

# Développement Web : "Zone Grand Débutant"

par Guillaume Rossolini ([Tutoriels Web](#) / [SEO](#) / [PHP](#)) ([Blog](#))

Date de publication : 2007

Dernière mise à jour : 16 janvier 2010

Si vous n'avez jamais (ou très peu) fait de pages Web, ce tutoriel est pour vous.

Nous allons vous présenter rapidement les concepts liés au thème du Web et nous vous proposerons de la lecture pour approfondir vos connaissances. Notre objectif ici est de vous guider pas à pas afin de vous permettre de créer rapidement vos pages Web : nous aborderons progressivement HTML, CSS, PHP, JavaScript et les bases de données.

Si vous suivez uniquement ce tutoriel, vous aurez une vision d'ensemble mais incomplète. Pour bien faire, consultez également les tutoriels que je vous recommande. Chaque page contient un paragraphe "formation" ayant pour finalité de sélectionner nos cours les plus didactiques.

I - Introduction.....	4
I-A - Remerciements.....	4
I-B - Préambule.....	4
I-C - Analogies.....	4
I-C-1 - Tableur (pour feuilles de calcul).....	4
I-C-2 - Traitement de texte.....	5
I-C-3 - Conclusion.....	7
II - HTML : Structurer le document.....	7
II-A - Introduction.....	7
II-A-1 - Vue d'ensemble.....	7
II-A-2 - Organisation d'un fichier HTML.....	8
II-B - Terminologie.....	9
II-B-1 - Balise.....	9
II-B-2 - Attribut.....	10
II-C - Outils nécessaires.....	10
II-D - Votre premier document HTML.....	11
II-E - Formation.....	12
II-E-1 - Organisme de référence en HTML.....	12
II-E-2 - Nos ressources habituelles.....	12
II-E-3 - Notre sélection de cours :.....	12
Cours sur les formulaires en HTML.....	12
Div et CSS : une mise en page rapide et facile.....	12
Écrire du bon code HTML.....	13
Introduction à l'XHTML.....	13
II-F - Les limites du HTML.....	13
III - CSS : Présentation du document.....	13
III-A - Introduction.....	13
III-B - Outils nécessaires.....	13
III-C - Votre premier style.....	14
III-D - Formation.....	14
III-D-1 - Organisme de référence.....	14
III-D-2 - Nos ressources habituelles.....	14
III-D-3 - Notre sélection de cours.....	14
CSS : Notions de base.....	14
Div et CSS : une mise en page rapide et facile.....	14
Écrire du bon code CSS.....	15
IV - JavaScript : Dynamisme du document.....	15
IV-A - Introduction.....	15
IV-B - Outils nécessaires.....	15
IV-C - Vos premiers scripts JS.....	15
IV-D - Formation.....	17
IV-D-1 - Organisme officiel.....	17
IV-D-2 - Nos ressources habituelles.....	17
IV-D-3 - Notre sélection de cours.....	17
Introduction à JavaScript.....	17
JavaScript dans les formulaires.....	17
Conseils d'écriture du code.....	17
Comparatif PHP/Javascript.....	17
IV-E - Les dangers de JavaScript.....	17
V - PHP : Dynamisme du contenu.....	18
V-A - Introduction.....	18
V-B - Analogie.....	18
V-C - Outils nécessaires.....	19
V-D - Vos premiers scripts PHP.....	19
V-D-1 - Introduction.....	19
V-D-2 - Découvrir les variables.....	19
V-D-3 - Découvrir les inclusions.....	22
V-E - Avertissement : ce que PHP n'est pas.....	23

V-F - Formation.....	23
V-F-1 - Organisme officiel.....	23
V-F-2 - Nos ressources habituelles.....	23
V-F-3 - Notre sélection de cours.....	24
Comparatif PHP/Javascript.....	24
Guide de style.....	24
Les formulaires et PHP5.....	24
Les sessions PHP.....	24
V-F-4 - Nos tests d'EDI.....	24
PHPEclipse : Programmez librement pour le Web.....	24
PHPEdit, un IDE complet pour PHP.....	24
Test complet de l'éditeur WebExpert de Visicom.....	24
VI - Base de données : Stockage des informations.....	25
VI-A - Introduction.....	25
VI-B - Outils nécessaires.....	25
VI-C - Votre première BDD.....	26
VI-C-1 - L'analyse (conception de la structure).....	26
Modèle Conceptuel des Données (MCD).....	26
Modèle Logique des Données (MLD).....	26
Modèle Physique des Données (MPD).....	27
VI-C-2 - Le SQL (création et mise à jour de la structure, consultation et mise à jour des informations).....	27
VI-C-3 - Exemples concrets.....	29
XML.....	29
MS Office Excel.....	31
MS Office Access.....	34
MySQL.....	36
VI-D - Formation.....	41
VI-D-1 - Organisme de référence.....	41
VI-D-2 - Nos ressources habituelles.....	41
VI-D-3 - Notre sélection de cours.....	41
Conception d'une base de données.....	41
Quelques notions de manipulation de données : SQL.....	41
Utiliser une BDD avec PHP.....	42
Écrire du bon code SQL.....	42
VII - Conclusion générale.....	42
VII-A - Épilogue.....	42
VII-B - Pour aller plus loin.....	42
VII-C - Contribuez.....	42

## I - Introduction

### I-A - Remerciements

Mes plus sincères remerciements vont à toute l'équipe Web/PHP, en particulier aux rédacteurs des tutoriels et des tests dont je parle ici.

### I-B - Préambule

Lorsque l'on souhaite créer des pages Web, il y a de très nombreuses manières d'arriver au but. Nous allons voir ici comment utiliser les technologies de manière simple et efficace.

Il s'agit ici d'un cours pour débutants. Nous n'aborderons pas de techniques complexes dans les tutoriels.

Cette introduction peut sembler courte mais je préfère y aller pas à pas : je vous donnerai une introduction spécifique dans chaque partie.

### I-C - Analogies

Une page Web est un *document*. Elle est enregistrée sur le disque du serveur, et affichée par le navigateur Web. Le navigateur Web affiche le contenu du fichier enregistré sur le serveur après l'avoir interprété, il offre donc une interprétation du fichier. J'aimerais mettre ici en avant une terminologie que j'affectionne : le *fichier* est sur le disque dur, tandis que le *document* est affiché par le navigateur. Les termes "fichier" et "document" font référence au même élément, mais ils ne désignent pas la même vision de cet élément. Voyons deux exemples qui nous sont familiers : un tableur et un traitement de texte.

#### I-C-1 - Tableur (pour feuilles de calcul)

Peut-être connaissez-vous le format CSV très souvent utilisé avec les tableurs comme Excel ? Il s'agit de fichiers de texte contenant des données respectant un format particulier : elles sont séparées par des points virgules ou par des tabulations. Un outil comme Microsoft Office Excel ou bien OpenOffice.org Calc vous permettent d'ouvrir un fichier CSV en découpant les lignes en cases.

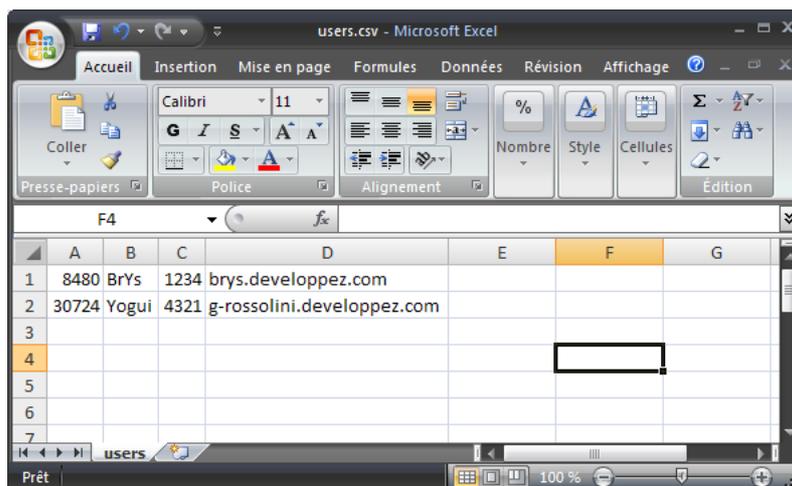
#### Dans ce fichier CSV, les règles sont :

- Les **colonnes** sont séparées par des point virgules ;
- Les **lignes** sont délimitées par des sauts de ligne.

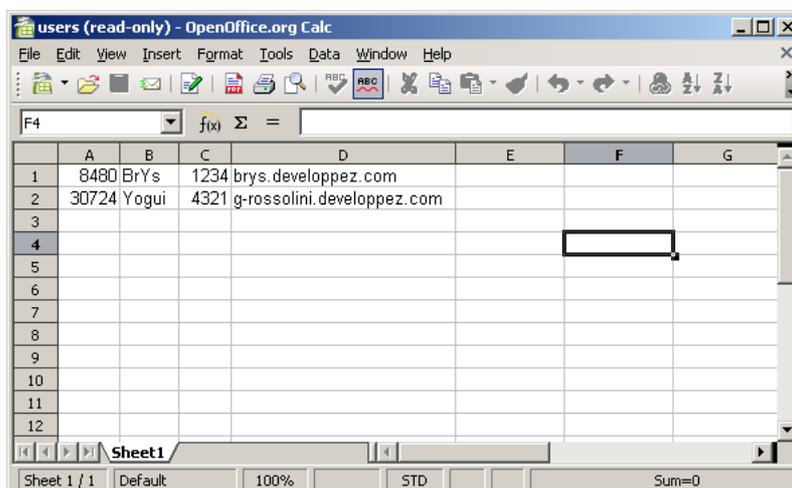
Aperçu dans Notepad :



Aperçu dans MS Office Excel :



Aperçu dans OpenOffice Calc :



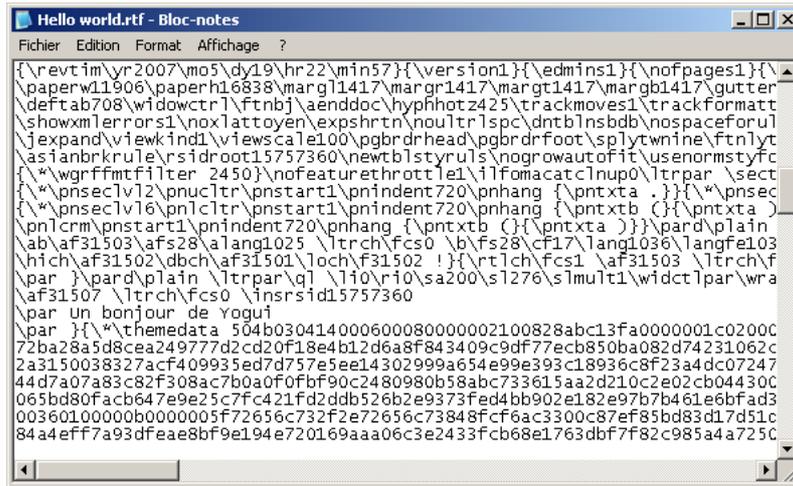
Nous pouvons remarquer un détail intéressant : Notepad, qui n'est pas prévu pour comprendre la syntaxe (le format) des documents CSV, nous affiche le contenu brut du fichier. En revanche, Excel et Calc connaissent la syntaxe CSV et permettent de voir le *document* en tant que tel plutôt que le code source du *fichier*.

## I-C-2 - Traitement de texte

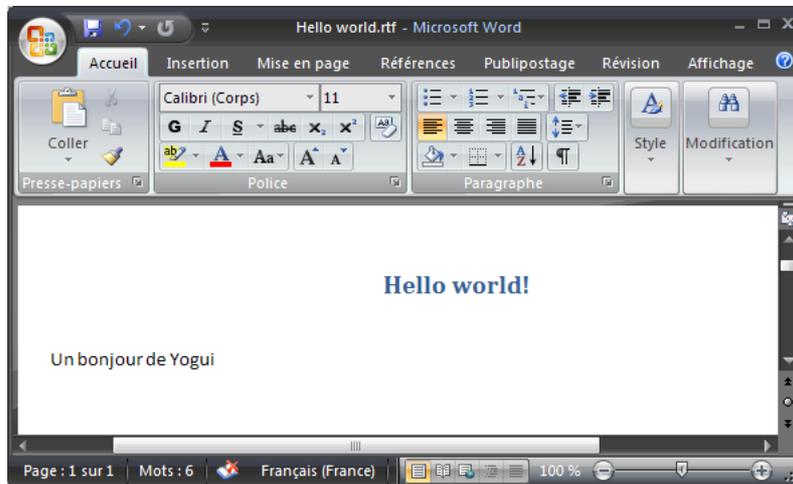
Prenons maintenant un fichier .rtf (Rich Text File), un format de document texte formaté compatible avec tous les processeurs de texte.

Le format RTF étant bien plus complexe que le format CSV, je ne vais pas le détailler ici.

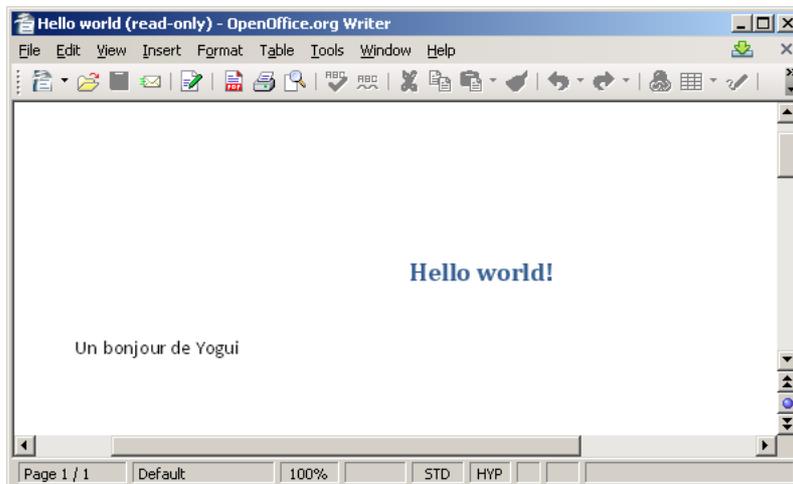
Aperçu dans Notepad :



Aperçu dans MS Office Word :



Aperçu dans OpenOffice Writer :



La conclusion est ici encore plus évidente que pour le format CSV : Notepad affiche le contenu du *fichier*, tandis que Word et Writer affichent le *document*. De plus, RTF étant un format standard, la mise en forme est la même dans

les deux outils de traitement de texte. Ici, le titre est centré dans la page, coloré en bleu et d'une taille plus grosse que le texte ; le texte normal est en noir, aligné à gauche.

