

Web Mining

Web Technologies: Server-Side Technologies





Langages côté serveur

Un exemple : PHP

Introduction aux bases de données

Outils et références






- Le Web, ce n'est pas qu'un ensemble de documents HTML statiques !
- Les programmes côté serveur permettent :
 - de traiter des soumissions de formulaire ;
 - d'afficher de manière uniforme l'ensemble des pages d'un site ;
 - de proposer des applications interactives ;
 - de permettre à l'utilisateur d'ajouter ou modifier du contenu ;
 - etc.



■ CGI (**Common Gateway Interface**) : interface normalisée permettant de faire communiquer le serveur Web avec un programme s'exécutant sur le serveur ;

- CGI permet d'utiliser n'importe quel langage de programmation (**compilés** comme C, C++, Java, ou **interprétés** comme Perl, Python, Ruby, etc.) pour écrire des langages côté serveur.
- Mais certains langage sont plus adaptés au développement Web :
 - comportent des fonctions spécialement dédiées à HTTP, HTML, etc. ;
 - s'intègrent de manière plus efficace et plus pratique avec le serveur Web (moyennant des extensions logicielles) ;
 - ont une syntaxe spécialement conçue, qui mêle code HTML envoyé tel quel et instructions de programmation interprétées.
- Quelle que soit la technologie utilisée, **le code du programme n'est pas accessible par le navigateur Web**, seulement le résultat de son exécution.



 : un des langages les plus populaires, s'intègre très facilement avec Apache (libre)

ASP et ASP.NET : destiné à être utilisé avec IIS (Microsoft, commercial)

ColdFusion (Adobe, commercial)

JSP (Java Server Pages) : permet de mêler instructions Java et code HTML ; nécessite un serveur d'applications Java (p. ex., Tomcat) en plus d'Apache (Sun, gratuit voire libre)

Servlets Java : véritables programmes Java, plutôt pour les applications complexes côté serveur avec peu d'interaction côté client ; nécessite un serveur d'applications Java en plus d'Apache (Sun, gratuit voire libre)



- Les langages présentés ci-avant restent assez basiques et généralistes.
- N'encouragent pas forcément une organisation propre d'un site Web.
- **Framework** : ensemble d'un langage de programmation, d'une bibliothèque de fonctions, d'outils externes, de bonnes pratiques à suivre. . .
- Permet d'abstraire la création d'une page Web.
- Suit en général le modèle **MVC** (voir transparent suivant).
- Inclut parfois la génération de code JavaScript **côté client** pour créer directement une application Web fortement dynamique (p. ex., validation de formulaire) ; intégration Ajax également.
- Fortement recommandé pour créer des applications complexes. . . mais également complexe à maîtriser !





- Principe de génie logiciel, utilisé dans d'autres domaines
- Particulièrement adapté au cas des applications Web !
- Séparation propre entre :
 - Modèle** : données manipulées par l'application et fonctions de manipulation de ces données ; **réutilisable** pour d'autres applications Web
 - Vue** : présentation des données ; facilement **échangeable** pour changer l'apparence et la structure du site
 - Contrôleur** : contrôle la manière dont l'utilisateur interagit, au travers de la vue, avec les données du modèle





ASPI.NET : DotNetNuke

ColdFusion : Model-Glue, Fusebox

Java : Struts, Spring, JavaServer Faces, Google Web Toolkit

Perl : Catalyst

PHP : CakePHP, Symphony, Zend

Python : Django

Ruby : Ruby on Rails (a eu beaucoup d'influence !)

Smalltalk : Seaside

... et beaucoup d'autres !



Permettent de créer des sites Web sans aucun développement.

■ Fonctionnalités :

- édition simplifiée de page (syntaxe wiki ou bbcode, ou contrôle JavaScript texte enrichi) ;
- ajout de contenu externe (images, documents annexes, etc.) ;
- gestion d'utilisateurs, contrôle d'accès, etc. ;
- modules de gestion de forums, de blogs ;
- thèmes graphiques prêt à l'emploi ;
- contrôle de version.

■ Suivant les CMS, extensions pouvant être nombreuses.

■ Certains CMS spécialisés : blogs (Dotclear, Movable Type, TypePad), commerce électronique (PrestaShop, Magento), forums (phpBB, MyBB), etc.



