Web dynamique (M2105)

"There are some kernel developers who would not speak to me again if I told them I was playing with web technologies" (James Bottomley, heard at LinuxCon Japan, 2016)

Eugen Dedu

Maître de conférences
Univ. de Franche-Comté, IUT de Belfort-Montbéliard
Dépt. R&T, 1ère année
Montbéliard, France

mars 2019

http://eugen.dedu.free.fr
eugen.dedu@univ-fcomte.fr

Intégration dans la formation R&T

- Pré-requis (S1) :
  - initiation au développement Web
  - langages de programmation (bases + consolidation)
  - bases de données
- Prolongement : aucun

- FI : 6h CM, 6h TD, 18h TP
- FA : 6h CM, 6h TD, 15h TP
- Examen : pratique, env. 2h15, lors de la dernière séance de TP

- Objectif : savoir concevoir des pages Web simples mais dynamiques en utilisant les dernières technologies
  - exemple : page Web montrant l'utilisation du CPU et du disque dur d'un serveur
- Ce n'est pas faire des sites Web complexes

Exemples de besoins pour un administrateur réseau

- Une page Web qui affiche l'espace disque utilisé et l'utilisation du CPU en graphique (en utilisant PHP et JavaScript) avec l'historique récent (PHP et script shell) – Cédric Michelot
- Gestion des absences des étudiants, avec droits différents pour les enseignants (PHP/MySQL) – Cédric Michelot
- Gestion des ordres de mission des agents (PHP et un peu de JavaScript) – moi-même

Plan du cours

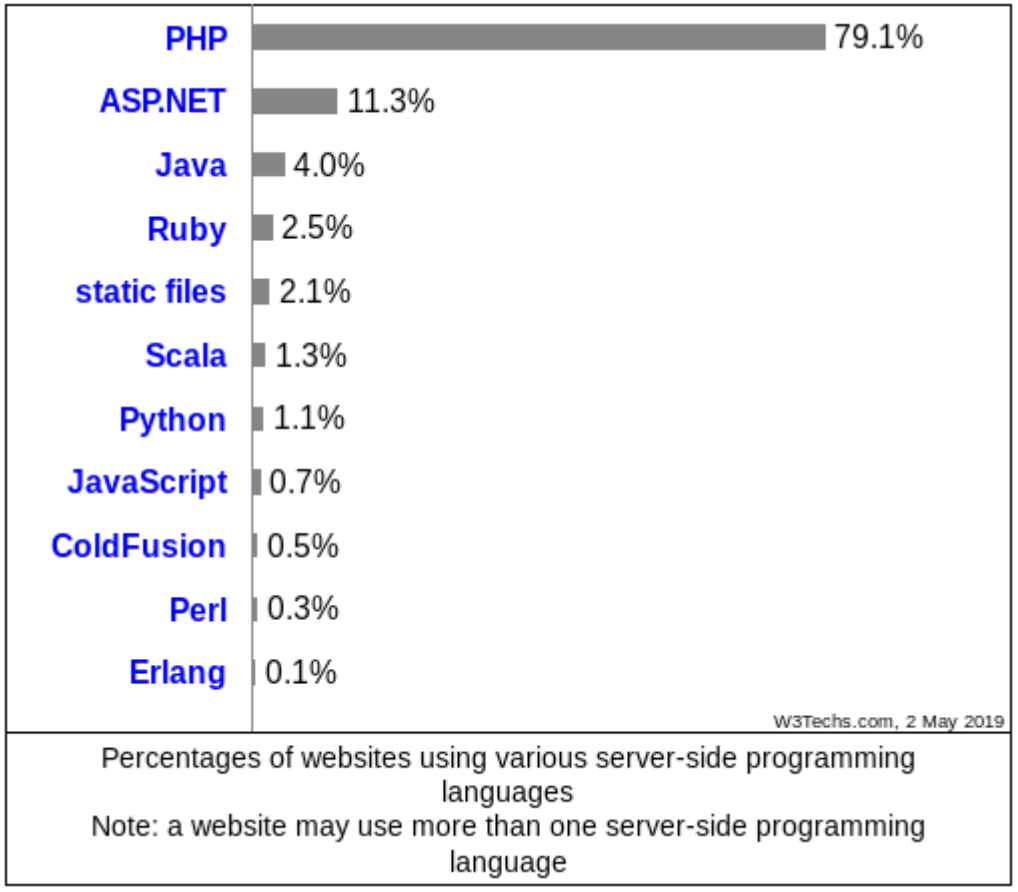
- Scripts côté serveur (PHP)
  - intégration avec une base de données (MySQL)
- Scripts côté client (JavaScript)
  - communication asynchrone client-serveur (AJAX)
- Sécurité du Web
- (CMS : mediawiki, joomla, drupal etc.)

