la fonction **echo** :

```
foreach ([1, 2, 3, 4, 5, 6, 7, 8, 9, 10] as $valeur) {
    echo $valeur . ' a pour carré ' .
        ($valeur**2) . ' et pour cube ' .
        ($valeur**3) . ".\n";
}
```

Comme elle fait partie du langage PHP, la structure foreach peut être imbriquée dans une autre structure, par exemple un autre foreach.
Pour afficher un à un les éléments d'un tableau à deux dimensions, il est donc possible de faire comme ceci :

```
$tableau_3 = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9],
];
foreach ($tableau_3 as $cle_1 => $valeur_1) {
    foreach ($valeur_1 as $cle_2 => $valeur_2) {
        echo '[' . $cle_1 . ', ' . $cle_2 . '] = ' . $valeur_2 .
"\n";
    }
    echo "----------\n";
}
```

Les fonctions

Les fonctions permettent d'éviter d'avoir à répéter du code PHP que l'on utilise souvent. Une fonction est une série d'instructions qui exécute des actions et qui retourne une valeur.
Une fonction s'appelle par la syntaxe **< ?php nomFonction($variable) ?>** et se récupère par **< ?php $res = nomFonction($variable) ?>**
Il existe des fonctions propres a php, nous avons vu **isset**, il y'a aussi **htmlspecialchar, trim, date,**

```
<?
function saluer($qui, $texte = 'Bonjour')
{

 if(empty($qui)){ // $qui est vide, on retourne faux
  return false;

 }else{
  echo "$texte $qui"; // on affiche le texte
  return true; // fonction exécutée avec succès
 }
}
?>
```

Les fonctions seront vues de manière plus approfondies lors de l'utilisation de la POO dans le php

Les variables super globales

Les variables super globales sont des variables un peu particulières générées automatiquement par PHP. leur nom est écrit en majuscules et commence par un underscore.

- Elles sont écrites en majuscules et commencent toutes, à une exception près, par un underscore (_). $_GET et $_POST en sont des exemple;
- Les superglobales sont des array car elles contiennent généralement de nombreuses informations ;
- Enfin, ces variables sont automatiquement créées par PHP à chaque fois qu'une page est chargée. Elles existent donc sur toutes les pages et sont accessibles partout : au milieu de votre code, au début, dans les fonctions, etc.

Les sessions

Les sessions constituent un moyen de conserver des variables sur toutes les pages d'un site le temps de connexion de l'utilisateur. Son fonctionnement est tres simple :

- Un visiteur arrive sur votre site. On demande à créer une session pour lui. **session_start()** démarre le système de sessions.
- Une fois la session générée, on peut créer une infinité de variables de session pour nos besoins. Par exemple, on peut créer une variable **$_SESSION['nom']** qui contient le nom du visiteur**, $_SESSION['prenom']** qui contient le prénom, etc.
- Lorsque le visiteur se déconnecte de votre site, la session est fermée et PHP « oublie » alors toutes les variables de session que vous avez créées. Vue qu'il n'est pas facile de savoir quand l'utilisateur se déconnecte, on fixera un délai d'inactivité pour le déconnecter automatiquement : on parle alors de **timeout**. session_destroy() : ferme la session du visiteur. Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page de votre site pendant plusieurs minutes (c'est le timeout), mais vous pouvez aussi créer une page « Déconnexion » si le visiteur souhaite se déconnecter manuellement.
-

NB : il faut appeler session_start() sur chacune des pages avant d'écrire le moindre code HTML (avant même la balise < !DOCTYPE>). Si vous oubliez de lancer session_start(), vous ne pourrez pas accéder aux variables superglobales $_SESSION.

Les cookies

Un cookie, c'est un petit fichier que l'on enregistre sur l'ordinateur du visiteur. Ce fichier contient du texte et permet de « retenir » des informations sur le visiteur. Les cookies sont donc des informations temporaires que l'on stocke sur l'ordinateur des visiteurs. Un cookie a un nom, une valeur, une date d'expiration

*<?php setcookie('pseudo', 'abCd', time() + 365*24*3600); ?>*

Comme pour session_start, cette fonction ne marche que si vous l'appelez avant tout code HTML.