Nombre de téléchargements hebdomadaires :



WordPress	PHP	433 767
Joomla!	PHP	189 429
Drupal	PHP	62 500
Umbraco	.NET/XSLT	5 670
eZ Publish	PHP	5 612
CMS Made Simple	PHP	4 903
SilverStripe	PHP	2 500
e107	PHP	2 242
Xoops	PHP	1 209
TikiWiki	PHP	373
phpWebSite	PHP	347
Typo3	PHP	100
Alfresco	Java	57
DotNetNuke	ASP.NET	?
Jahia	Java	?
Liferay	Java	?
modx	PHP	?
OpenCMS	Java	?
Plone	Python	?
TextPattern	PHP	?

(Chiffres difficiles à obtenir, source : Water&Stone)





Même en utilisant un CMS, il est utile de connaître les technologies de base du Web (HTML, CSS, JavaScript, langage côté serveur) :

- pour créer ses propres styles CSS (quasi-indispensable) ;
- pour développer des applications complexes, propres au site ;
- pour développer ou adapter des petites extensions ;
- pour comprendre ce qui se passe en cas de problème ;
- pour s'assurer que les pages respectent certaines conditions (validité W3C, accessibilité).





Frontière peu nette entre les deux... Les Wikis mettent l'accent sur :

- a priori, tout est éditable par un grand nombre d'utilisateurs ; pas de notion de « propriété » d'une page ou d'un document.
- insistance sur le contrôle de versions

Nombreux wikis : MediaWiki, TWiki, Dokuwiki...





Point auxquels faire attention :

- la licence (la plupart des CMS populaires sont libres, mais pas tous)
- le langage sous-jacent, important s'il y a besoin de modifier/personnaliser le logiciel, et également important pour savoir comment le déployer
- la disponibilité de fonctionnalités spécifiques au site : blogs, forums, support vidéo, etc. Il est possible d'utiliser des composants externes pour chacune des tâches, mais plus simple de rester sur un seul système.





- Plus lent qu'un site classique ; fonctionnalités de cache permettant d'accélérer, mais peut poser problème dans certains contextes.
- Failles de sécurité, code sur lequel on n'a pas de contrôle.
- Logiciels orphelins.
- Migration vers un autre système potentiellement coûteuse.
- Non adapté à tous les usages.





nécessaire d'avoir accès au code source, donc comment savoir avec quelle technologie côté serveur un site a été réalisé ?

- URL : (.php pour PHP, .jsp pour JSP, .asp pour ASP, /servlet/ pour une servlet. .). Tout cela est configurable, mais bien souvent la configuration par défaut est utilisée.
- Ça peut être écrit en toutes lettres sur le site (souvent pour un CMS)
- Regarder les commentaires dans le code HTML, CSS. . .
- Organisation des fichiers et répertoires, bibliothèques JavaScript ou CSS chargées, etc.
- Cookies, en particulier identifiants de session





Langages côté serveur

Un exemple : PHP

Introduction aux bases de données

Outils et références



- Script PHP : document HTML (par exemple), dans lequel est incorporé du code PHP.

- Le code PHP est à l'intérieur d'une pseudo-balise `<?php ... ?>` (ou `<? ... ?>`, ou `<?= ... ?>` qui est un raccourci pour `<? echo ... ?>`).

Exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<html>
  ...
  <body>
    <h1><?php echo 2+2; ?></h1>
  </body>
</html>
```





- Un script PHP est une suite d'instructions terminées par des points-virgules.

Exemple (Écrire une phrase avec l'instruction echo)

```
echo 'Ceci apparaîtra dans la page générée.';
```

- Ces instructions contiennent des éléments variables ou constants, peuvent être conditionnées ou encore itérées plusieurs fois.





Une **variable** est le nom d'un **réceptier** destiné à contenir une valeur. Dans un ordinateur, ce réceptier est en fait une zone mémoire.

- L'**affectation** `$var=valeur` est une instruction qui permet de changer la **valeur** d'une variable `$var` :
 - La valeur de la variable à gauche de `=` est remplacée par la valeur à droite de `=`.
- L'affectation est une instruction dite **destructive** : l'ancienne valeur de la variable est détruite, écrasée, effacée par la nouvelle valeur !
- Une **valeur** peut être :
 - un nombre entier ou réel ;
 - une chaîne de caractères définie entre deux guillemets simple (`'`) ou entre deux guillemets double (`"`).





- La valeur affectée à une variable peut également être le résultat d'une **expression**.
- **Attention**, à droite d'une affectation `=`, les variables utilisées dans l'expression désignent les valeurs qu'elles contiennent.

Exemple

`$Compteur = $Compteur + 1;` a pour effet de mettre le résultat de la somme de la valeur de `$Compteur` avec la valeur 1 dans le récipient `$Compteur`.



■ Une instruction conditionnelle est de la forme :

si la condition C est vraie alors faire liste d'instructions T
sinon faire liste d'instructions E

- où une **condition** est une expression et *T* (resp. *E*) est un bloc d'instructions.
- En PHP, on utilise la syntaxe suivante :