Programming languages used in most popular websites*

Websites	Popularity (unique visitors per month)[1]	Front-end (Client-side)	Back-end (Server-side)	Database	Notes
Google.com[2]	1,600,000,000	JavaScript	C, C++, Go,[3] Java, Python, PHP (HHVM)	Bigtable,[4] MariaDB[5]	The most used search engine in the world
Facebook.com	1,100,000,000	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D,[6] XHP,[7] Haskell[8]	MariaDB, MySQL,[9] HBase, Cassandra[10]	The most visited social networking site
YouTube.com	1,100,000,000	JavaScript	C, C++, Python, Java,[11] Go[12]	Vitess, BigTable, MariaDB[5][13]	The most visited video sharing site
Yahoo	750,000,000	JavaScript	PHP	PostgreSQL, HBase, Cassandra, MongoDB,[14]	Yahoo is presently[*when?*] transitioning to Node.js[15]
Amazon.com	500,000,000	JavaScript	Java, C++, Perl[16]	Oracle Database[17]	Popular internet shopping site
Wikipedia.org	475,000,000	JavaScript	PHP, Hack	MariaDB[18]	"MediaWiki" is programmed in PHP, runs on HHVM; free online encyclopedia
Twitter.com	290,000,000	JavaScript	C++, Java[19], Scala[20], Ruby	MySQL[21]	Popular social network.
Bing	285,000,000	JavaScript	C++, C#	Microsoft SQL Server, Cosmos	
eBay.com	285,000,000	JavaScript	Java,[22] JavaScript,[23] Scala[24]	Oracle Database	Online auction house

https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites

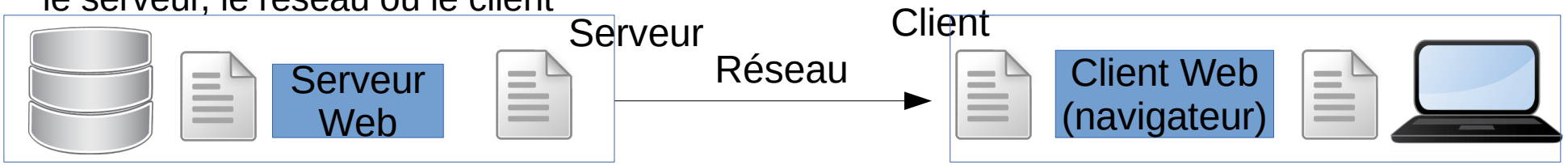
Langage côté serveur par nombre de sites



https://w3techs.com/technologies/overview/programming_language/all

Où les pages Web sont modifiées

- Au début du Web, les pages Web étaient statiques, mais de nos jours beaucoup (voire la plupart ?) des pages sont dynamiques
- La page (fichiers HTML et CSS, y compris les images et autres objets) peut être modifiée par le serveur, le réseau ou le client



- Exemples de pages dynamiques :
  - "l'heure courante est ...h ...m ...s" – il vaut mieux que ce soit le client qui l'actualise en permanence
  - "les maisons à vendre dans la région choisie sont : ..." – il vaut mieux que ce soit le serveur qui l'exécute
  - "les maisons à vendre aujourd'hui à ...h ...m ...s sont : ..." – il vaut mieux utiliser le serveur et le client
- Le changement dépend de l'entrée de l'utilisateur, des conditions environnementales (par ex. la date courante, navigateur) etc.
- Voir plus tard une comparaison plus détaillée