Il est recommandé d'activer l'option httpOnly sur le cookie. Cela rendra votre cookie inaccessible en JavaScript sur tous les navigateurs qui supportent cette option. Cette option permet de réduire drastiquement les risques de faille XSS sur votre site, au cas où vous auriez oublié d'utiliser **htmlspecialchars**.

PhP Oriente Objet

L'objectif de la **programmation orientée objet** est de se concentrer sur l'**objet** lui-même et les données, plutôt que sur la logique nécessaire et les actions à mener pour faire cette manipulation.

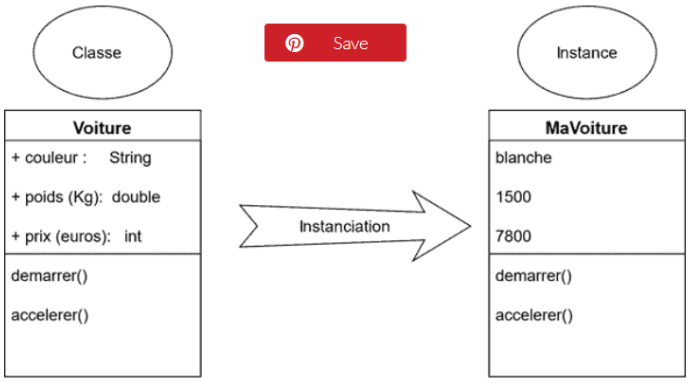
La programmation orientée objet repose sur 5 concepts fondamentaux à savoir

La classe

Une classe est une structure abstraite qui décrit des objets du monde réel sous deux angles : ses propriétés (ses caractéristiques) et ses méthodes (les actions qu'elle peut effectuer ou son comportement).

L'instanciation d'une classe fait appel à 3 méthodes spéciales dont la compréhension est très importante :

- **Le constructeur :** on distingue trois types de constructeurs le constructeur par défaut, le constructeur par recopie (ou constructeur de copie), le constructeur paramétrique
- **Les accesseurs (get) et les mutateurs (set) :** ces méthodes spéciales permettent d'appeler les propriétés et modifier les propriétés d'une classe depuis l'extérieur, un peu comme une API. C'est grâce à elles que l'extérieur peut « appeler » les fonctionnalités de la classe. Les accesseurs permettent de récupérer la valeur des propriétés d'une instance de classe depuis l'extérieur sans y accéder directement. Ce faisant, ils sécurisent l'attribut en restreignant sa modification. Les mutateurs quant à eux permettent de modifier la valeur des propriétés tout en vérifiant que la valeur que l'on veut donner à l'attribut respecte les contraintes sémantiques qui ont été imposées à la classe.
- **Le destructeur** : est une méthode qui met fin à la vie d'une instance de classe. Il peut être appelé à la suppression de l'objet, explicitement ou implicitement.



```
Public class Voiture{
        //Déclaration des attributs
        private String couleur;
        private double poids;
        private int prix;
        //Déclaration des méthodes
        void demarrer(){...}
        void accelerer(){...
    // accesseur sur couleur
        public String getCouleur(){
                return this.couleur;
        }
        //mutateur sur couleur
        public void setCouleur(String uneCouleur){
                this.couleur = uneCouleur;
        }
}
```