 **Retenez cette notion importante** : le respect des standards permet d'afficher le même document de la même manière, quel que soit l'outil utilisé pour le consulter. Je terminerai donc chaque partie de ce tutoriel par un lien vers l'organisme de référence ayant défini le format de fichier de la technologie.

## I-C-3 - Conclusion

De très nombreux *documents* sont enregistrés en tant que *fichiers* et selon un *format* bien défini. Les fichiers ne sont pas toujours lisibles dans leur format brut, c'est pourquoi nous utilisons des outils comme Microsoft Office ou OpenOffice.org pour nous faciliter la tâche.

Le développement Web est fondamentalement le même principe : proposer à l'internaute des documents qui sont en fait des fichiers au format HTML. Le navigateur Web (Internet Explorer, Firefox, Opera...) est l'outil qui sert à l'internaute pour visualiser ces documents, car le code source des documents n'est pas prévu pour être lu par un humain. Le code source des documents est prévu simplement pour définir ce que contient le document et comment ce contenu doit être affiché dans le navigateur. Nous verrons que, tout comme pour les formats CSV et RTF il existe des standards qu'il faut respecter pour que tous les navigateurs affichent vos documents de la même manière.

### Trois notions fondamentales :

- Le **format** décrit la syntaxe du code source (= le fichier) ;
- Le **document** est le fichier affiché (= traduit) et lisible par l'utilisateur ;
- L'**outil** permet d'interpréter le fichier en fonction de son format afin de l'afficher à l'utilisateur.

## II - HTML : Structurer le document

### II-A - Introduction

#### Les trois notions fondamentales :

- Le **fichier** est au format HTML ;
- Le **document** est à destination de l'internaute ;
- L'**outil** est un navigateur Web.

#### II-A-1 - Vue d'ensemble

Le  **HTML** (HyperText Markup Language) est le "langage" qui permet à un navigateur Web d'afficher un document Web selon une certaine structure. Il ne s'agit pas d'un langage à proprement parler (dans le sens "interprété" ou "compilé") mais plutôt d'un format de document.

Une page Web est généralement écrite dans un fichier en HTML (ou assimilé). Ce fichier est en fait du texte, formaté en suivant certaines règles. Ainsi, lorsqu'un fichier HTML est ouvert avec Notepad, on voit du code HTML pur ; en revanche, lorsque l'on utilise un navigateur Web, on voit au contraire un document correctement présenté : le navigateur Web *interprète* le code HTML pour *présenter* le document Web à l'internaute.

#### Exemple de code source HTML :

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

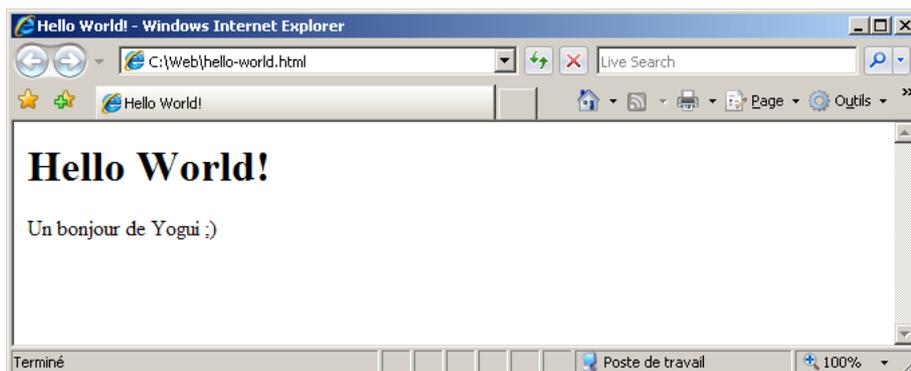
**Exemple de code source HTML :**

```

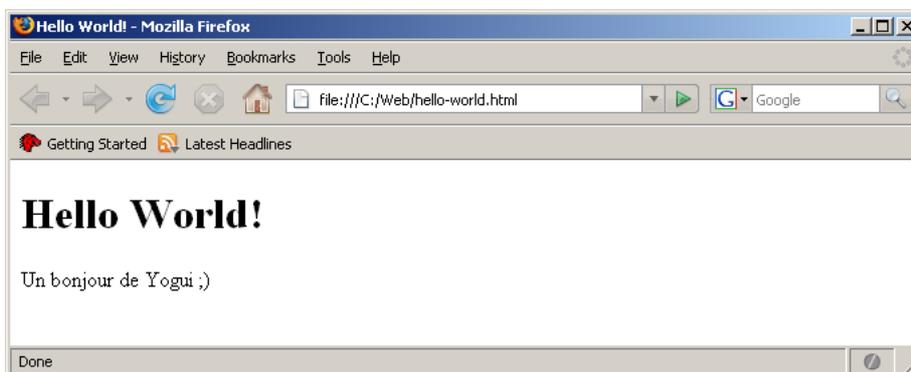
<title>Hello World!</title>
</head>
<body>
  <h1>Hello World!</h1>
  <p>Un bonjour de Yogui ;) </p>
</body>
</html>

```

Aperçu dans Internet Explorer 7 :



Aperçu dans Firefox 2 :



L'on peut remarquer ici la même chose qu'avec mes exemples précédents : le format HTML de mon fichier permet d'afficher le document de la même manière dans les deux navigateurs. C'est rendu possible par le langage HTML, un standard au même titre que RTF.

**II-A-2 - Organisation d'un fichier HTML**

Un fichier HTML contient principalement deux parties : l'en-tête et le corps.

```

<html>
  <head></head>
  <body></body>
</html>

```

L'*en-tête* (head) est une succession de balises (tags) permettant au navigateur Web de savoir de quoi il s'agit. On peut y trouver un résumé de la page, les mots clefs principaux, le titre pour la fenêtre, la fréquence de mise à jour, etc.

Le *corps* (body) est constitué du contenu du document. On y trouve habituellement des divisions (div), des paragraphes (p), des images (img), etc.

Le HTML est un format structuré. Chaque balise peut en contenir d'autres, de manière hiérarchique. Dans le document vu plus haut, l'en-tête contient un titre principal tandis que le corps contient un titre de niveau 1 et un paragraphe.

Il est primordial de comprendre que chaque balise a une signification particulière. Utiliser les bonnes balises au bon moment permet d'obtenir des documents affichés sans surprises dans tous les navigateurs du marché. En revanche, utiliser incorrectement les balises peut donner des résultats imprévisibles... Reportez-vous toujours à la documentation officielle pour savoir comment utiliser les balises ; les tutoriels sont là pour vous aider à comprendre la documentation officielle (qui est malheureusement souvent indigeste).

Lorsque vous souhaitez ajouter un élément à votre document, pensez à sa signification. Chaque type de contenu dispose d'une balise en HTML : si c'est du texte, il s'agit probablement d'un paragraphe ; si vous commencez un chapitre composé d'un titre et de paragraphes, alors vous êtes en train de diviser votre document ; un titre est de niveau 1, 2, 3, etc. suivant sa hiérarchie. Je vous recommande de construire votre document de la même manière que vous construisez un document avec un traitement de texte : pensez au type de contenu que vous souhaitez mettre et utilisez les balises en fonction de cela.

## II-B - Terminologie

### II-B-1 - Balise

En HTML, les informations sont organisées à l'aide de "balises". Une balise permet de contenir quelque chose, par exemple "un lien", "un paragraphe", etc. La balise se caractérise par un mot encadré des signes inférieur < et supérieur > comme par exemple <b>, <body>, etc. Les balises HTML vont généralement par paire : l'une ouvre, l'autre referme et ce qui est situé entre les deux balises est affecté. Toutes les balises de la page sont organisées de manière hiérarchique.

Il faut penser à cette hiérarchie car c'est très important. Par exemple, il est fondamental de refermer les balises dans l'ordre inverse de leur ouverture. La première balise ouverte englobe toutes les autres, elle est la dernière à être refermée. La dernière balise ouverte a une durée de vie plus courte car elle est refermée avant toutes les autres.

La balise ouvrante est la balise permettant de délimiter le début, et la balise fermante délimite la fin. Dans le cas d'un titre, "<h1>" est une balise ouvrante et "</h1>" est la balise fermante correspondante.

#### Ordre de fermeture non respecté :

```
<b><i>du texte en gras italique...</b></i>
```

#### Ordre correct :

```
<b><i>du texte en gras italique...</i></b>
```

Une balise HTML est très simple : le signe "inférieur" < suivi de quelques lettres et enfin du signe "supérieur" >. Si la balise est double (c'est souvent le cas) car elle doit contenir quelque chose, alors la balise fermante est très similaire à la balise ouvrante : < suivi d'une barre oblique / puis des quelques lettres (les mêmes que pour la balise ouvrante) et enfin >.

#### Le corps du document est une balise double :

```
<body>
...
...
...
</body>
```

Un titre est une balise double :

```
<h1>...</h1>
```

 Les balises ayant du contenu doivent être hiérarchisées car elles vont par paires. En revanche, les balises simples n'ont pas cette contrainte.

Un passage à la ligne est une balise simple (sans contenu) :

```
<br/>
```

Une ligne de séparation est une balise simple (sans contenu) :

```
<hr/>
```

 L'identifiant d'une balise ("body" ou bien "h1" par exemple) peut être écrit en minuscules, majuscules ou un mélange des deux mais il est fortement recommandé d'utiliser systématiquement les minuscules.

## II-B-2 - Attribut

En grammaire, le sens d'un *nom commun* peut être précisé par des *compléments*. En HTML, une *balise* peut être complétée par des *attributs*. Une "balise" correspond à un "nom commun" dont le sens peut être précisé par des "attributs" (ses "compléments").

Toutes les balises HTML peuvent contenir un ou plusieurs attributs. Ce sont comme des paramètres, des éléments qui nous permettent de préciser le comportement des balises. Par exemple, la balise "lien" peut être précisée au moyen des attributs "destination" et "titre".

Certaines balises ont des attributs obligatoires. Certains attributs sont spécifiques à certaines balises, tandis que d'autres peuvent être mis dans n'importe quelle balise HTML.

**Simplement à titre d'exemple, voici quelques attributs que nous verrons très souvent :**

- **href** : Principalement dans les liens ;
- **title** : Principalement dans les liens ;
- **src** : Principalement dans les images ;
- **id** : Pour n'importe quelle balise, il est utile pour CSS et JavaScript (cf. plus loin) ;
- **name** : Dans les contrôles des formulaires.

Un lien appelé 'Description' et renvoyant vers 'page.html' :

```
<a href="page.html" title="Description">lien</a>
```

Une image appelée 'Nom du logo' ayant comme source 'images/logo.jpg' :

```

```

 La valeur d'un attribut s'écrit entre guillemets ou bien entre apostrophes. Je vous recommande les guillemets car cela pose moins de problèmes.

## II-C - Outils nécessaires

Tout comme pour les fichiers CSV que j'évoquais plus haut, n'importe quel éditeur de texte peut être utilisé pour créer des pages en HTML. Notepad est suffisant si vous êtes sous Windows mais je vous recommande l'un des outils décrits dans notre page  **Outils (X)HTML**.

Pour visualiser un document HTML, n'importe quel navigateur Web fait l'affaire :  **Internet Explorer**,  **Firefox**,  **Opera**, etc.

## II-D - Votre premier document HTML

Créez le fichier suivant, puis ouvrez-le avec votre navigateur Web (double clic sur le nom du fichier) :

index.html

```
<html>
<head>
  <title>Hello world</title>
</head>
<body>Hello World, I am Yogui!</body>
</html>
```

Le navigateur Web vous affiche une page contenant les mots "Hello World, I am Yogui!" et dont la barre de titre contient simplement "Hello World".

 *Je ne vous présente ici que le strict minimum afin d'obtenir du HTML reconnu à peu près correctement par votre navigateur Web. En situation réelle, il vous faudra respecter les standards tels que le  **W3C** (World Wide Web Consortium) les a définis. Pour faire du véritable code HTML ou XHTML, je vous recommande la lecture de tutoriels plus complets (cf. la liste ci-dessous).*

 *Tout le contenu d'un document HTML se trouve à l'intérieur de la balise **<body>**.*

Ainsi, je vous ai indiqué **la structure** d'une page Web. En effet, le HTML sert principalement à une chose : structurer un document. Bien que ce soit possible, il est préférable d'éviter de mettre en forme le document au moyen de HTML : ce sera le travail de CSS (cf. plus loin).

**Pour faire une page bien présentée, vous pouvez utiliser les balises suivantes :**

- **<center>** : Le texte est centré dans la page (balise double) ;
- **<font color="">** : Changement de la couleur du texte (balise double) ;
- **<font size="">** : Changement de la taille du texte (balise double) ;
- **<b>** ou **<strong>** : Le texte est mis en gras (balise double) ;
- **<i>** ou **<em>** : Le texte est mis en italique (balise double) ;
- **<u>** : Le texte est souligné (balise double) ;
- **<h1>** ou **<h2>** (jusqu'à h6) : Un titre dans la page (balise double) ;
- **<br/>** : Passage à la ligne (balise simple).

 *Ces balises sont à utiliser avec beaucoup de précautions : il s'agit ici de vous enseigner à faire du HTML qui donne un résultat mais, en situation réelle, il convient d'utiliser les technologies correctes (en l'occurrence, beaucoup de CSS). Nous reviendrons par la suite sur tout cela.*

Un élément fondamental à savoir faire en HTML est un **lien hypertexte**. Cela donne au visiteur de votre site la possibilité de consulter une autre page, qu'elle soit sur votre site ou ailleurs.

Des liens :

```
<a href="http://www.site.ext/dossier/page-1.html">Texte affiché dans le lien</a>
<a href="page-2.html">Autre lien</a>
```

Il faut aussi savoir comment insérer **une image** dans votre page Web. Cela se fait au moyen d'une balise simple, en une seule partie (il n'y a pas de **</img>**).

Insérer des images :

```


```

Pour les liens et les images, ce qui est dans l'attribut **href** ou **src** est ce que l'on appelle une URL (ou bien URI). Elle peut être *absolue* (avec le **http://** et l'adresse du site) ou bien *relative*. Je vous recommande d'utiliser des adresses relatives tant que possible.