Scripts côté serveur – exemples de pages créées dynamiquement sur le serveur

- Formulaire fait en module Web1
- Google search
- Recherche sur un site d'achats
- Convertisseur devises
- Dans tous les cas, c'est la même page que le client demande (mais avec des paramètres différents en principe)
- Nous allons étudier PHP

Transformation d'une page Web dynamique avec un script côté serveur

Fichier exemple.php :

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page simple</title>
</head>
<body>
  Un tableau :
  <table>
  <?php
    for ($i=0 ; $i < 4 ; $i ++)
      echo '<tr><td>Itération ' .
        $i . '</td></tr>';
  ?>
  </table>
</body>
</html>
```

DD

Serveur Web

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page simple</title>
</head>
<body>
  Un tableau :
  <table>
  <tr><td>Itération 0</td></tr>
  <tr><td>Itération 1</td></tr>
  <tr><td>Itération 2</td></tr>
  <tr><td>Itération 3</td></tr>
  </table>
</body>
</html>
```

Réseau

Navig

Un tableau :
Itération 0
Itération 1
Itération 2
Itération 3

- Le script s'exécute sur le serveur Web
- Lors de la demande de la page, il s'exécute ***une seule fois***, pour générer la page, autrement dit une fois la page générée le script ne change plus la page (mais un script côté client peut continuer à la modifier)
- Le script sort du texte HTML

PHP

- Beaucoup de documentation sur PHP
- Voir cours séparé : http://www.cril.univ-artois.fr/~lecoutre/teaching/web/cours/slidesPHPFr.pdf
- http://www.lephpfacile.com/cours/
- http://cours-info.iut-bm.univ-fcomte.fr/docs/php/bigmanual.html
- http://public.iutenligne.net/informatique/langages/roose/php/general

Formulaires, envoi des paramètres, GET vs POST

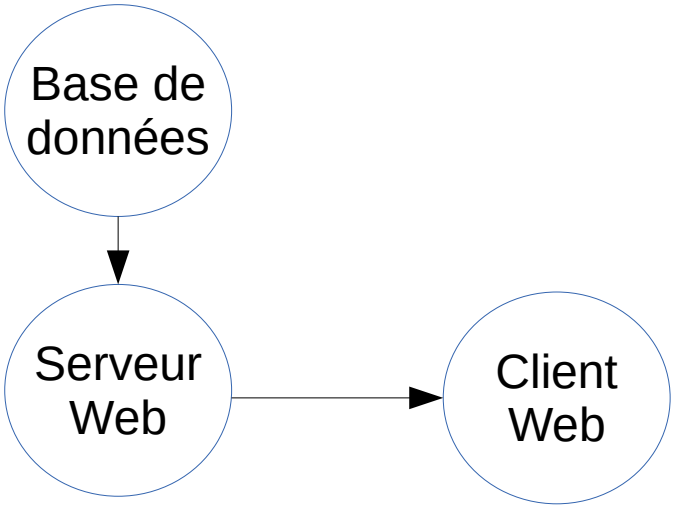
- Rappel des formulaires
- Deux méthodes d'envoi des paramètres :
  - GET : paramètres dans l'URL même
  - POST : paramètres non affichés par le navigateur, envoyés à part, <form ... method="post">
- Utilisation POST :
  - quand l'action ne doit pas être répétée, par ex. achat sur Internet
  - pour des données de grande taille (> 2ko en pratique)
  - d'autres cas spécifiques
- Utilisation GET :
  - dans tous les autres cas
  - avantage : permet de sauvegarder le lien (ADE RT, mes collègues omni) et d'automatiser des tâches (par ex. calcul distances viamichelin pour les OM) !
- Mauvaise utilisation de POST au lieu de GET : [horaire ADE, prix carburant en France, dictionnaire ATILF]
  - utiliser des sessions (sessionID) alors qu'il ne faut pas (parfois c'est fait exprès, par ex. site IEEE)
  - ça ne marche pas chez un autre
  - ça expire au bout d'un certain temps (dico fr atilf)

MySQL – types de données

- Création de la BD et des tables : avec phpmyadmin ou en code PHP/MySQL
- Types de données pour les colonnes : INT (N), VARCHAR (N), DATE, FLOAT(N) etc.
- Colonne AUTO_INCREMENT