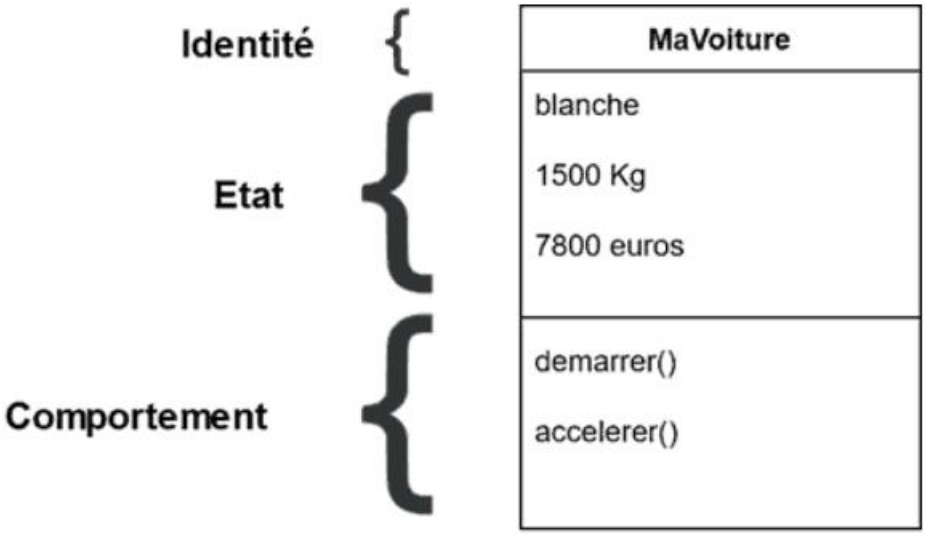
Toutes les instances de classe s'appellent des objets. Les objets sont construits à partir de la classe, par un processus appelé instanciation. De ce fait, tout objet est une instance de classe.

Les objets

Un objet est une instance de classe. un objet est caractérisé par 3 choses :

- **Une identité** : l'identité doit permettre d'identifier sans ambiguïté l'objet (adresse/ référence ou nom)

- **Des états** : chaque objet a une valeur par défaut (lorsqu'elle est indiqué à l'instanciation) pour chacune de ses propriétés. On appelle ces valeurs, des états de l'objet.
- **Des méthodes** : chaque objet est capable d'exécuter les actions ou le comportement défini dans la classe. Ces actions sont traduites en POO concrètement sous forme de méthodes. Les actions possibles sur un objet sont déclenchées par des appels de ces méthodes ou par des messages envoyés par d'autres objets.



L'encapsulation

Les propriétés des objets ne peuvent être accédées que par ses méthodes. Ainsi, la classe encapsule à la fois les attributs et les méthodes qui permettent de manipuler les objets indépendamment de leurs états.

L'encapsulation permet de restreindre l'accès direct aux états et empêche la modification de l'objet hors de ses méthodes.

L'encapsulation est un mécanisme qui empêche donc de modifier ou d'accéder aux objets par un autre moyen que les méthodes proposées, et de ce fait, permet de garantir l'intégrité des objets.

L'utilisateur d'une classe n'a pas forcément à savoir de quelle façon sont structurées les méthodes dans l'objet, ni le processus par lequel l'objet obtient tel ou tel état. En interdisant l'utilisateur de modifier directement les attributs, et en l'obligeant à utiliser

les fonctions définies pour les modifier, on est capable de s'assurer de l'intégrité des objets. On pourra donc effectuer des contrôles sémantiques sur les données entrées par l'utilisateur.

L'encapsulation permet de définir des niveaux de visibilité des éléments de la classe. Ces niveaux de visibilité définissent ce qu'on appelle la portée (ou encore le périmètre) de la l'attribut/méthode. La portée est ainsi définie par méthode et par attribut et indique les droits à leur accès. Il existe trois niveaux de visibilité :

Publique (+): les attributs publics sont accessibles à tous

Protégée (#): les attributs protégés sont accessibles seulement dans la classe elle-même et aux classes dérivées.

Privée (-): les attributs privés sont accessibles seulement par la classe elle-même.

L'héritage

C'est un concept en POO qui désigne le fait qu'une classe peut hériter des caractéristiques (attributs et méthodes) d'une autre classe.

Les objets de classes peuvent hériter des propriétés d'une classe parent. Par exemple on peut définir une classe *Employé* et une classe *Manager* qui est une classe spécialisée de employé, qui hérite de ses propriétés.