 *Puisque le HTML permet principalement de définir la structure d'un document (contrairement à la manière dont il est présenté), les navigateurs Web ignorent tous les passages à la ligne ainsi que les espaces (en double) insérés dans le code source. Cela vous permet de présenter votre code source HTML de manière à le rendre lisible, sans que cela ait d'impact sur le document final affiché par le navigateur. Les deux exemples suivants produisent le même affichage :*

#### Exemple 1

```
<html><body>Hello world, I am Yogui!<br/>Ceci est un test.</body></html>
```

#### Exemple 2

```
<html>
  <body>
    Hello world, I am Yogui<br/>
    Ceci est un test.
  </body>
</html>
```

## II-E - Formation

### II-E-1 - Organisme de référence en HTML

L'organisme de référence est le  **World Wide Web Consortium (W3C)**.

### II-E-2 - Nos ressources habituelles

- La  **FAQ (X)HTML** répond à vos questions posées dans nos forums ;
- Les  **tutoriels (X)HTML** vous permettront d'apprendre tout ce dont vous avez besoin ;
- Les  **livres (X)HTML** pour toujours avoir une référence sous le coude ;
- Les forums  **HTML**,  **général conception Web** et  **outils de développement Web** vous permettent de poser toutes vos questions qui ne trouvent pas réponse dans nos autres ressources.

### II-E-3 - Notre sélection de cours :

#### Cours sur les formulaires en HTML

J'ai rédigé  **un cours complet sur les formulaires** accessible à tous les lecteurs. Il vous guidera depuis vos premiers pas jusqu'à leur exploitation. Attention, il vous faut avoir des connaissances en PHP pour exploiter pleinement les formulaires (cf. plus loin), mais vous pouvez dès à présent vous renseigner sur leur fonctionnement.

#### Div et CSS : une mise en page rapide et facile

Pierre-Baptiste Nageon vous propose un cours de  **mise en page au moyen de la balise HTML <div>**. Les plus débutants d'entre vous n'ont pas besoin de s'en soucier, néanmoins il peut être intéressant à lire par tout le monde.

## Écrire du bon code HTML

Voici un tutoriel que je citerai à plusieurs reprises...

Adrien Pellegrini a compilé pour vous  **les bonnes pratiques de développement en HTML**.

## Introduction à l'XHTML

Adrien Pellegrini vous  **présente le XHTML** (une évolution de HTML) en long et en large.

## II-F - Les limites du HTML

Bien qu'il existe des outils très puissants pour créer des pages en HTML, il serait fastidieux de créer manuellement, une à une, toutes les pages d'un catalogue de produits. L'outil informatique existe pour nous faciliter la tâche, en particulier les tâches répétitives. Nous verrons plus loin que le HTML est en fait un objectif : c'est un document que nous allons créer à l'aide d'un langage de script situé sur le serveur. En général, nous ne créons pas intégralement de document HTML sans utiliser un langage de script. Suivant la terminologie que vous préférez, le HTML est habituellement un patron (couture), un moule (cuisine) ou encore un **gabarit** (c'est le terme adéquat).

## III - CSS : Présentation du document

### III-A - Introduction

#### Les trois notions fondamentales :

- Le **fichier** est au format CSS ;
- Le **document** est à destination de l'internaute ;
- L'**outil** est un autre document (une page Web HTML).

Les  **CSS** (Cascading Style Sheet = feuille de styles) sont des documents au format texte, tout comme le HTML. La différence est qu'une CSS n'est prévue ni pour être visualisée par l'internaute ni pour être affichée directement dans le navigateur. Une feuille de styles n'est qu'un document qui définit comment une page HTML doit être affichée. Le navigateur Web s'occupe de charger la CSS sans que l'internaute doive intervenir ; une CSS a pour vocation d'agir en arrière plan.

En utilisant des styles, il est par exemple possible de dire que tous les liens, au lieu d'être bleus ou violets comme on en a l'habitude, doivent être gris ; ou bien que le texte est toujours orange.

En fait, l'idée est que le navigateur parcourt le document HTML. Lorsqu'il rencontre une balise, il demande à la CSS de quelle manière il doit l'afficher. La CSS ne sait rien faire d'autre que dire comment doit être affiché tel ou tel élément de la page Web.

 *Les CSS sont l'un des éléments qui constituent le DHTML : Dynamic HTML.*

### III-B - Outils nécessaires

Pour créer des feuilles de style, vous pouvez utiliser les mêmes outils que pour vos documents HTML. Vous n'avez besoin de rien pour les visualiser puisque ce n'est pas l'objectif : leur utilité est d'être combinées aux documents HTML. Votre navigateur Web sait ce qu'il doit faire des CSS. N'hésitez pas à consulter notre page d'  **Outils CSS**.

## III-C - Votre premier style

Il y a trois manières de définir des styles dans une page Web. La plus simple est d'utiliser l'attribut **style** de la balise que l'on veut modifier.

Exemple :

```
<h1 style="font-style: italic; size: 1.5em;">Titre de la page</h1>
```

À l'intérieur de l'attribut *style*, il est possible de mettre n'importe quelle quantité de **propriétés** CSS : zéro, une, autant que l'on veut. L'ordre dans lequel on met ces propriétés influe rarement, nous n'allons donc pas nous en occuper pour le moment.

 *Un style s'écrit de la manière suivante : identifiant (nom du style) suivi de deux points, de la valeur et d'un point virgule. Dans l'exemple ci-dessus, nous avons les propriétés "font-style" et "size" avec les valeurs (respectivement) "italic" et "1.5em".*

 *L'attribut **style** que je viens d'utiliser peut être ajouté à n'importe quelle balise HTML.*

## III-D - Formation

### III-D-1 - Organisme de référence

L'organisme de référence est le  **World Wide Web Consortium** (W3C).

### III-D-2 - Nos ressources habituelles

- La  **FAQ CSS** répond à vos questions posées dans nos forums ;
- Les  **tutoriels CSS** vous permettront d'apprendre tout ce dont vous avez besoin ;
- Les  **livres CSS** pour toujours avoir une référence sous le coude ;
- Le  **forum CSS** vous permet de poser toutes vos questions qui ne trouvent pas réponse dans nos autres ressources.

### III-D-3 - Notre sélection de cours

#### CSS : Notions de base

Pierre-Baptiste Naigeon vous présente son utilisation des CSS dans les pages Web avec son  **cours d'initiation**.

#### Div et CSS : une mise en page rapide et facile

Pierre-Baptiste a également prévu pour vous un tutoriel vous enseignant à moins utiliser la balise **<table>** :  **utilisez la balise <div> pour structurer vos documents**.

## Écrire du bon code CSS

Adrien Pellegrini a parlé des feuilles de style dans son guide :  **style de programmation**.

## IV - JavaScript : Dynamisme du document

### IV-A - Introduction

#### Les trois notions fondamentales :

- Le **fichier** est au format JavaScript ;
- Le **document** (ou **programme**) est à destination de l'internaute ;
- L'**outil** est un navigateur Web.

Le code  **JavaScript** est interprété par le navigateur Web (s'il n'est pas trop préhistorique et si sa configuration ne l'a pas désactivé).

Je viens déjà de vous présenter un inconvénient de JS : il peut être désactivé. Néanmoins sachez que c'est ce langage qui est derrière l'acronyme  **AJAX**... JavaScript est donc **fondamental** dans le développement des sites Web d'aujourd'hui (de type Web 2.0), à savoir des sites qui ne se contentent pas d'afficher des pages Web mais qui demandent à l'internaute de s'impliquer, de participer, voire des sites qui s'adaptent au profil de leurs visiteurs.

Incruster du code JS dans une page Web permet à la page de réagir aux actions de l'internaute, par exemple "la souris survole tel bouton" ou encore "le formulaire vient d'être envoyé". Ainsi, nous pouvons utiliser JavaScript pour attraper des événements et agir en conséquence.

 *ActionScript 3, la dernière version du langage de script pour Flash, utilise une syntaxe quasiment identique à celle de JavaScript.*

### IV-B - Outils nécessaires

Pour développer en JavaScript, il est recommandé d'avoir un éditeur de code plus puissant que ceux recommandés au début de cet article (avoir la documentation du langage à portée de main commence à devenir une condition importante). Cependant, surtout dans vos débuts, vous pourrez vous contenter de rechercher sur Internet et d'utiliser des codes sources déjà existants, en les adaptant à vos besoins.

En clair, il vous faut un bon éditeur de code + la documentation officielle + une bonne FAQ (*cf.* les liens ci dessous).

Un bon outil de développement JavaScript est  **Aptana**, dans la famille d'Eclipse.

### IV-C - Vos premiers scripts JS

Le code JavaScript doit être appelé depuis un attribut situé dans une balise HTML. Le nom de l'attribut utilisé commence par "on" et se poursuit par l'évènement désiré (exemples : *onload*, *onsubmit*, *onmouseover*, etc.).

Exemple d'action au chargement de la page Web :

```
<html>
<body onload="alert('Hello world!');"></body>
</html>
```

JavaScript est très souvent utilisé pour gérer des formulaires en ligne. Je ne compte pas vous montrer ici une utilisation complète mais j'espère que mes exemples vous donneront une vision d'ensemble.

Envoyer le formulaire uniquement si le nom d'utilisateur n'est pas vide :

```
<html>
<body>
<form onsubmit="if(document.getElementById('login').value == ''){ return false; } else{ return true; }">
    Nom d'utilisateur : <input type="text" id="login"/><br/>
    Mot de passe : <input type="text" id="password"/><br/>
    <input type="submit"/>
</form>
</body>
</html>
```

Oui, bien sûr, ce n'est ni proprement écrit ni facile à lire... Il faut bien voir que nous commençons à voir des techniques un peu plus complexes que HTML et CSS, il faut donc commencer à bien s'organiser.

Envoyer le formulaire uniquement si le nom d'utilisateur n'est pas vide + avertir l'internaute :

```
<html>
<head>
    <script type="text/javascript">
        function checkForm()
        {
            if(document.getElementById("login").value == "")
            {
                alert("Formulaire invalide : le nom d'utilisateur est vide");
                return false;
            }
            else
            {
                alert("Le formulaire est valide");
                return true;
            }
        }
    </script>
</head>
<body>
<form onsubmit="return checkForm();">
    Nom d'utilisateur : <input type="text" id="login"/><br/>
    Mot de passe : <input type="text" id="password"/><br/>
    <input type="submit" value="Envoyer"/>
</form>
</body>
</html>
```

Ici, le comportement de notre page Web est bien plus évident. Lorsque l'évènement *onsubmit* est déclenché par le formulaire (c'est-à-dire que l'internaute a validé le formulaire), la fonction **checkForm()** vérifie si le champ "login" est rempli. Si c'est bon, le formulaire est transmis ; sinon, il n'est pas envoyé.

Cela permet d'éviter d'envoyer des informations au serveur lorsque l'on sait qu'elles sont incomplètes, et ainsi d'éviter à l'internaute d'attendre que le serveur réponde. Ici, c'est le navigateur qui se charge de tout vérifier.

 Ici encore, je vous ai proposé un exemple minimaliste. En situation réelle, il faut notamment utiliser les attributs "id" et "name" des contrôles du formulaire : j'en parle plus en détail dans mon cours sur les formulaires (cf. les liens en bas de page)

### Sécurité

Dans l'exemple ci-dessus, le serveur Web ne doit pas partir du principe que tout formulaire reçu a été validé : ce serait une erreur grave. Le code JavaScript présenté ici sert uniquement à accélérer l'arrivée des réponses données à l'utilisateur ; il **ne permet pas** d'assurer que 100% des formulaires transmis sont valides.

## IV-D - Formation

### IV-D-1 - Organisme officiel

Depuis que la fondation Mozilla a repris le développement du navigateur Netscape, elle a repris également le langage  **JavaScript**.

La fondation Mozilla met à disposition la  **Référence** et le  **Guide** de JavaScript 1.5, deux collections à toujours garder à portée de main.

### IV-D-2 - Nos ressources habituelles

- La  **FAQ JavaScript** et la  **FAQ AJAX** répondent à vos questions posées dans nos forums ;
- Les  **sources JavaScript** vous donnent des codes permettant de résoudre des problématiques précises ;
- Les  **tutoriels JavaScript** vous permettront d'apprendre tout ce dont vous avez besoin ;
- Les  **livres JavaScript** pour toujours avoir une référence sous le coude ;
- Le  **forum JavaScript** et le  **forum AJAX** vous permettent de poser toutes vos questions qui ne trouvent pas réponse dans nos autres ressources.

### IV-D-3 - Notre sélection de cours

#### Introduction à JavaScript

Serge P. vous  **présente le langage JavaScript** de manière très didactique et complète.

#### JavaScript dans les formulaires

Dans mon cours sur les formulaires, j'ai présenté une méthode simple de  **validation de formulaire** au moyen de JavaScript.

#### Conseils d'écriture du code

Adrien Pellegrini a également couvert le langage JavaScript dans son  **guide de style**.

#### Comparatif PHP/Javascript

PHP et JavaScript étant très souvent confondus, Julp vous a préparé  **un tableau comparatif** qui ne vous laissera plus aucun doute sur leurs différences.

## IV-E - Les dangers de JavaScript

Ce paragraphe n'est qu'une reformulation de ce que j'ai déjà évoqué plus haut. Il est absolument fondamental que tout développeur ait parfaitement conscience de tous ces dangers.

Puisque JavaScript permet de vérifier certaines informations envoyées par l'internaute, la tentation est forte de faire confiance à une page Web dont on sait que du code JS effectue ces vérifications. Cependant, il faut bien se souvenir

que la majorité des navigateurs Internet peuvent désactiver le JavaScript, ce qui rend ces précautions inutiles (tout du moins du point de vue sécurité). Par conséquent, JavaScript ne doit être utilisé que pour améliorer l'expérience utilisateur, **pas pour la sécurité**. Si un champ est vérifié par JS, il doit impérativement l'être à nouveau par le script de destination côté serveur.

J'insiste : JavaScript est à utiliser **exclusivement** pour **faciliter la navigation** dans votre site.

## V - PHP : Dynamisme du contenu

### V-A - Introduction

#### Les trois notions fondamentales :

- Le **fichier** est au format PHP ;
- Le **document** est habituellement au format HTML et à destination du navigateur Web (qui se chargera de le transmettre à l'internaute) ;
- L'**outil** est le serveur Web avec l'exécutable PHP.

Vous est-il déjà arrivé de créer toute une suite de documents HTML pour faire une galerie de photos ou un catalogue de produits ? Vous en conviendrez, c'est fastidieux...

Eh bien, la solution existe : un langage de script tel que  **PHP** !