```
if (C) { T } else { E }
```

Exemple

```
if ($sexe=='f') {  
    echo 'Madame';  
} else {  
    echo 'Monsieur';  
}
```





La condition est généralement un test de comparaison, ou plusieurs tests **reliés** par des opérateurs logiques.

- Le `else` et son bloc associé sont optionnels.
- Si un bloc ne contient qu'une seule instruction, on pourra omettre les accolades.

Exemple

```
if ($sortie=='q') { echo 'Merci et au revoir!'; }  
  
if ($a > $b) { echo "$a est plus grand que $b"; }  
else {  
    if ($a==$b) { echo "$a est égal a $b"; }  
    else { echo "$a est plus petit que $b"; }  
}
```



- Boucle : permet de répéter une opération

```
for( I ; C ; P ) { B }
```

- où *I* est une affectation permettant d'initialiser la variable itérée,
- *C* est la condition d'arrêt de la boucle (valeur limite que doit atteindre la variable itérée),
- *P* est le pas de la boucle : c'est une affectation qui permet de modifier à chaque itération la valeur de la variable itérée,
- *B* est le bloc d'instructions itéré.

Exemple

```
for( $i=0 ; $i<10 ; $i=$i+1 ) { echo 'Au secours!' ; }  
  
$fact=1;  
for( $i=13 ; $i>0 ; $i=$i-1 ) { $fact = $fact*$i ; }
```





Les paramètres HTTP peuvent être récupérées en PHP grâce au tableaux associatifs `$_GET` et `$_POST`.

- Les valeurs de ce tableau peuvent être des variables simples ou des tableaux indicés : ces derniers sont les paramètres à choix multiples dont on a suffixé le nom de `[]` dans le code HTML.

Exemple

```
echo "<p>Votre login est : " . $_POST["login"] . "</p>";
echo "<p>Vous avez coché les genres : ";
for($i=1;$i<=count($_POST['genre']);$i=$i+1) {
    echo $_POST['genre'][$i] . " ";
}
echo "</p>";
```





Un script PHP avec du code PHP incorporé au sein de code HTML n'est pas un document XHTML valide !

- Ce qu'il faut valider, c'est une (des) page(s) HTML produites par le script.
- Possibilité d'indiquer une URL au validateur du W3C.
Malheureusement pas utilisable quand l'URL est privée.
Possibilité dans ce cas de sauvegarder le fichier HTML produit, et de l'envoyer au validateur du W3C comme fichier local.
- Il n'y a pas de « validateur » PHP, mais les erreurs de syntaxe causeront des erreurs à l'exécution du script.
- Rien ne change pour les CSS, on peut les valider à part.





Langages côté serveur

Un exemple : PHP

Introduction aux bases de données

Outils et références





- **SGBD** : Système de Gestion de Bases de Données
- Fournit des méthodes **efficaces** pour gérer des données qui répondent à une structure (un **schéma**) précis.
- Rechercher des données : **requêtes**
- Ajouter, supprimer, modifier des données : **mises à jour**
- Traite de manière rapide de grandes quantités de données.
- Gérer des transactions (ne pas donner à deux clients la même place dans un train !)





Modèle le plus répandu et le plus classique.

- Les données sont organisées en des **tables**, chacune des colonnes représentant un **attribut** des données.

Exemple

Prénom	Nom	Date de naissance
Jean	Dupont	1967-08-07
Pascale	Dupuis	1981-09-12
Alfred	Lambert	NULL

- Chaque attribut (colonne) est **typé**.
- SQL (**S**tructured **Q**uery **L**anguage) : langage standard de requête et de mise à jour des données (petites variantes suivant les SGBD).





Oracle

IBM DB2

Microsoft SQL Server

Microsoft Access

MySQL

PostgreSQL

commercial, le plus utilisé en entreprise

commercial

commercial

commercial, pauvre en fonctionnalités

libre, le plus utilisé sur le Web

libre, plus abouti et plus complexe que MySQL





(Types MySQL)

INT : entier (42)

REAL : nombre en virgule flottante (3.14159)

VARCHAR(N) : chaîne de caractères ayant au plus N caractères ; les valeurs sont délimitées par des apostrophes ('Ceci est une chaîne').

TEXT : longue chaîne de caractères, sans limite de taille

DATE : date (2005-11-08)

TIME : temps (09:30:00)





- **NULL** : valeur spéciale
- Dénote l'absence de valeur.
- Différent de `0`, de `''` ...
- Une comparaison normale (`=`, `<>`) avec **NULL** renvoie toujours FAUX.
- **IS NULL**, **IS NOT NULL** peuvent être utilisées pour tester une valeur.
- Chacune des colonnes doit être déclarée comme acceptant ou non la valeur **NULL**.





Langage standardisé pour interagir avec un SGBD.

- Exemple de requête :