PHP/MySQL

- Architecture 3-tier
- Quatre opérations principales SQL sur une base de données (BD) : select, insert, update, delete
- Trois méthodes (API) pour accéder à une BD MySQL :
  - extension MySQL de PHP – vieille, effacée dans PHP 7
  - extension mysqli de PHP – deux styles : procédural et orienté objet ; on utilisera le premier
  - PHP Data Objects (PDO)



PHP/MySQL – insertion, modification et effacement

Connexion à la BD :

```
require ("variables.php");  // initialise HOST, USER, PASS et DB
$ct = mysqli_connect (HOST, USER, PASS, DB) or die ("Unable to connect");
...
mysqli_close ($ct);
```

Insertion d'un enregistrement :

```
$query = "INSERT INTO om VALUES ('$nom', '$prenom', '$age')";
$result = mysqli_query ($ct, $query) or die ("Unable to execute $query: " . mysqli_error ($ct));
```

Modification :

```
$query = "UPDATE om SET nom='Dupond', prenom='Alex', age='20' WHERE id='7'";
$result = mysqli_query ($ct, $query) or die ("Unable to execute $query: " . mysqli_error ($ct));
```

Effacement de données :

```
$query = "DELETE FROM om WHERE nom='Dupond'";
$result = mysqli_query ($ct, $query) or die ("Unable to execute $query: " . mysqli_error ($ct));
```

PHP/MySQL – select

3 manières pour récupérer les données : par indice, par nom (associatif) ou les deux
mysqli_fetch_row ($result) // indicé uniquement
mysqli_fetch_assoc ($result) // associatif uniquement
mysqli_fetch_array ($result, MYSQLI_BOTH); // les deux

Récupération de données :

```
$query = "SELECT * FROM om WHERE nom='Dupond'";
$result = mysqli_query ($ct, $query) or die ("Unable to execute $query: " . mysqli_error ($ct));

// indicé
// $nbFields = mysqli_num_fields ($result);
// $nbLines = mysqli_num_rows ($result);
while ($row=mysqli_fetch_row ($result))
  echo $row[0] . " " . $row[1] . " " . $row[2] . "<br>";

// associatif
while ($row=mysqli_fetch_assoc ($result))
  echo $row['nom'] . " " . $row['prenom'] . " " . $row['age'] . "<br>";

// indicé et associatif
while ($row=mysqli_fetch_array ($result))
  echo $row[0] . " " . $row['prenom'] . " " . $row[2] . "<br>";
```

Scripts côté client

- Exemple : page Web avec le temps qui s'affiche en permanence ?
- Le script s'exécute sur le client (navigateur)
- Interprété, pas compilé

La date courante en php est : <?php echo time(); ?>.
La date courante en php est : 10 mars 2015, 12h20'25".

- Langages : JavaScript (n'a rien à voir avec Java), ActionScript, VBScript etc.
- Nous allons étudier JavaScript, recommandé par W3C
- Bibliographie :
  - tout livre sur JavaScript
  - sites, par ex. http://fr.openclassrooms.com/informatique/cours/tout-sur-le-javascript

Insertion du code JavaScript

- Direct dans la page HTML
  - dans <script>, soit dans le body pour du code à s'exécuter au chargement de la page, soit dans head pour du code à s'exécuter plus tard
  - dans les attributs d'événement (par ex. onmouseover) de toute balise HTML
- Dans un fichier à part, avec <script src="scripts.js"></script>

Utilisation de JavaScript

- Exécution du code : au chargement de la page ou lors d'un événement utilisateur
- Interaction avec le navigateur (BOM) : afficher boîte de dialogue, lecture de l'historique, des paramètres de l'écran etc.
- Interaction avec la page Web (DOM)
  - vérification des champs de formulaires
  - mise en forme de la page
  - modification du contenu de la page

- Cours : http://www.cril.univ-artois.fr/~lecoutre/teaching/web/cours/slidesJavascriptFr.pdf

Canvas

- Le canvas est une région rectangulaire où on peut dessiner des graphiques et des images 2D dynamiquement par des scripts

```
<canvas id="dessin" width="100" height="100"></canvas>

<script>
var example = document.getElementById ("dessin");
var context = example.getContext ("2d");
context.fillStyle = "red";
context.fillRect (0, 0, 100, 100);
context.strokeStyle = "blue";
context.beginPath();
// x centre, y centre, rayon, start angle, fin angle
context.arc (50, 50, 20, 0, 2*Math.PI);
context.stroke ();
</script>
```