À l'origine, PHP fut créé par Rasmus Lerdorf pour ses besoins personnels : Personal Home Page (version 1). Depuis, le nom fut changé à Hypertext Preprocessor (ce qui présage de son utilité principale : créer dynamiquement des documents HTML). Depuis le 2 novembre 2006, PHP est en version 5.2 ; la branche 5.3 est disponible depuis le 30 juin 2009 ; la version 6 est reportée à une date encore inconnue.

### V-B - Analogie

Lorsque vous êtes dans un bar parisien, vous commandez votre boisson au serveur qui s'occupe d'aller la chercher au comptoir et de vous l'apporter. Vous pouvez alors consommer votre boisson.

Sur Internet, il se passe quelque chose de similaire : lorsque vous êtes dans un site Web, vous cliquez sur des liens pour consulter des documents. Le serveur Web reçoit vos demandes, s'occupe de rechercher le document voulu et vous l'envoie afin que vous puissiez le lire (c'est la consommation).

Dans le cas de scripts PHP, la "recherche" du document s'accompagne de ce que l'on appelle son "interprétation". Dans l'exemple du bar parisien, vous ne demandez pas systématiquement un "jus d'ananas" que le serveur se contente d'ouvrir devant vous : parfois, vous demandez un "café au lait" qui lui demande un peu de préparation. En fait, dans ce cas, ce n'est pas le serveur qui prépare la boisson : il sous traite votre demande à un collègue car cela lui prendrait trop de temps de le faire lui-même, or il a d'autres clients à servir.

Là encore, il se passe la même chose dans l'environnement Web. Lorsque le serveur Web reçoit une demande de document, il sous traite la tâche à PHP afin de pouvoir retourner à ses affaires : c'est PHP (et non le serveur Web) qui s'occupe de construire le document. Une fois qu'il est prêt, le document est envoyé de PHP au serveur Web, puis du serveur Web au navigateur Internet (c'est-à-dire à vous, l'internaute).

## V-C - Outils nécessaires

Afin de programmer en PHP, il est fondamental d'utiliser un véritable  **EDI** tel que  **PHPEdit** ou  **PHPEclipse**. Ces outils vous permettront d'être efficace. Toutefois, de nombreux développeurs continuent à utiliser des éditeurs simples comme Notepad++.

Avec l'expérience, vous verrez ce qui vous convient le mieux...

À part d'un logiciel d'édition de code et de votre navigateur Internet, vous aurez besoin d'un serveur Web.  **Apache** est le plus connu d'entre eux mais il n'est pas toujours facile à installer, c'est pourquoi dans un premier temps je vous recommande  **WAMP Server**,  **XAMPP** (ce dernier ayant le mérite de très bien s'intégrer dans PHPEclipse) ou encore  **Zend Core** (validé/supporté par Zend).

## V-D - Vos premiers scripts PHP

### V-D-1 - Introduction

Les scripts PHP s'intègrent dans la page Web. Lorsque le document est demandé par le navigateur Web, le serveur Web (qui est à l'écoute) entend la demande et fait interpréter le script par PHP, puis il renvoie le résultat au navigateur.

Un script PHP est identifié par une balise particulière : **<?php** ouvre le script et **?>** le referme. Tout ce qui se trouve à l'intérieur de cette balise est du PHP.

Vous devez enregistrer tous vos scripts à l'intérieur de la racine du serveur Web dans votre système de fichiers.

#### Le chemin par défaut dépend de votre serveur Web :

- **Zend Core 2** : C:\Program Files\Zend\Apache2\htdocs\
- **XAMPP** : C:\Program Files\Xampp\htdocs\
- **WAMP Server** : C:\wamp\www\
- **EasyPHP 1-8** : C:\Program Files\EasyPHP1-8\www\
- **Apache 2.2** : C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\
- **Apache 2.0** : C:\Program Files\Apache Group\Apache2\htdocs\
- **Apache 1.3** : C:\Program Files\Apache Group\Apache\htdocs\
- *etc.*

### V-D-2 - Découvrir les variables

Mon premier script PHP : test.php

```
<html>
<head>
  <title><?php echo "Hello World!"; ?></title>
</head>
<body><h1><?php echo "Hello World!"; ?></h1><p><?php echo "Un bonjour de Yogui."; ?></p></body>
</html>
```

Pour visualiser le résultat, il n'est pas suffisant de double cliquer sur le fichier... Il vous faut passer par le serveur Web. Ainsi, ouvrez votre navigateur Internet et chargez la page :

<http://localhost/test.php>

Le résultat de ce script ne manquera pas de vous rappeler l'exemple de document HTML vu auparavant...

De la même manière qu'en HTML, les espaces et sauts de ligne ont peu d'importance en PHP. Le script suivant est équivalent au précédent :

```
<html>
<head>
  <title>
    <?php
      echo "Hello World!";
    ?>
  </title>
</head>
<body>
  <h1>
    <?php
      echo "Hello World!";
    ?>
  </h1>
  <p>
    <?php
      echo "Un bonjour de Yogui.";
    ?>
  </p>
</body>
</html>
```

 Évidemment, l'intérêt de PHP est très limité dans ces exemples... Allons-y pas à pas, ceci étant une introduction.

Voyons maintenant avec une variable (une variable est un mot commençant par le signe "dollar" \$) :

```
<html>
<head>
  <title>
    <?php
      $title = "Hello World!";
      echo $title;
    ?>
  </title>
</head>
<body>
  <h1>
    <?php
      $title = "Hello World!";
      echo $title;
    ?>
  </h1>
  <p>
    <?php
      $text = "Un bonjour de Yogui.";
      echo $text;
    ?>
  </p>
</body>
</html>
```

Voici un code équivalent mais à la construction plus flexible :

```
<?php
$title = "Hello World!";
$text = "Un bonjour de Yogui.";
?>
<html>
<head>
  <title><?php echo $title; ?></title>
</head>
<body>
```

```
<h1><?php echo $title; ?></h1>
<p><?php echo $text; ?></p>
</body>
</html>
```

Ici, j'utilise les variables \$title et \$text. Cela me permet de ne pas répéter "Hello World". Le gain est ici minime, cependant j'imagine que vous commencez à voir l'intérêt des variables : elles permettent de réutiliser du texte à l'infini, sans devoir le réécrire. Par ailleurs, si je souhaite modifier le titre de ma page, je n'ai plus qu'à modifier la variable au début du script pour que toute ma page soit mise à jour d'un seul coup. C'est là tout l'intérêt de PHP, mais il faut aller bien plus loin pour l'exploiter selon tout son potentiel...

 *Dans les équations mathématiques, une variable est une valeur inconnue ou recherchée. En PHP, c'est une valeur que l'on connaît mais que l'on peut modifier à volonté. Dans les deux cas, on utilise une équation : "x = 5" en mathématiques ou bien "\$x = 5;" en PHP.*

Ci-dessus, les lignes qui se trouvent dans les balises PHP s'appellent des **expressions**. Elles sont délimitées par des points virgules. Il faut prendre garde à ne pas oublier ce point virgule : il est fondamental car il sépare les expressions. Si on l'oublie, PHP a tendance à crier très fort qu'il y a une "syntax error" et il nous dit vers quelle ligne de notre script cela se produit.

 *Un peu de vocabulaire : donner une valeur à une variable s'appelle "assigner une valeur à une variable".*

Allons maintenant un tout petit peu plus loin en utilisant ce que l'on appelle une *structure de contrôle* : la conditionnelle **if**.

```
<?php
if(isset($_GET["title"]))
{
    $title = $_GET["title"];
}
else
{
    $title = "Hello World";
}

$text = "$title, I am Yogui!";
?>
<html>
<head>
    <title><?php echo $title; ?></title>
</head>
<body>
    <h1><?php echo $title; ?></h1>
    <p><?php echo $text; ?></p>
</body>
</html>
```

Appelez ce script comme vous en avez maintenant l'habitude : le résultat est le même qu'auparavant. Maintenant, ajoutez ceci à la fin de la barre d'adresse du navigateur :

```
?title=bouh
```

Cela s'appelle un *paramètre GET*. Dans notre script, nous utilisons une variable **\$\_GET** qui teste la présence d'une variable "title". Si elle existe dans **\$\_GET**, alors on la met dans une *variable locale* que nous appelons **\$title**, sinon on construit cette variable locale **\$title** avec une valeur par défaut. Lorsque vous validez le chargement de la page, le titre et le contenu de la page ont pris la valeur que vous avez mise dans la barre d'adresse. Vous venez juste d'envoyer avec succès des informations au serveur Web et de faire votre première page réellement dynamique !

**i** Dans ces exemples d'utilisation, nous avons utilisé **empty()** et **isset()**. Ces constructions du langage PHP s'apparentent à des fonctions, c'est-à-dire qu'on les utilise avec leur nom ("echo" ou "isset") et en mettant quelque chose dans des parenthèses. Dans le cas d'echo, les parenthèses sont facultatives.

**i** Notez la syntaxe de la structure **if** : elle accepte un paramètre entre parenthèses comme le fait une fonction, puis il faut lui donner du code entre accolades. Ce qui est placé dans la branche du **if** n'est exécuté que si la condition est avérée, sinon seulement la branche du **else** est exécutée. C'est l'une ou l'autre, jamais les deux à la fois.

**!** **Sécurité**  
 Je me dois de vous prévenir que le dernier exemple est vulnérable à ce que l'on appelle la faille XSS. Pour le moment, sachez simplement qu'il faut toujours faire attention aux informations que vous prenez de `$_GET`, `$_POST`, `$_FILES`, `$_REQUEST` et toutes ces variables remplies de données provenant de l'internaute.

### V-D-3 - Découvrir les inclusions

Un autre avantage majeur des scripts PHP (dits *dynamiques*) sur les fichiers HTML (dits *statiques*) est la possibilité d'appeler un script depuis un autre script. Ce sont des **inclusions** de scripts.

Une application typique de cette fonctionnalité est pour les éléments identiques de page en page sur un site : parties supérieure (*header*) et inférieure (*footer*), menu de navigation, espace de publicité, etc. En effet, si vous avez déjà fait un site de plusieurs pages en HTML, vous n'avez pas manqué de constater que la moindre modification dans l'un des espaces communs à toutes les pages, vous oblige à modifier de la même manière chacune de ces pages. C'est fastidieux, n'est-ce pas ?

PHP vous permet de découper vos pages en éléments indépendants que vous pouvez inclure les uns dans les autres, ce qui vous permet d'avoir **un seul** script pour chaque élément reproduisible. Cela simplifie les mises à jour de votre site puisque chaque élément se situe à un seul emplacement.

Voici un site minimaliste de trois pages avec des inclusions :

```

index.php
<?php
$title = 'Bienvenue';
include 'header.php';
?>
<h1><?php echo $title; ?></h1>
<!--
    Mettre ici le contenu de la page d'accueil
-->
<?php include 'footer.php'; ?>
    
```

```

catalogue.php
<?php
$title = 'Notre catalogue';
include 'header.php';
?>
<h1><?php echo $title; ?></h1>
<!--
    Mettre ici le contenu de la page catalogue
-->
<?php include 'footer.php'; ?>
    
```

```

contact.php
<?php
$title = 'Nous contacter';
include 'header.php';
    
```

#### contact.php

```

?>
<h1><?php echo $title; ?></h1>
<!--
    Mettre ici le formulaire de contact
-->
<?php include 'footer.php'; ?>
    
```

#### header.php

```

<?php
if (empty($title))
{
    $title = 'Nouveau document';
}
?>
<html>
<head>
    <title><?php echo $title; ?></title>
</head>
<body>
    
```

#### footer.php

```

</body>
</html>
    
```

 Je n'ai pas mis la balise h1 dans le script header.php ; vous verrez à l'utilisation pourquoi j'ai fait de cette manière.

## V-E - Avertissement : ce que PHP n'est pas

PHP est avant tout un langage de script. Il permet de développer des applications très performantes et très utiles, mais il ne peut pas tout faire. Il est principalement utilisé pour des applications Web, c'est-à-dire pour exécuter du code sur un serveur Web. Ainsi, il ne faut pas s'étonner si PHP offre très peu de moyens de lire le port série, parallèle ou USB de votre machine, par exemple.

## V-F - Formation

### V-F-1 - Organisme officiel

PHP est développé et maintenu par le  **PHP Group**. La documentation du langage y est facilement accessible, elle est traduite en français et il est possible de lire les commentaires des internautes.

### V-F-2 - Nos ressources habituelles

- La  **FAQ PHP** répond à vos questions posées dans nos forums ;
- Les  **sources PHP** vous donnent des codes permettant de résoudre des problématiques précises ;
- Les  **tutoriels PHP** vous permettront d'apprendre tout ce dont vous avez besoin ;
- Les  **livres PHP** pour toujours avoir une référence sous le coude ;
- Le  **forum PHP** vous permet de poser toutes vos questions qui ne trouvent pas réponse dans nos autres ressources.

## V-F-3 - Notre sélection de cours

### Comparatif PHP/Javascript

Si vous n'êtes pas encore sûr de vous, commencez par réviser  **les différences entre les langages JavaScript et PHP** grâce au comparatif de Julp.

### Guide de style

Adrien Pellegrini n'a bien sûr pas omis PHP dans son  **guide de style**.

### Les formulaires et PHP5

Mon  **cours sur les formulaires** contient toute une partie sur la récupération des données au moyen de PHP.

### Les sessions PHP

Les sessions sont un mécanisme fondamental permettant de conserver des informations entre les diverses pages visitées par l'internaute. Par exemple, dans le modèle classique du Web, si l'on veut restreindre l'accès à une portion d'un site, il faut utiliser un formulaire de connexion. Cependant, sans utiliser les sessions, il faut que l'internaute remplisse et soumette ce formulaire à chaque fois qu'il souhaite visualiser une page, puisque le serveur a la mémoire courte... Les sessions permettent au serveur de ne pas la perdre aussi facilement.

Julp s'est associé à Mathieu Lemoine pour vous présenter  **les sessions PHP** sous tous leurs aspects.

## V-F-4 - Nos tests d'EDI

 Retrouvez la liste de nos ressources dans la page **EDI PHP**.

### PHPEclipse : Programmez librement pour le Web

Comme un grand nombre de développeurs ici-bas, il est très probable que dans le futur vous étendiez vos connaissances à d'autres langages que PHP. En ce cas, Eclipse est fait pour vous. Jean-Pierre Grossglauer a testé le  **plugin PHPEclipse**.

Il existe d'autres dérivés d'Eclipse pour programmer en PHP. Quelques exemples sont : Eclipse PHP, Eclipse PDT, Zend Studio, etc.