```
SELECT a.name, COUNT(*)  
FROM actors a, casting m  
WHERE a.id=m.actor_id  
GROUP BY a.name
```

- Exemple de mise à jour :

```
UPDATE availabilities  
SET nb_available=nb_available-1  
WHERE train_id=1234
```





- SGBD très utiles pour beaucoup d'applications Web (annuaires, catalogues, réservations, forums de discussion, blogs, etc.)
- Possibilité d'accéder aux bases de données plus ou moins simplifiée suivant les langages :
 - En PHP, bibliothèques de fonctions propres à chaque SGBD (p. ex., `mysql` ou `mysqli` pour MySQL).
 - En JSP, bibliothèque JDBC avec les scriptlets et balises `<sql:*>` avec la JSTL, pour accéder à n'importe quel SGBD.



```
mysql_pconnect("server","login","password")
{ echo "<p>Desolé, connexion impossible</p>"; exit; }
if(!mysql_select_db('database'))
{ echo "<p>Desolé, accès à la base impossible</p>"; exit; }

$resultat= mysql_query("SELECT * FROM Films");
if($resultat) {
    while($film=mysql_fetch_assoc($resultat)){
        echo "<p>".$film["Titre"]." est paru en ".
            $film["Annee"]." </p>";
    }
} else {
    echo "<p>Erreur dans l'exécution de la requête.</p>";
    echo "<p>Message de MySQL: ".mysql_error()."</p>";
}
```





Langages côté serveur

Un exemple : PHP

Introduction aux bases de données

Outils et références



■ Outils HTTP : Netcat, Wget, Curl

■ Firebug pour analyser du trafic HTTP

- N'importe quel éditeur de textes pour la plupart des langages côté serveur
- Un serveur Web (p. ex., Apache)
- L'interpréteur du langage côté serveur (p. ex., PHP et mod_php pour Apache)
- Pour JSP, Servlets : un serveur d'applications Java (p. ex., Tomcat)
- Un SGBD (p. ex., MySQL) ; sous Windows, EasyPHP regroupe Apache+PHP+MySQL
- La plupart du temps, un moyen de transférer les programmes vers le serveur Web distant (SSH, FTP, application Web. . .)





HTTP

- <http://www.faqs.org/rfcs/rfc2616.html>
- <http://www.ietf.org/rfc/rfc2109.txt>
- <http://www.ietf.org/rfc/rfc2965.txt>

PHP

- <http://www.php.net/>
- *Pratique de MySQL et PHP*, Philippe Rigaux, O'Reilly

MySQL : <http://dev.mysql.com/doc/>





Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après et à l'exclusion expresse de toute utilisation commerciale.

Le droit d'usage défini par la licence autorise un usage à destination de tout public qui comprend :

- le droit de reproduire tout ou partie du document sur support informatique ou papier,
- le droit de diffuser tout ou partie du document au public sur support papier ou informatique, y compris par la mise à la disposition du public sur un réseau numérique,
- le droit de modifier la forme ou la présentation du document,
- le droit d'intégrer tout ou partie du document dans un document composite et de le diffuser dans ce nouveau document, à condition que :
 - L'auteur soit informé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel et non exclusif.

Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur : sitepedago@telecom-paristech.fr