Canvas API

- fillStyle, strokeStyle
- createLinearGradient, createRadialGradient
- lineWidth
- fillRect, strokeRect
- beginPath -> moveTo, lineTo, arc -> stroke/fill
- fillText, strokeText
- drawImage

Exemple d'utilisation :
camembert sans utiliser des fichiers image

Comparaison entre scripts côté client et côté serveur

- morpion, calculatrice, convertisseur devises – est-il mieux de les écrire côté serveur ou côté client ou les deux ?
- Visibilité du code : si serveur, code invisible (par ex. mots de passe), si client, code visible
- Le côté serveur limite l'accès du client au serveur (sa BD, ses fichiers, ...)
- Le côté serveur consomme des ressources sur le serveur
- Le côté serveur a besoin d'un serveur Web, i.e. ces pages ne marchent pas en local
- Côté client – sécurité des clients (exécution en sandwich)
- Côté client – il faut faire attention aux incompatibilités
- JS est plus réactif aux changements dues à l'utilisateur, mais la page initiale est plus lourde ; PHP recharge toute la page, alors que JS charge/modifie juste une partie de la page (AJAX)

Technologies (langages) Web pour l'écriture des sites dynamiques

- Classification selon l'endroit où le script s'exécute :
  - sur le serveur : CGI, PHP, JSP, ASP.NET, Node.js, ...
    - bibliothèques/frameworks PHP : symfony, zend, cakephp etc.
  - sur le client : JavaScript, ...
    - frameworks JS : jQuery, ...
    - Bootstrap – le développeur compile son HTML/CSS et y ajoute du JS haut niveau
  - combinaison des deux :
    - AJAX
- Classification selon la spécialisation :
  - jQuery – général
  - Bootstrap – pour affichage et RWD
  - d'autres – pour communication, pour graphisme (e.g. 3D) etc.

Interactions asynchrones client-serveur (AJAX)

- AJAX, Asynchronous JavaScript and XML, est une manière d'utiliser JavaScript dans une page Web pour faire des requêtes au serveur et d'utiliser la réponse dans la page même
- Exemples :
  - affichage en live du score des matchs [sofascore]
  - télécharger et afficher les items d'un menu en fonction du choix de l'utilisateur [123pneus, choisir par véhicule]
  - moteur de recherche qui affiche des complétions au fur et à mesure que l'utilisateur écrit dans la zone de texte [google]
- Le format des données : XML, JSON, HTML, voire texte tout simplement

AJAX, exemple

Script dans la page Web, exécuté sur le client :

```
var req = new XMLHttpRequest ();

req.onreadystatechange = function () {
    // 1=ouvert, 2=en-tête HTTP reçu, 3=contenu commencé, 4=contenu transféré
    if (this.readyState == 4) {
        if (this.status == 200)  // 200 = code HTTP pour transfert avec succès
            alert (this.responseText);  // 'This is the returned text'
        else
            alert ('Erreur : ' + this.status);  // An error occurred during the request
    }
};

req.open ('get', 'ajax.txt');
req.send ();
```

Fichier ajax.txt, sur le serveur :

```
This is the returned text
```

Évolutions possibles :

- le serveur modifie ajax.txt en temps réel et les clients le lisent régulièrement
- utiliser ajax.php et l'appeler avec ?param=..., où param est un paramètre écrit par l'utilisateur

Sécurité MySQL

- Attaque :
  - soit une page avec un champ texte appelé nom pour le nom à chercher
  - le code PHP suivant est vulnérable :

    $req = "SELECT * FROM produits WHERE nom='" . $_GET['nom'] . "'";
  - si l'utilisateur met comme nom :

    x' OR 1='1 => affiche toute la table !
- Solution : en PHP, au lieu d'utiliser $_GET['nom'] utiliser :

  mysqli_real_escape_string (..., $_GET['nom'])
- Autre solution : requêtes préparées