### PHPEdit, un IDE complet pour PHP

 **PHPEdit** étant mon éditeur favori, j'en ai testé les majeures fonctionnalités pour vous.

### Test complet de l'éditeur WebExpert de Visicom

BWP-Necromance et Pierre-Baptiste Naigeon ont testé  **WebExpert 6.5**.

## VI - Base de données : Stockage des informations

### VI-A - Introduction

#### Les trois notions fondamentales :

- Le **fichier** est dans un format géré en interne par le SGBD et dont les détails ne nous importent pas ;
- Le **document** est à destination d'un langage de script ;
- L'**outil** est le SGBD et son  **API** associée.

Je vous préviens, vous allez apprendre du vocabulaire !

Les bases de données ( **BDD**) désignent en fait tous les systèmes de stockage qui permettent de conserver des informations (c'est-à-dire des données) dans un lieu sûr, à l'écart de l'application. Il peut s'agir de fichiers de texte, de fichiers au format XML, de systèmes de gestion de bases de données ( **SGBD**), etc.

Vous avez certainement entendu l'un des noms suivants : Oracle, SQL Server, Access, MySQL ? Ce sont tous des SGBDR (le "R" signifie "relationnelles"). MS Office Excel permet également de faire des BDD mais pas au même niveau que les SGBDR que je viens de mentionner.

Une  **base de données relationnelle** permet d'enregistrer les informations sans avoir aucun duplicata, ce qui réduit l'espace de stockage utilisé ainsi que les temps d'accès, tout en simplifiant la mise à jour. Les données (informations) y sont en relation les unes avec les autres (c'est ce qui évite les doublons).

#### Le principe d'une BDD relationnelle est d'enregistrer les informations de manière hiérarchisée :

- Le SGBD contient des *bases de données* (exemples : "livre-d-or") : dans Excel, ce sont les "fichiers" ;
- Chaque base de données est structurée en *tables* (exemples : "user", "message", "sujet") : dans Excel, ce sont les "feuilles" ;
- Chaque table regroupe tous les *champs* d'une même entité (exemples : "id", "login") : dans Excel, ce sont les "colonnes" ;
- Un champ est une *valeur* (exemples : "8480", "BrYs", "30724", "Yogui") : dans Excel, ce sont les "cellules".

 Vous trouverez plus bas des exemples de bases de données.

### VI-B - Outils nécessaires

Il est parfois utile d'avoir sous la main un outil permettant de dessiner les schémas de l'analyse. Vous pouvez vous tourner vers  **l'un des outils conseillés par SQLPro**.

Pour le SGBD (le programme lui-même), vous avez un choix très large. En effet, vous pouvez opter pour n'importe lequel des noms cités en introduction (ou un autre). Prenez simplement garde d'installer à la fois le service et le client.

Vous pouvez vous aider de  **notre comparatif**.

#### Vous aurez besoin :

- D'un outil pour développer votre code SQL (certains IDEs comme Zend Studio, PHPEdit et PHPEclipse intègrent cette fonctionnalité) ;
- D'un outil de gestion de base de données (le SGBD, cf. notre comparatif) ;
- D'un outil d'administration (sauf si celui qui est fourni en standard avec votre SGBD vous convient) :  **phpMyAdmin** (MySQL),  **phpPgAdmin** (PostgreSQL), TOAD pour  **Oracle** ou  **MySQL**, etc.

## VI-C - Votre première BDD

En général, la construction d'une base de données commence par une phase d'analyse. Cela se fait au moyen d'une **méthode d'analyse** : historiquement, les français préfèrent la méthode  **Merise**, tandis que le reste du monde utilise la méthode  **UML** (qui peut également servir pour la programmation). Cette analyse conceptuelle permet de se représenter le fonctionnement du  **système d'informations** pour lequel on cherche à construire une base de données.

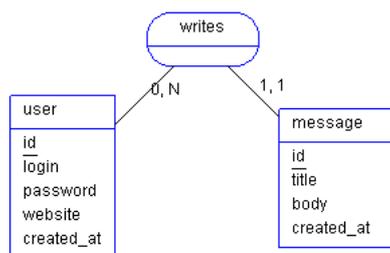
Je vais vous proposer d'utiliser la méthode Merise car je la trouve simple d'approche et très performante.

### VI-C-1 - L'analyse (conception de la structure)

#### Modèle Conceptuel des Données (MCD)

Dans Merise, un MCD décrit le fonctionnement d'un système d'information d'un point de vue fonctionnel, sans aucune considération technique.

Voici comment un livre d'or est représenté selon une *analyse conceptuelle* avec la méthode Merise (pour rappel, un livre d'or est simplement une suite de messages écrits par des utilisateurs) :



Modèle conceptuel d'un livre d'or

 Pour dessiner ce schéma, j'ai utilisé  **AnalyseSI**, gratuit et très simple à utiliser.

Sans entrer dans les détails, il s'agit ici de deux *entités* "user" et "message" liées par une *association* "écrit". Chacune des deux entités est composée de *champs*, certains d'eux jouant le rôle d'*identifiants* (ils sont généralement soulignés). Notez que les entités sont représentées par des noms communs au singulier et que la *relation* est un verbe conjugué.

#### Modèle Logique des Données (MLD)

Suite à l'étape de modélisation conceptuelle ci-dessus, la méthode Merise fournit un procédé permettant d'aboutir à la structure finale de la base de données. C'est le Modèle Logique des Données (MLD), qui décrit comment sont organisées les données.

Le passage au MLD détermine la liste des champs de chacune des tables du système d'information décrit par le MCD. C'est une étape intermédiaire semi technique : les relations du MCD sont devenues des tables et des champs, mais aucun choix technique n'est encore fait.

#### Le schéma précédent se traduit par ce MLD :

- user (id, login, password, website)
- message (id, #user\_id, title, text)

À chaque ligne du MLD est définie une *table* ainsi que la liste de ses *champs*. Les champs soulignés permettent de différencier un enregistrement d'un autre au sein d'une table, ils forment donc la *clef primaire*. Les champs marqués d'un signe dièse (#) sont des informations permettant de relier un enregistrement d'une table avec un enregistrement d'une autre table (lorsqu'ils ont la même valeur), et donc de faire référence à une clef primaire d'une autre table, c'est pourquoi on appelle ces champs des *clefs étrangères* (le champ "user\_id" dans la table "message" est une clef étrangère pour le champ "id" dans la table "user").

La clef primaire est identifiée dès le MCD. Elle permet de récupérer un seul enregistrement dans une table.

La clef étrangère est identifiée à partir du MLD. Elle permet, à partir d'un enregistrement d'une table, de retrouver l'enregistrement correspondant dans une autre table. Dans notre exemple, le numéro d'un message permet de retrouver le nom de son auteur.

## Modèle Physique des Données (MPD)

La dernière étape de la méthode Merise est la conversion du Modèle Logique en Modèle Physique des Données (MPD). En fait, le MPD peut être composé soit de tableaux décrivant le schéma, soit du code SQL de création du schéma : le code SQL est une déduction directe du MPD (en fonction du SGBD choisi et de sa version).

Il s'agit simplement de décrire le type (chaîne de caractères, numérique...) de chacun des champs des tables. C'est très proche du SQL de création de la base de données.

Voici un MPD simplifié pour le MLD précédent :

Nom	Type	Clef
id	numérique	primaire
login	chaîne	unique
password	chaîne	
website	chaîne	

Nom	Type	Clef
id	numérique	primaire
user_id	numérique	étrangère
title	chaîne	
body	texte	

## VI-C-2 - Le SQL (création et mise à jour de la structure, consultation et mise à jour des informations)

Le  **SQL** (Structured Query Language) permet de manipuler la structure des informations ainsi que les informations elles-mêmes.

Pour créer les tables de notre analyse (la structure de notre base de données), on peut utiliser le code SQL suivant :

### Création de la structure en SQL :

```
CREATE TABLE guestbook.user (
  id INT(8) UNSIGNED NOT NULL auto_increment,
  login VARCHAR(50) NOT NULL,
  password VARCHAR(16) NOT NULL,
  website VARCHAR(255) DEFAULT '',
  created_at DATETIME DEFAULT NULL,
  PRIMARY KEY (id)
);
```

### Création de la structure en SQL :

```
CREATE TABLE guestbook.message (
  id INT(8) unsigned NOT NULL auto_increment,
  user_id INT(10) unsigned NOT NULL,
  title VARCHAR(50) NOT NULL,
  body TEXT NOT NULL,
  created_at DATETIME DEFAULT NULL,
  PRIMARY KEY (id)
);

ALTER TABLE guestbook.user
ADD CONSTRAINT uk_user_login UNIQUE (login);

ALTER TABLE guestbook.message
ADD CONSTRAINT fk_message_user FOREIGN KEY (user_id)
REFERENCES guestbook.user (id) ON DELETE CASCADE ON UPDATE CASCADE;
```

Ici, nous voyons qu'un champ `user_id` a été ajouté à la table `message` depuis l'analyse : c'est dû au fait qu'un message ne peut appartenir qu'à un utilisateur. Ainsi, nous pouvons déterminer qui a écrit quel message. L'ID de l'auteur est une information propre au message.

Une fois la structure du schéma prête, nous pouvons envoyer des requêtes pour en modifier ou pour en lire les données.

Voici comment remplir le schéma avec des informations :

### Utilisateurs :

```
INSERT INTO user (login, password, website, created_at)
VALUES ("BrYs", "1234", "brys.developpez.com", NOW());

INSERT INTO user (login, password, website, created_at)
VALUES ("Yogui", "4321", "g-rossolini.developpez.com", NOW());
```

### Messages :

```
INSERT INTO message (user_id, title, body, created_at)
VALUES (1, "Bonjour", "Un bonjour de Paris ;)", NOW());

INSERT INTO message (user_id, title, body, created_at)
VALUES (2, "Super site", "J'apprécie ton site, continue ainsi !", NOW());

INSERT INTO message (user_id, title, body, created_at)
VALUES (2, "Merci pour tout", "J'oubliais de dire que ton site m'a beaucoup servi !", NOW());
```

La recherche (lecture) se fait au moyen de l'instruction `SELECT`.

### Toutes les infos de tous les utilisateurs :

```
SELECT *
FROM user;
```

L'instruction `WHERE` permet de filtrer les données retournées par la requête SQL.

### Toutes les infos de l'utilisateur 'Yogui' :

```
SELECT *
FROM user
WHERE login = 'Yogui';
```

L'instruction JOIN permet de mettre en relation deux tables qui ont un champ en commun : dans cet exemple, il s'agit de user.id et message.user\_id. Notez qu'il est préférable de réutiliser le nom des tables pour éviter les conflits : indiquer seulement id porterait à confusion entre l'ID du message et l'ID de l'utilisateur.

Tous les messages de l'utilisateur "Yogui" :

```
SELECT user.login, message.title, message.body
FROM message
INNER JOIN user ON message.user_id = user.id
WHERE user.login = 'Yogui';
```

Ici, nous pouvons voir un exemple de la relation entre les tables user et message : nous utilisons le numéro de l'utilisateur pour dire à qui appartient chaque message. De cette manière, nous ne répétons pas les informations contenues dans la table "users", ce qui réduit l'espace disque utilisé et simplifie la mise à jour des informations des utilisateurs : il suffit de modifier le contenu de la table "users".

 *Le SQL est prévu pour être simple à lire et relativement intuitif à comprendre. Cependant, il existe mille manières d'arriver à un seul résultat. Je vous recommande de longuement l'étudier avant de vous lancer dans des schémas complexes, vous gagnerez en efficacité. C'est ce qui s'appelle la normalisation.*

### VI-C-3 - Exemples concrets

Je vais utiliser le modèle conceptuel présenté ci-dessus pour vous donner un exemple de BDD avec plusieurs systèmes : XML, Excel, Access et MySQL.

Cette suite d'exemples est principalement destinée à vous donner une vision d'ensemble, de vous montrer qu'il est possible d'obtenir le même résultat de différentes manières.

#### XML

XML est simplement un format de stockage des données dans un fichier texte. On peut donc visualiser les données avec n'importe quel éditeur de texte, mais il faut développer une application avec un langage de programmation pour pouvoir effectuer des consultations complexes. Il existe un langage pour parcourir un arbre XML :  XPath.

users.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<users>
  <user>
    <id>1</id>
    <login>BrYs</login>
    <password>1234</password>
    <website>brys.developpez.com</website>
    <created_at>2010-01-16 23:04:46</created_at>
  </user>
  <user>
    <id>2</id>
    <login>Yogui</login>
    <password>4321</password>
    <website>g-rossolini.developpez.com</website>
    <created_at>2010-01-16 23:04:47</created_at>
  </user>
</users>
```

messages.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<messages>
  <message>
    <id>1</id>
    <user_id>1</user_id>
```

## messages.xml

```

        <title>Bonjour</title>
        <body>Un bonjour de Paris ;)</body>
        <created_at>2010-01-16 23:04:48</created_at>
    </message>
    <message>
        <id>2</id>
        <user_id>2</user_id>
        <title>Super site</title>
        <body>J'apprécie ton site, continue ainsi !</body>
        <created_at>2010-01-16 23:04:49</created_at>
    </message>
    <message>
        <id>3</id>
        <user_id>2</user_id>
        <title>Merci pour tout</title>
        <body>J'oubliais de dire que ton site m'a beaucoup servi !</body>
        <created_at>2010-01-16 23:04:50</created_at>
    </message>
</messages>
    
```

Exemple d'application permettant de retrouver les messages de Yogui :

## xml-messages-yogui.php

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US" lang="en-US">
<head>
    <title>Messages de Yogui (source : XML)</title>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<?php

function display($string)
{
    echo htmlentities(utf8_decode($string), ENT_QUOTES, 'ISO-8859-1');
}

$xml_users = file_get_contents("users.xml");
$xml_messages = file_get_contents("messages.xml");

$src_users = simplexml_load_string($xml_users);
$src_messages = simplexml_load_string($xml_messages);

$users = $src_users->xpath("/users/user[login = 'Yogui']");
$yogui = $users[0];

if($yogui)
{
    ?>
    <table border="1">
        <tr>
            <th>login</th>
            <th>title</th>
            <th>text</th>
        </tr>
        <?php
        $messages = $src_messages->xpath("/messages/message[user_id = '". $yogui->id.'"]");
        foreach($messages as $message)
        {
            ?>
            <tr>
                <td><?php display($yogui->login); ?></td>
                <td><?php display($message->title); ?></td>
                <td><?php display($message->text); ?></td>
            </tr>
        <?php
    