Soient deux classes A et B

- Relation d'héritage : Classe B « étend la » Classe A
- A est la super-classe ou classe mère/parent de B
- B est la sous-classe ou classe fille de A

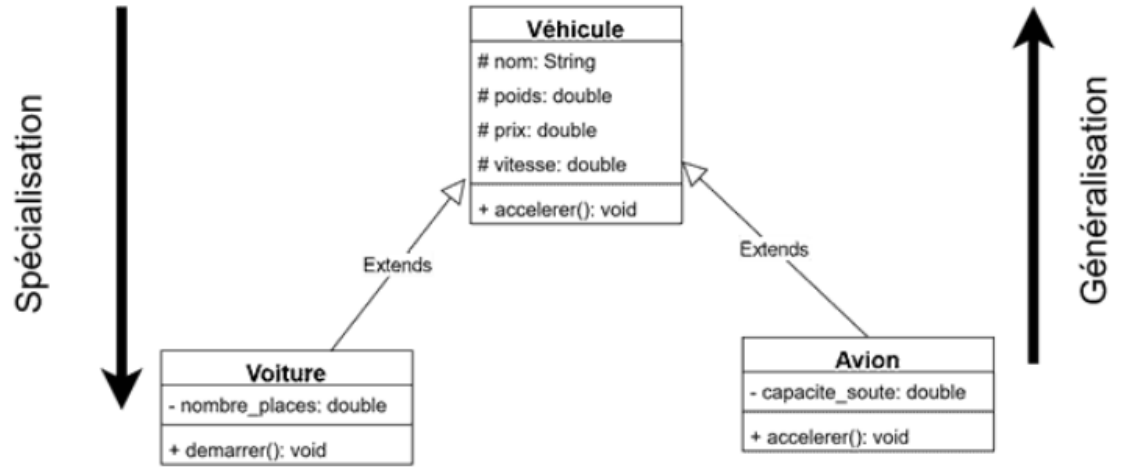


Figure : l'héritage permet de créer des classes spécialisées.

L'héritage présente 2 avantages principaux en POO :

- ***la spécialisation :*** une nouvelle classe réutilise les attributs et les méthodes d'une classe en y ajoutant des opérations particulières à la nouvelle classe.
- ***la réutilisation :*** pour chaque classe spécialisé, on n'a pas besoin de recréer à chaque fois la même classe.

Le polymorphisme

Le polymorphisme décrit un modèle dans la programmation orientée objet dans lequel les classes ont des fonctionnalités différentes, tout en partageant une interface commune.

Dans le monde de la programmation, le polymorphisme est utilisé pour faire des applications plus modulaire et extensible. Au lieu de déclarations conditionnelles (qui font désordre) pour décrire les différentes actions possibles, vous créez des objets interchangeables que vous sélectionnez en fonction de vos besoins. Tel est l'objectif de base du polymorphisme.

INSTRUCTION DU TP

1- Concevoir un site web en respectant la structure du site a la page 27 du cours :
   - Pour le header on met le menu à gauche et le logo à droite
   LE CORPS DU SITE Pour le nav on met aussi le menu
   - Pour la 1ere section :
   - Article : on écrit un article sur notre filière
   - 1er aside : on met une photo de l'auteur de l'article, notre photo
   - 2em aside : info sur l'auteur
   - Pour la 2em section: on écrit sur la programmation génie logiciel
   aside: on met une photo sur le genie logiciel
   LE PIED DE PAGE : écrire complètement à droite
   copyright@toutdroitréserve

2- creer la base de donnée MiageGi et créer une table Etudiant: contenant nom, prénom , age , classe

- ensuite créer la connexion php

- écrire du php pour faire une connexion a la base de donnée

Résolution

- Création de la base de donnée
- Création de la table étudiant et de la table utilisateurs

```
CREATE TABLE `users` (
`id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
`username` varchar(100) NOT NULL,
`email` varchar(100) NOT NULL,
`password` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- Créer la page db_config tel que vue dans le cours