Sécurité JavaScript

- XSS (Cross-Site Scripting)
  - injecter des scripts côté client introduits par un utilisateur dans la page vue par d'autres utilisateurs
- Exemple :
  - un site où les utilisateurs, après login, peuvent ajouter des commentaires
  - M remarque que s'il met comme commentaire **J'aime les fleurs !<script src="...">**, le texte s'affiche et le script **s'exécute**
  - chaque utilisateur qui regarde la page verra le texte, et le script s'exécutera, et de plus cela passe inaperçu
  - le script récupère le cookie d'authentification et l'envoie au serveur de M, permettant donc à M de se faire passer pour l'utilisateur
- Cas réels : bugzilla
- Problème : le navigateur/serveur exécute le **code** d'un utilisateur dans le contexte d'un **autre** utilisateur
- Solution : « escape » les caractères spéciaux : <, >, ", &, ', mais aussi d'autres, dans toutes les entrées des utilisateurs (y compris l'URL)

Sécurité des pages Web, dans PHP

- Ne jamais faire confiance aux valeurs reçues de l'utilisateur :
  - vérifier toujours ces valeurs avant utilisation, par ex. entier entre 2 et 6
  - pour afficher une valeur : utiliser strip_tags pour enlever les balises, htmlspecialchars et htmlentities pour remplacer les caractères spéciaux
  - utiliser preg_match pour accepter que les chaînes avec certains caractères seulement
- D'où viennent les valeurs des utilisateurs ? Cela dépend de votre page :
  - paramètres de formulaires (GET ou POST)
    - attention : même si un combobox est utilisé ou une vérification JS est faite, l'utilisateur peut modifier lui-même l'URL, par ex. : ...?name=Jean
  - BD, cookies, autres pages Web etc.
- Où le problème peut-il se manifester ?
  - chez les autres utilisateurs (capture de leurs mots de passe)
  - sur le serveur (effacement BD)

Complexité de la sécurisation des sites

- La sécurisation, d'un mot de passe par ex., apparaît à plusieurs endroits :
  - **l'utilisateur** prend soin de son mot de passe : choix d'un mot de passe complexe, ne pas l'écrire sur un papier, ne pas le taper quand un autre est à côté etc.
  - **le client** : le navigateur ne stocke pas les mots de passe en clair dans un endroit publique, pas de virus qui enregistre le clavier etc.
  - **le serveur** ne divulgue pas le mot de passe : ne pas tester la validité du le mot de passe dans la page (par ex. <script if pwd == "xyz" ...>), mais sur le serveur
    - aussi, le serveur doit stocker les mots de passe cryptés, sinon un intrus dans le serveur peut voir tous les mots de passe (imaginez la gravité de la situation pour le site d'une banque)
  - **le réseau** : si un utilisateur introduit son mot de passe sur un site non sécurisé, il circule en clair sur le réseau (navigateur -> serveur), donc un administrateur réseau peut le voir ; solution : crypter la communication, https
- Nous allons présenter le cryptage de la communication (https) seulement

HTTPS, TLS

- TLS se met entre le protocole de transport et l'application
- TLS fournit :
  - authentification du serveur : par infrastructure à clés publiques
  - cryptage et intégrité des données : par chiffrement symétrique
- Applications :
  - HTTPS = HTTP Secure = HTTP sur TLS
    - configurer le serveur Web pour utiliser HTTPS, port 443 par défaut [onlamp]
  - programmes de courriel, visioconférence (VoIP), IM etc.
- Tout navigateur a une liste de CA, voir Preferences->Privacy->Certificates
- Quand il contacte un site S, il vérifie que le certificat qui lui est donné est le même que le certificat de S donné par la CA
- Plus d'informations : [wikipedia]

CMS

- Un CMS (Content Management System) permet d'éditer de manière simple des pages Web à partir d'une interface (Web) au lieu d'un programme
  - exemple avec wikipedia
- Couramment utilisés dans les blogs, serveur de news, vente en ligne etc.
- Caractéristiques usuelles : gestion des utilisateurs, rechercher un mot dans le site, stocker les versions anciennes (revision control), multilangue etc.
- Les pages Web sont généralement stockées dans une BD
- Exemples : WordPress, Joomla, Drupal, MediaWiki