```

xml-messages-yogui.php

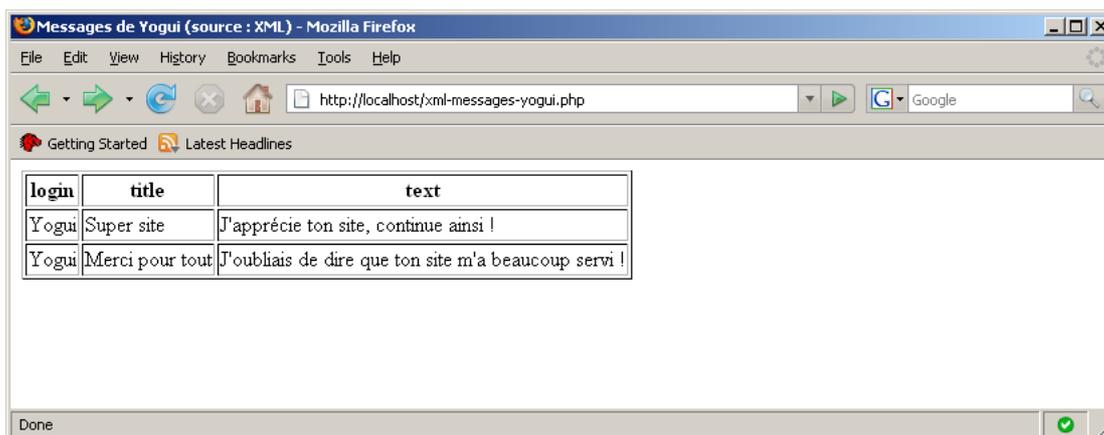
```

    }
    ?>
</table>
<?php
}

?>
</body>
</html>

```

Et voici le résultat dans le navigateur Web :



**!** XML est un très bon moyen d'enregistrer des données mais il comporte un inconvénient majeur : il n'y a aucun automatisme. La cohérence des données n'est pas contrôlée par un SGBD, ainsi je peux modifier n'importe quelle valeur user.id du fichier users.xml sans me rendre compte qu'ensuite je ne pourrai probablement plus utiliser la liaison avec le champ message.user\_id du fichier messages.xml.

## MS Office Excel

Excel ne dispose pas du langage SQL mais il permet néanmoins de sauvegarder les données avec l'organisation hiérarchique adoptée par les autres SGBD. C'est en quelque sorte une alternative graphique à XML, il n'y a pas besoin de s'occuper de remplir les balises mais seulement de remplir les cellules. C'est plus simple à remplir, mais plus difficile à utiliser avec une application externe.

Excel est à la fois le moteur de stockage et l'outil de visualisation des données. Le fichier de données n'est pas lisible avec n'importe quel éditeur de texte mais Excel permet d'effectuer des consultations complexes sur les données.

Dans un classeur "Livre d'or", créons les feuilles "user" et "message" :

	A	B	C	D	E	F	G
1	1	BrYs	1234	brys.developpez.com			
2	2	Yogui	4321	g-rossolini.developpez.com			
3							
4							
5							
6							
7							

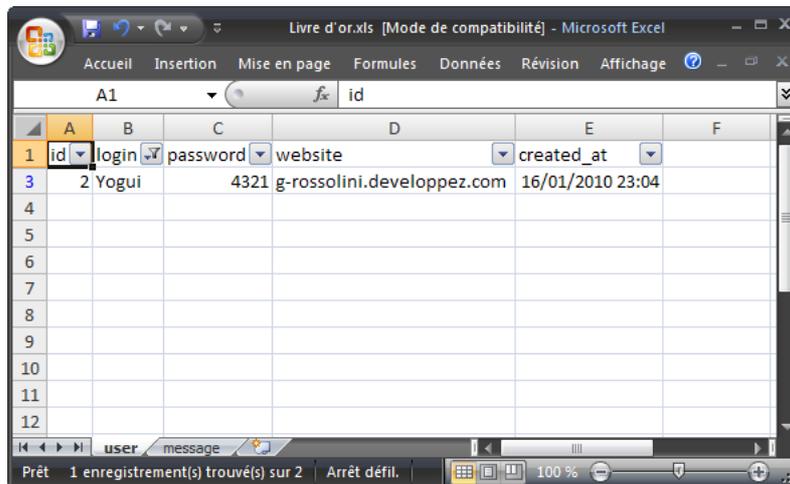
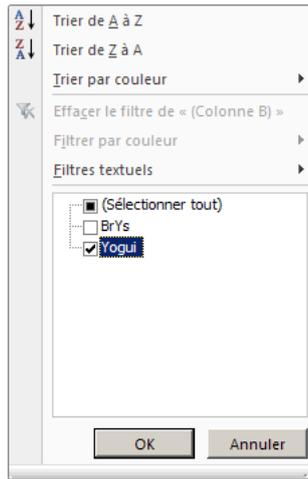
	A	B	C	D	E
1	1	1	Bonjour	Un bonjour de Paris ;)	
2	2	2	Super site	J'apprécie ton site, continue ainsi !	
3	3	2	Merci pour tout	J'oubliais de dire que ton site m'a beaucoup servi !	
4					
5					
6					
7					

### Les feuilles du classeur :

- **user :**
  - A Identifiant automatique ;
  - B Pseudonyme ;
  - C Mot de passe ;
  - D Site Web.
- **message :**
  - A Identifiant automatique ;
  - B Identifiant de l'auteur ;
  - C Titre ;
  - D Texte.

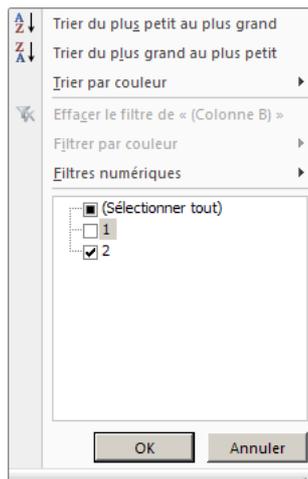
 Excel permet de filtrer les données feuille par feuille, mais pas de mettre en relation plusieurs feuilles. Il m'est par exemple impossible d'avoir directement la liste des messages de l'utilisateur "Yogui".

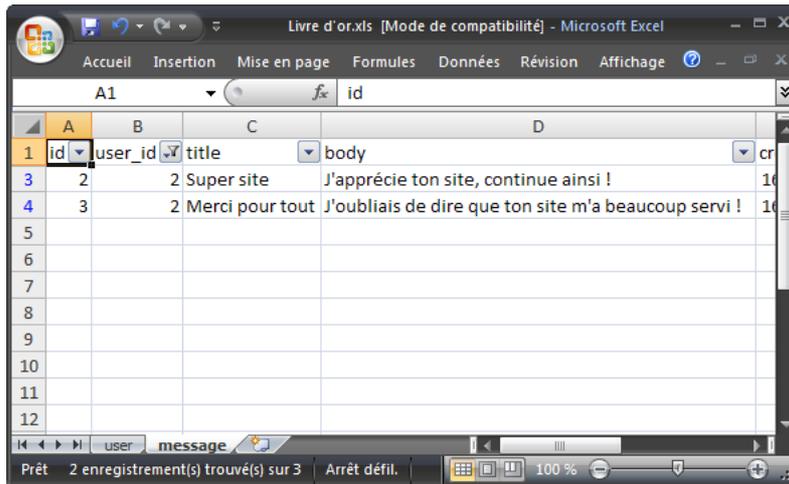
Pour obtenir la liste des messages de "Yogui", il faut commencer par obtenir son ID (colonne A) en filtrant la feuille "user" par la colonne B ("login") :



Nous n'avons que les informations de l'utilisateur 'Yogui'

💡 En filtrant ensuite la feuille "message" sur la colonne B ("user\_id") à l'aide de l'ID que nous venons d'obtenir, nous obtenons indirectement les messages de "Yogui" :





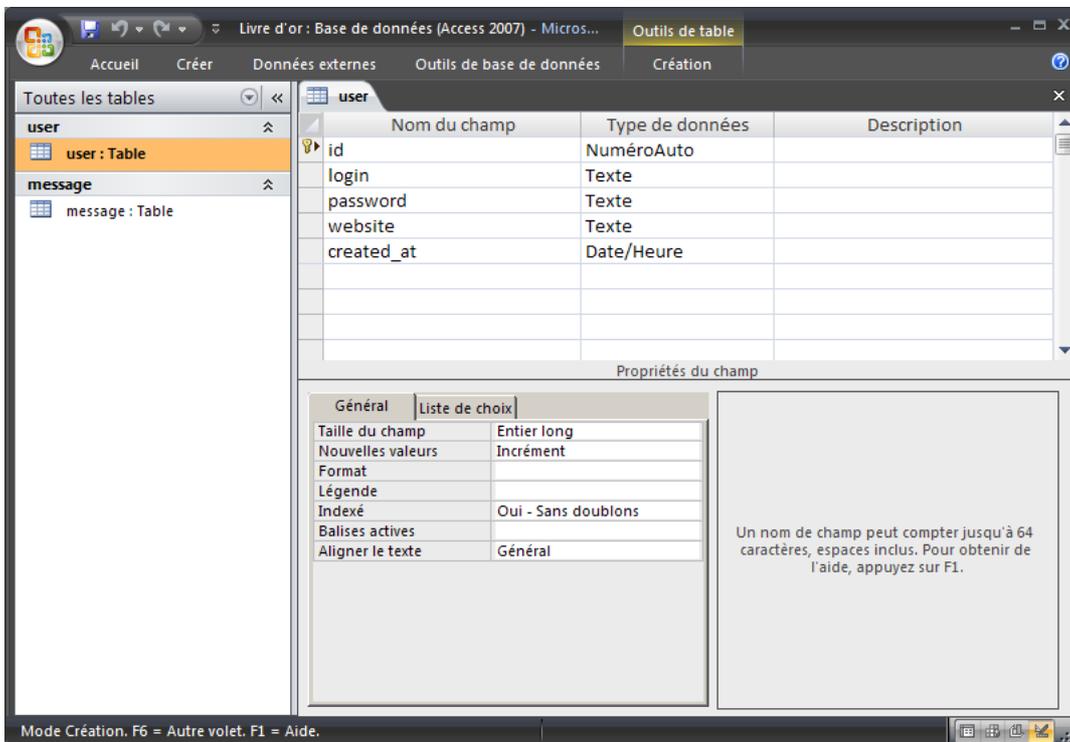
*Nous n'avons que les messages de l'utilisateur 2, c'est-à-dire 'Yogui'*

**⚠** Cette méthode de recherche est fastidieuse, peu pratique. Il vaut mieux utiliser un SGBD dès lors que les feuilles sont liées les unes aux autres (même raison que pour XML).

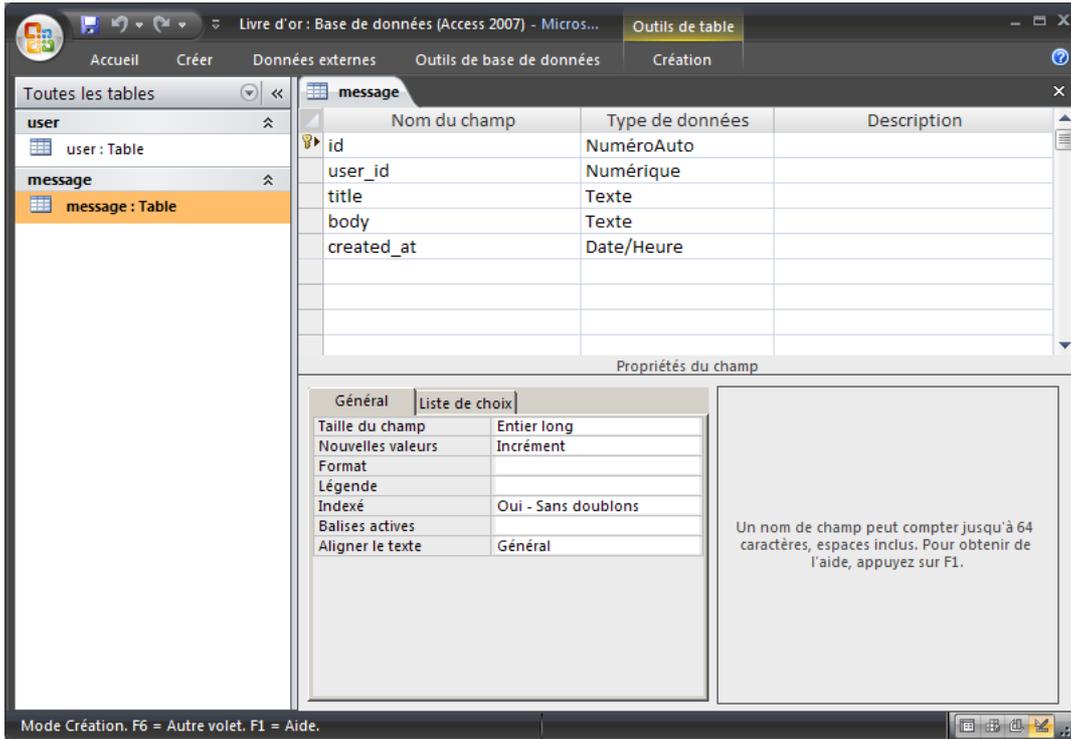
## MS Office Access

Access propose des fonctionnalités plus complètes qu'Excel pour la gestion de bases de données.

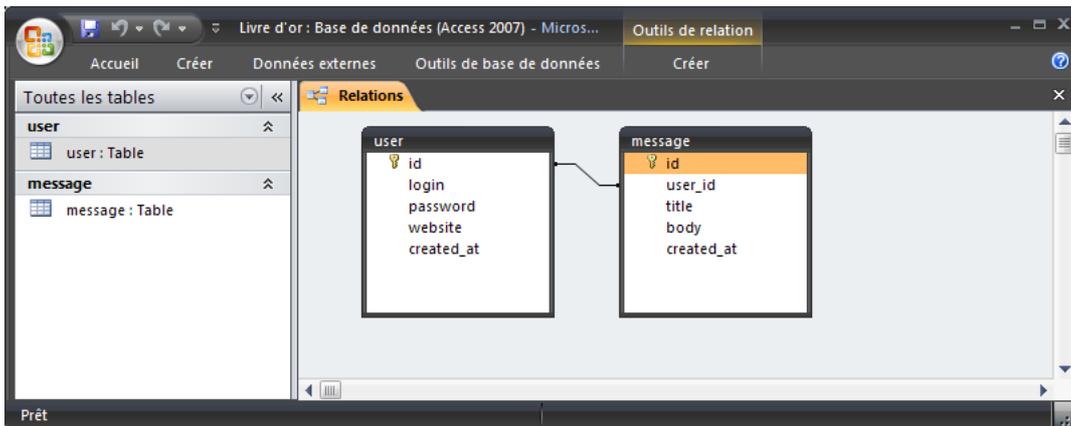
Access, tout comme Excel, est à la fois le moteur de stockage et l'outil de visualisation des données. Le fichier de données n'est pas lisible avec n'importe quel éditeur de texte. Access permet d'effectuer des consultations complexes sur les données. Il est difficile d'utiliser une application externe pour consulter les données.



*Ci-dessus : structure de la table "user"*

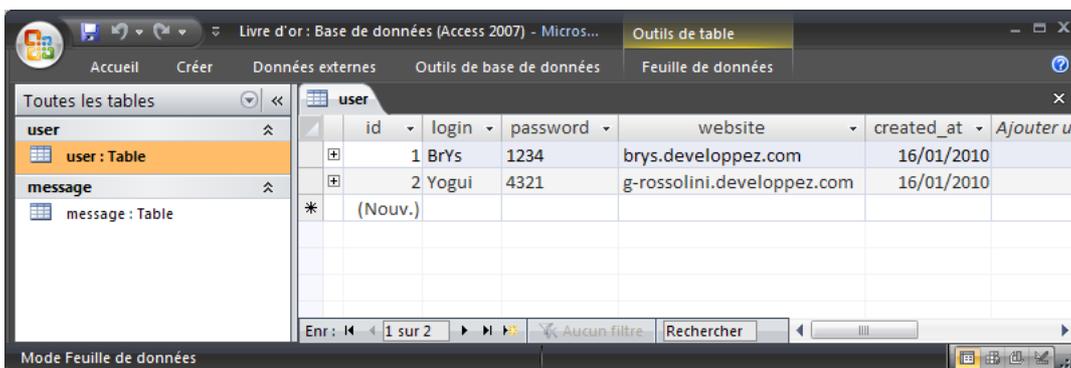


*Ci-dessus : structure de la table "message"*



*Ci-dessus : relations entre les tables "user" et "message"*

Les données contenues dans les tables sont les mêmes que précédemment :



*Ci-dessus : informations contenues dans la table "user"*

The screenshot shows the Microsoft Access interface with the 'message' table selected in the 'Outils de table' ribbon. The table data is displayed as follows:

id	user_id	title	body
1	1	Bonjour	Un bonjour de Paris ;)
2	2	Super site	J'apprécie ton site, continue ainsi !
3	2	Merci pour tout	J'oubliais de dire que ton site m'a beaucoup ser
* (Nouv.)			

Ci-dessus : informations contenues dans la table "message"

Access étant un SGBDR, il permet de relier les tables les unes aux autres. Cela nous permet donc d'obtenir en une opération la liste des messages de l'utilisateur "Yogui" à l'aide de la requête SQL proposée plus haut :

The screenshot shows the 'Outils de requête' ribbon with a query named 'Requête1'. The SQL query is as follows:

```
SELECT user.login, message.title, message.body
FROM message
INNER JOIN user ON message.user_id = user.id
WHERE user.login = 'Yogui';
```

Ci-dessus : requête en mode SQL

The screenshot shows the result of the SQL query in 'Outils de requête'. The result is displayed as follows:

login	title	body
Yogui	Super site	J'apprécie ton site, continue ainsi !
Yogui	Merci pour tout	J'oubliais de dire que ton site m'a beaucoup ser
* (Nouv.)		

Ci-dessus : résultat d'une requête

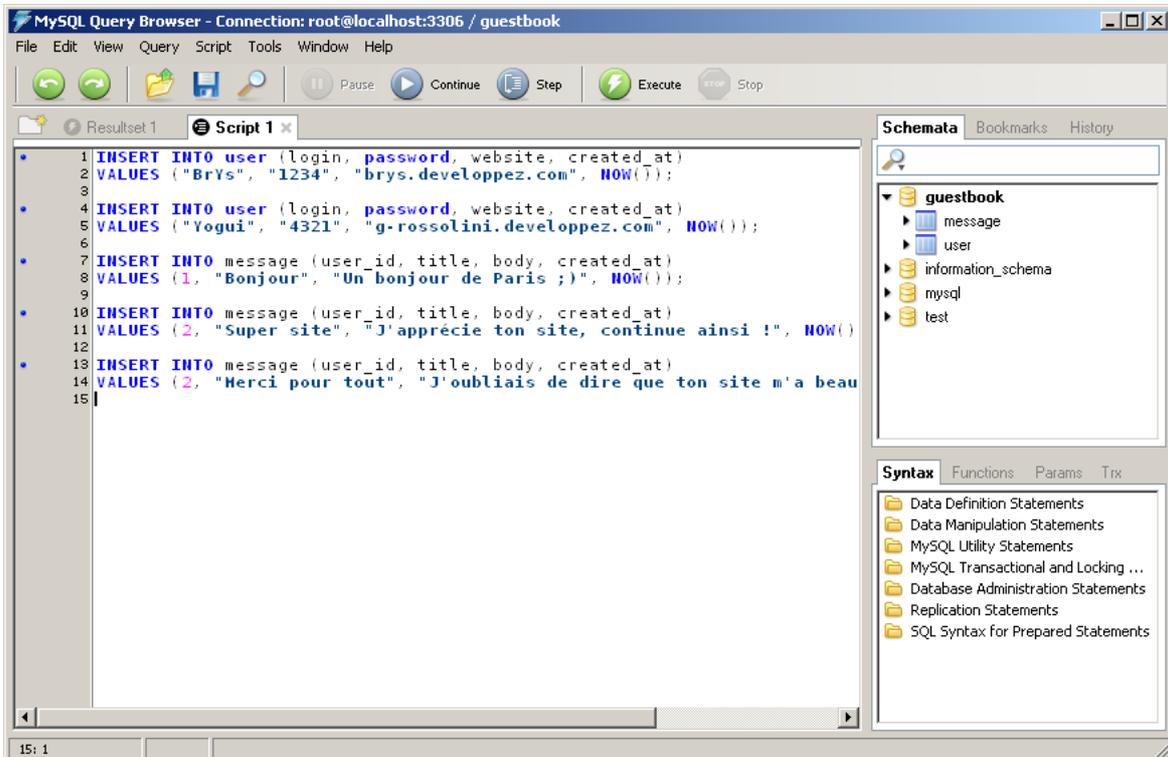
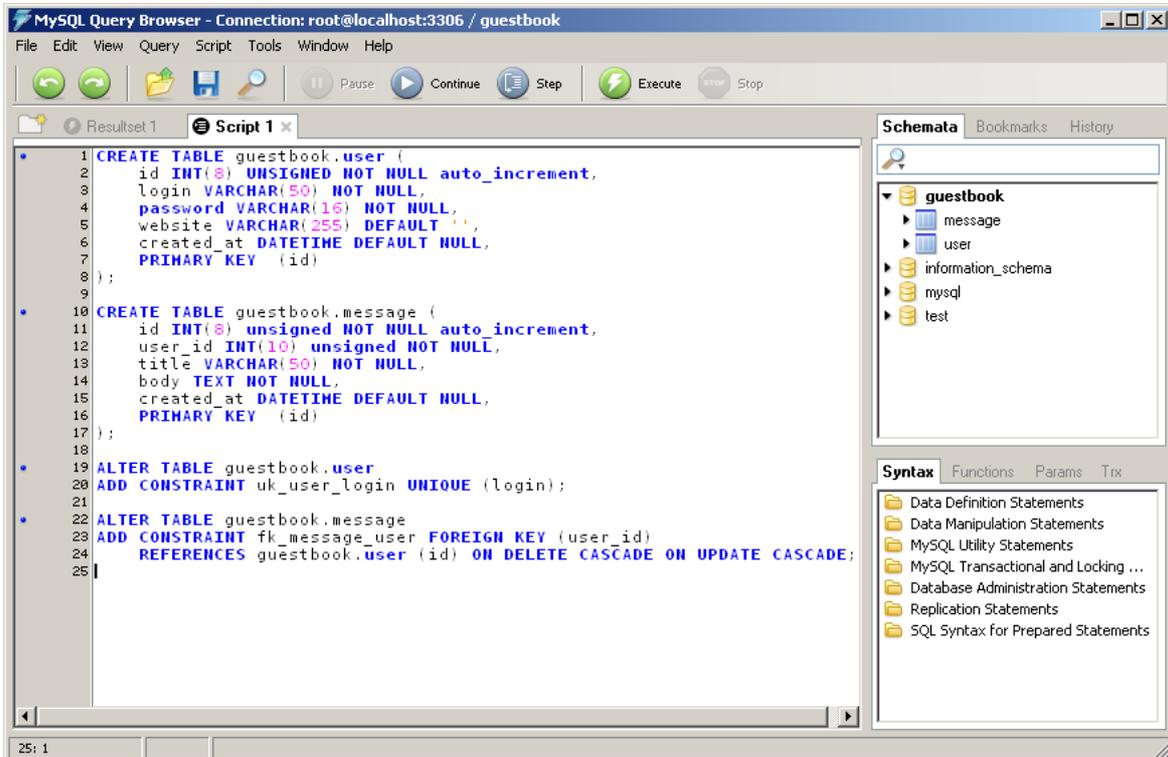
**!** Access (et tous les équivalents des autres suites de bureautique, par exemple OpenOffice.org Base) est un très bon outil de gestion de BDD en bureautique mais il souffre de certaines lourdeurs. Une BDD Access ou Base est adaptée pour de petites applications pour un seul utilisateur sur son poste de travail, mais elle est impraticable en environnement Web.

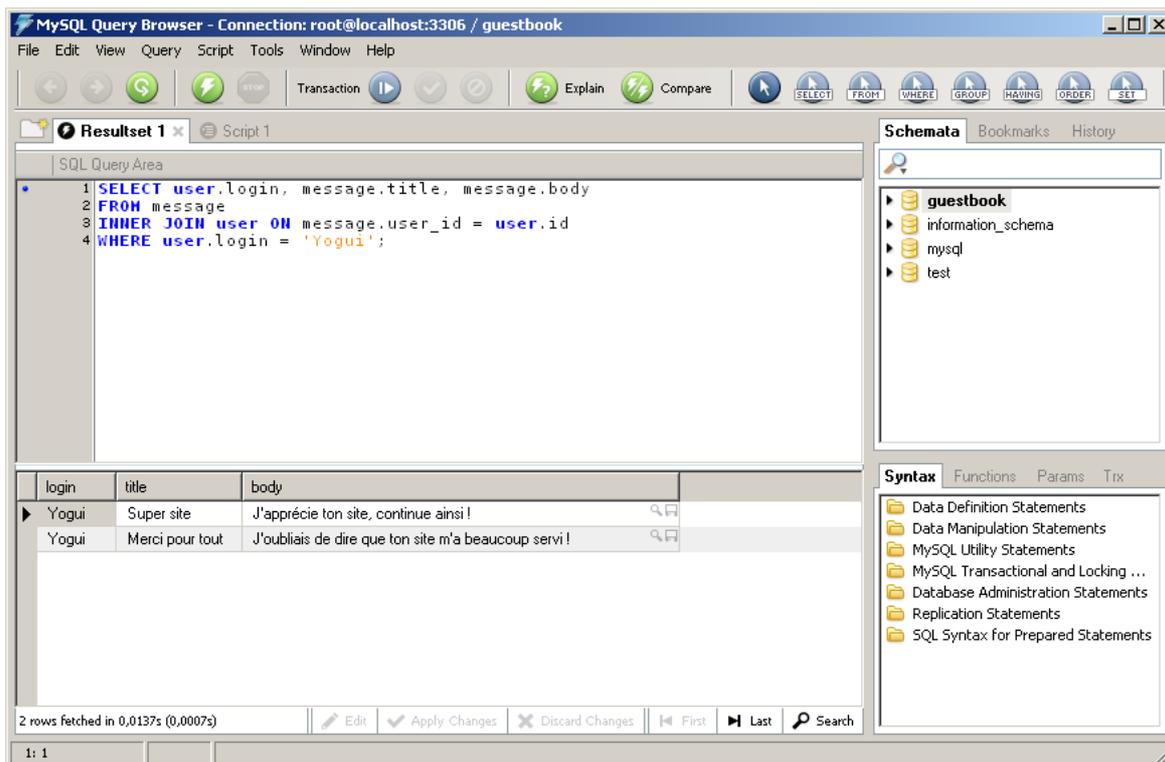
## MySQL

MySQL est un véritable SGBDR, c'est-à-dire qu'il permet principalement de conserver les données sur le disque dur et qu'il dispose d'une gestion interne des droits d'accès. Pour visualiser les données, il faut développer une application à l'aide d'un langage de programmation et du langage SQL. Des outils d'administration peuvent se substituer cette

l'application externe pour les tâches d'administration de la base de données, mais l'application devra être développée à un moment ou à un autre afin de permettre aux utilisateurs d'utiliser la BDD. Dans le cas d'Internet, les utilisateurs sont les internautes, les visiteurs du site Web.

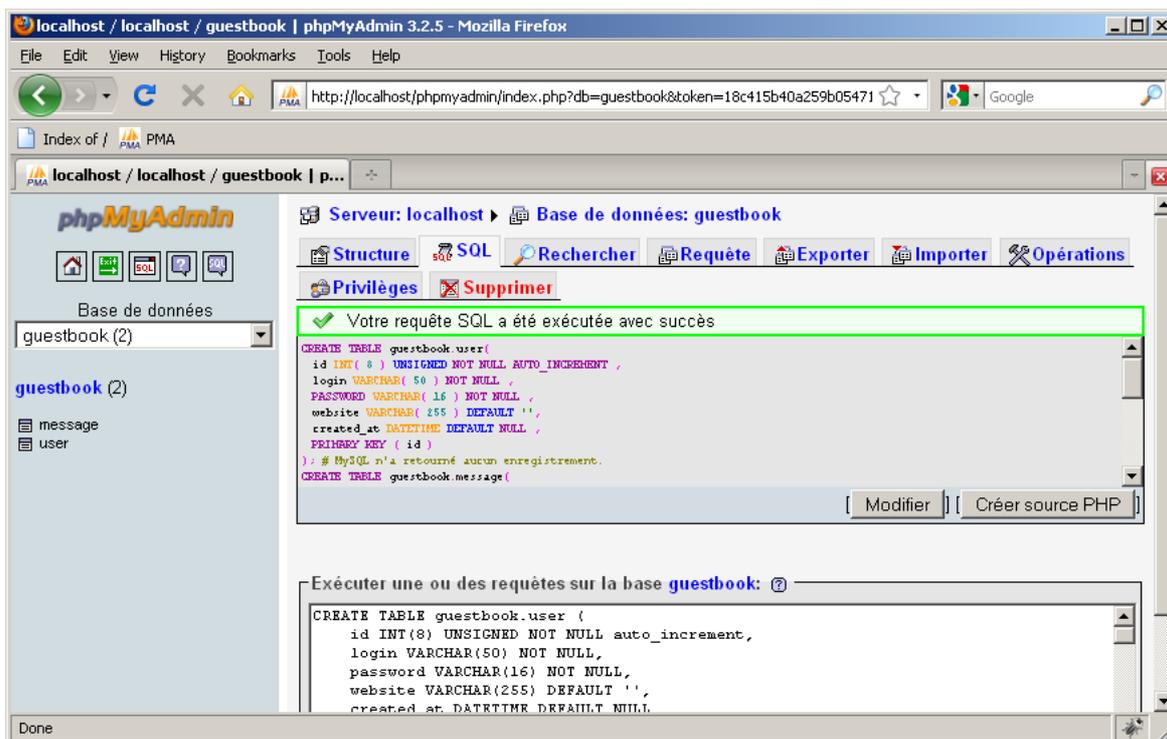
Nous pouvons par exemple utiliser  **MySQL Query Browser**, un outil d'administration gratuit :

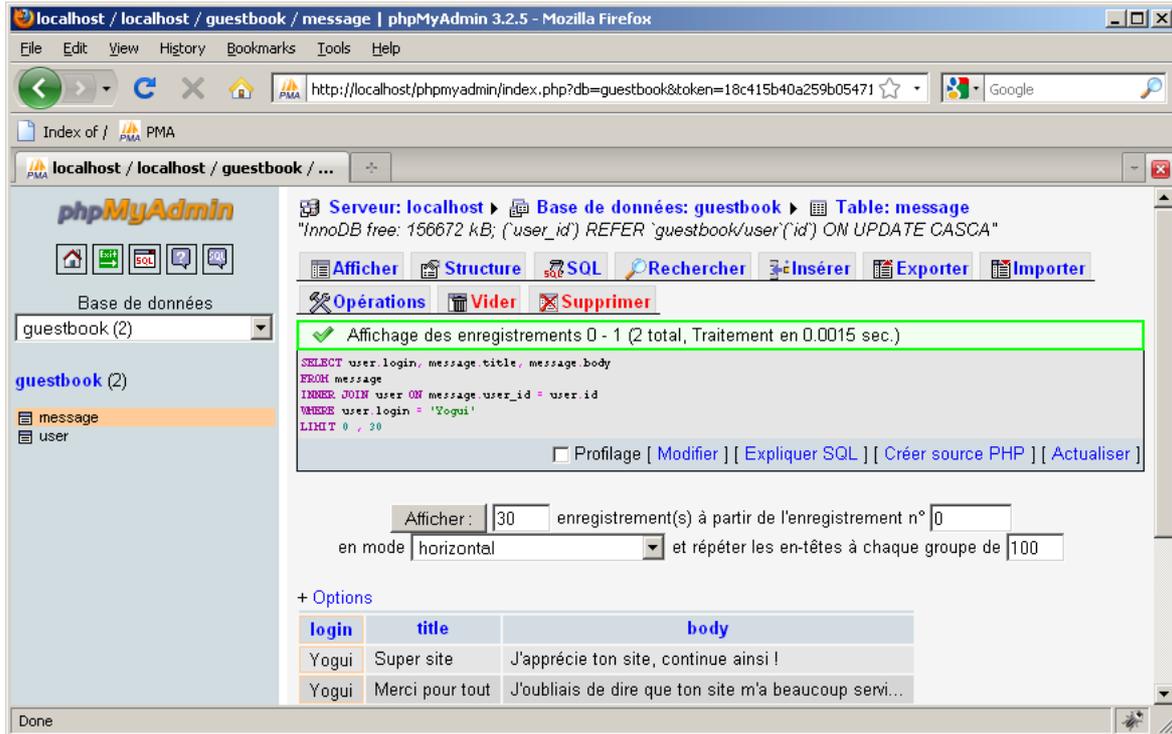
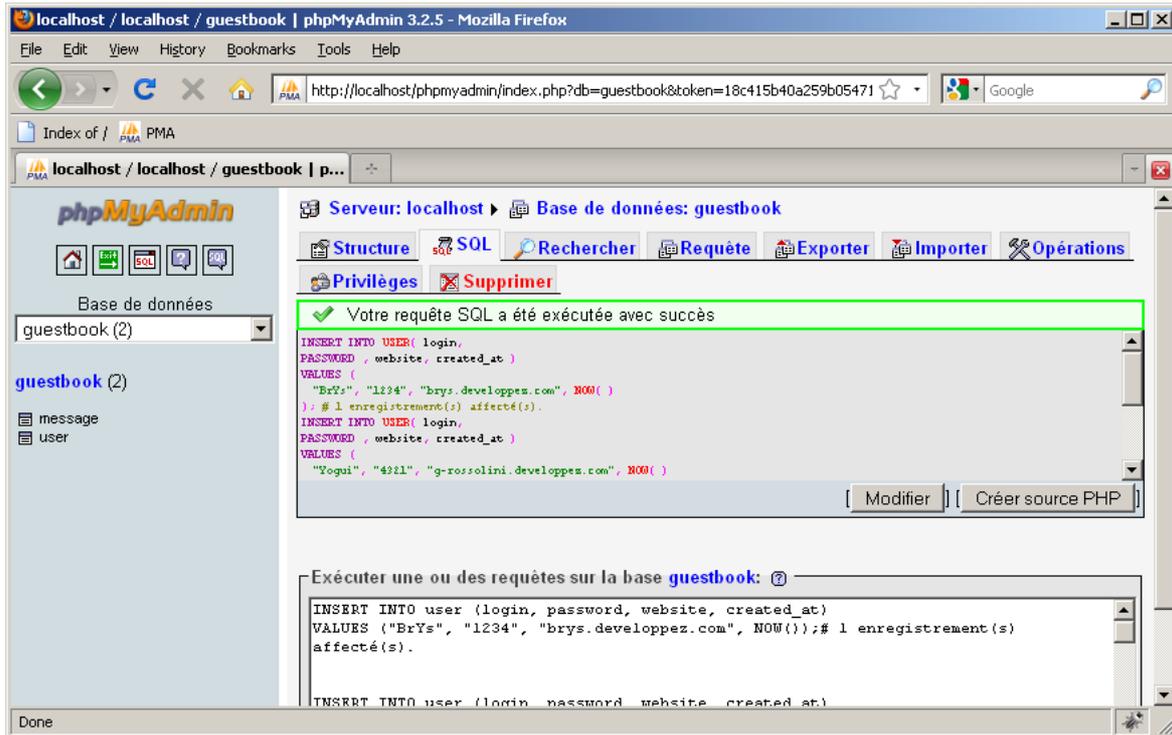




L'intérêt d'utiliser un SGBD comme MySQL est d'avoir la possibilité de choisir l'application "client", c'est-à-dire l'application qui permet de consulter et de mettre à jour la BDD. Ci-dessus, j'ai utilisé une application de bureau mais, en développement Web, nous utiliserons très peu ce genre d'outils. Il est préférable de développer des pages Web à l'aide d'un langage de script comme PHP.

Exemple avec l'outil phpMyAdmin :





Voici un exemple de script PHP donnant le même résultat :