```
<?php
try{
    $bdd = new PDO('mysql:host=localhost;dbname = l3mathg4','root','');
}
catch(exception $e){
    die('Erreur:'.$e->getMessage());
}
?>
```

- Créer la page ajouter un utilisateur :

```
<?php
require('config.php');
if (isset($_REQUEST['username'], $_REQUEST['email'], $_REQUEST['password'])){
// récupérer le nom d'utilisateur et supprimer les antislashes ajoutés par le formulaire
$username = stripslashes($_REQUEST['username']);
$username = mysqli_real_escape_string($bdd, $username);
// récupérer l'email et supprimer les antislashes ajoutés par le formulaire
$email = stripslashes($_REQUEST['email']);
$email = mysqli_real_escape_string($bdd, $email);
// récupérer le mot de passe et supprimer les antislashes ajoutés par le formulaire
$password = stripslashes($_REQUEST['password']);
$password = mysqli_real_escape_string($conn, $password);
//requéte SQL + mot de passe crypté
$query = "INSERT into `users` (username, email, password)
VALUES ('$username', '$email', '".hash('sha256', $password)."')";
// Exécuter la requête sur la base de données
$res = mysqli_query($bdd, $query);
if($res){
echo "<div class='sucess'>
```

```
<h3>Vous êtes inscrit avec succès.</h3>
<p>Cliquez ici pour vous <a href='login.php'>connecter</a></p>
</div>";
}
}else{
?>
<form class="box" action="" method="post">
<h1 class="box-title">S'inscrire</h1>
<input type="text" class="box-input" name="username" placeholder="Nom d'utilisateur"
required />
<input type="text" class="box-input" name="email" placeholder="Email" required />
<input type="password" class="box-input" name="password" placeholder="Mot de passe"
required />
<input type="submit" name="submit" value="S'inscrire" class="box-button" />
<p class="box-register">Déjà inscrit? <a href="login.php">Connectez-vous ici</a></p>
</form>
<?php } ?>
</body>
</html>
```

- Creer la page de connexion (login.php)

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="style.css" />
</head>
<body>
<?php
require('db_config.php');
session_start();
```

```php
if (isset($_POST['username'])){
$username = stripslashes($_REQUEST['username']);
$username = mysqli_real_escape_string($bdd, $username);
$password = stripslashes($_REQUEST['password']);
$password = mysqli_real_escape_string($bdd, $password);
$query = "SELECT * FROM `users` WHERE username='$username' and
password='".hash('sha256', $password)."'";
$result = mysqli_query($bdd,$query) or die(mysql_error());
$rows = mysqli_num_rows($result);
if($rows==1){
$_SESSION['username'] = $username;
header("Location: index.php");
}else{
$message = "Le nom d'utilisateur ou le mot de passe est incorrect.";
}
}
?>
<form class="box" action="" method="post" name="login">
<h1 class="box-title">Connexion</h1>
<input type="text" class="box-input" name="username" placeholder="Nom d'utilisateur">
<input type="password" class="box-input" name="password" placeholder="Mot de passe">
<input type="submit" value="Connexion " name="submit" class="box-button">
<p class="box-register">Vous êtes nouveau ici? <a href="register.php">S'inscrire</a></p>
<?php if (! empty($message)) { ?>
<p class="errorMessage"><?php echo $message; ?></p>
<?php } ?>
</form>
</body>
</html>
```

- Afficher la liste des étudiants :

```
< ?php
Require('db_config.php'>
$etudiant = $PDO->query('SELECT * FROM etudiant ;')->fetchAll() ;
?>
```

Creer un tableau ensuite dans la balise

```
<tbody>
  < ?php foreach($etudiant as $etudiant) : ?>
   <tr>
        <td><?= $etudiant['matricule']  ?></td>
        <td><?= $etudiant['nom]  ?></td>
        <td><?= $etudiant['prénom]  ?></td>
       <td><?= $etudiant['age]  ?></td>
       <td><?= $etudiant['classes]  ?></td>
  </tr>
< ?php endforeach?>
< /tbody>
```

[i] https://datatracker.ietf.org/doc/html/rfc2616