```

mysql-messages-yogui.php
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US" lang="en-US">
<head>
  <title>Messages de Yogui (source : MySQL)</title>
  <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
</head>

```

mysql-messages-yogui.php

```

<body>
<?php

function display($string)
{
    echo htmlentities($string, ENT_QUOTES, 'ISO-8859-1');
}

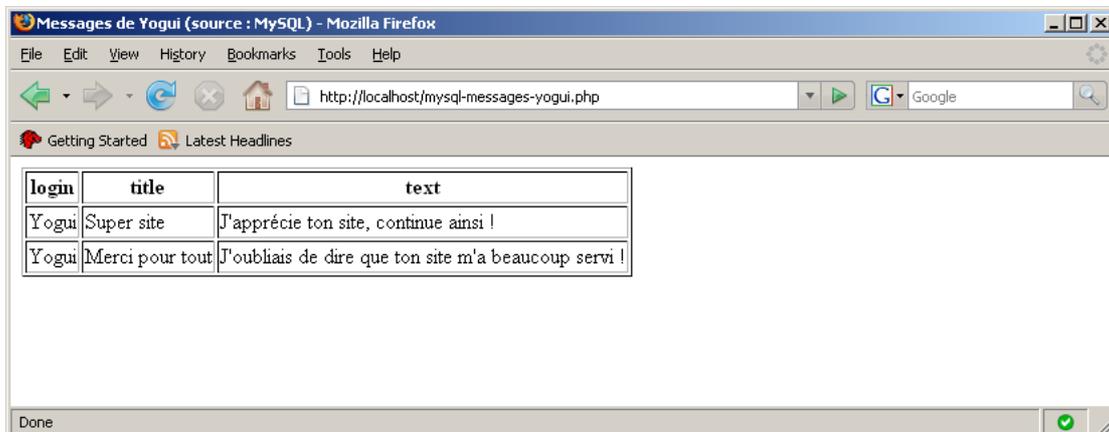
mysql_connect('localhost', 'root', '') or die(mysql_error());
mysql_select_db('guestbook') or die(mysql_error());

$sql = "SELECT user.login, message.title, message.body
FROM message
INNER JOIN user ON message.user_id = user.id
WHERE user.login = 'Yogui'";

$result = mysql_query($sql) or die(mysql_error());
if($result)
{
    ?>
    <table border="1">
        <tr>
            <th>login</th>
            <th>title</th>
            <th>text</th>
        </tr>
        <?php
        while($message = mysql_fetch_assoc($result))
        {
            ?>
            <tr>
                <td><?php display($message['login']); ?></td>
                <td><?php display($message['title']); ?></td>
                <td><?php display($message['text']); ?></td>
            </tr>
            <?php
        }
        ?>
    </table>
    <?php
}
?>
</body>
</html>

```

Et voici le résultat dans le navigateur Web :



 C'est à ce dernier résultat que nous voulons parvenir : MySQL en stockage des données + des scripts PHP pour les consulter.

💡 Il est possible de mettre en place des règles de gestion afin d'empêcher la mise à jour de certains champs, ou bien d'effectuer certaines actions lorsque des événements précis ont lieu. Par exemple ici, nous avons défini une contrainte d'intégrité qui relie les champs `user.id` et `message.user_id` : lorsque nous modifions une valeur `user.id`, MySQL met automatiquement à jour les valeurs correspondantes dans `message.user_id` de telle manière que la base de données reste toujours cohérente.

## VI-D - Formation

### VI-D-1 - Organisme de référence

Malgré l'existence d'une norme SQL, il faut consulter chaque SGBD pour obtenir une documentation fiable. Les organismes  ANSI et  ISO se sont chargés de la normalisation du langage.

### VI-D-2 - Nos ressources habituelles

- La  FAQ MySQL et la  FAQ PostgreSQL répondent à vos questions posées dans nos forums ;
- Les  sources PostgreSQL vous donnent des codes permettant de résoudre des problématiques précises ;
- Les  tutoriels MySQL vous permettront d'apprendre tout ce dont vous avez besoin ;
- Les  livres MySQL et les  livres PostgreSQL pour toujours avoir une référence sous le coude ;
- Le  forum BDD, le  forum MySQL et le  forum PostgreSQL vous permettent de poser toutes vos questions qui ne trouvent pas réponse dans nos autres ressources.

### VI-D-3 - Notre sélection de cours

#### Conception d'une base de données

Cyril Gruau vous présente la conception de bases de données grâce à  Merise. Laurent Audibert, pour sa part, vous propose  UML2.

#### Quelques notions de manipulation de données : SQL

SQL fonctionne au moyen d'un serveur de bases de données. On y envoie des *instructions* au moyen d'un *client* : PHP jouera cet office dans vos applications Web.

J'ai mis plus haut un exemple de code utilisant une syntaxe "CREATE". C'est ce que l'on appelle le langage de définition de données ( DDL) : il sert à manipuler la **structure** des informations de la base de données. Baptiste Wicht traite uniquement la  création des tables, tandis que  SQLPro est plus complet.

Après avoir créé vos tables avec le DDL, il peut être intéressant de les remplir avec vos informations : cela se fait au moyen du langage de manipulation de données ( DML), vu ci-dessus par l'instruction "INSERT INTO". Elle sert à manipuler le contenu de la base de données, à savoir les informations elles-mêmes. Baptiste Wicht couvre simplement la  manipulation de données, alors que SQLPro est  là aussi exhaustif.

Enfin, le but ultime d'une base de données étant d'être consultée, vous pouvez utiliser l'instruction "SELECT" vue ci-dessus. De nouveau, Baptiste Wicht fait une  présentation plus sommaire que SQLPro,  fidèle à lui-même.

## Utiliser une BDD avec PHP

En fait, SQL et PHP sont deux choses totalement différentes. L'un peut fonctionner sans l'autre mais ils peuvent également collaborer. Pour y parvenir, je vous recommande la lecture du cours " **passer des requêtes MySQL en PHP**" par Eusebius. Une fois que vous êtes un peu mieux préparés, je vous recommande le tutoriel de Pierre-Baptiste Nageon sur la " **création d'un site dynamique**".

## Écrire du bon code SQL

Et bien sûr, Adrien Pellegrini revient avec son " **guide de style pour SQL**".

## VII - Conclusion générale

### VII-A - Épilogue

Apprendre à faire des pages Web n'est pas une mince affaire mais cela peut en valoir la peine. Je pense que le plus important est de ne pas perdre le fil, d'utiliser les technologies à bon escient et non dans tous les sens.

### VII-B - Pour aller plus loin

Pour aller plus loin, je vous recommande la lecture de nos autres cours  **HTML**,  **CSS**,  **PHP**,  **JavaScript** et  **BDD**.

En particulier, le résumé " **Comprendre l'environnement Web**" par Éric Berger pourra vous être bien utile.

### VII-C - Contribuez

Si vous souhaitez contribuer à l'enrichissement de nos ressources, nous serions heureux d' **accueillir vos propositions** !