Compléments au cours de programmation web

A.-L. Ligozat
ENSIIE

dernière màj: mai 2017

Outils de développement

CMS (*Content Management Systems* ou Systèmes de gestion de contenu)

Objectif : création et mise à jour de sites web sans nécessité d'expertise informatique

- exemples : Joomla, Spip, Drupal, Wordpress

Frameworks web

Objectif : faciliter le développement de sites web en proposant librairies (gestion des utilisateurs, accès aux données...)

- exemples : Symfony, Angular JS, Laravel, Django, Ruby on Rails, CodeIgniter

CMS en détails

Fonctionnalités

- gestion des utilisateurs et de leurs droits (notamment administrateurs vs utilisateurs)
- édition de pages WYSIWYG
- gestion des contenus multimédia
- modularité et extensibilité (plugins, tels que calendrier, nuage de tags...)
- statistiques
- référencement
- modèles pour design

CMS : avantages et inconvénients

Avantages

- limite le besoin de compétences informatiques (création et màj)
- sécurité intégrée
- homogénéité du site
- séparation du contenu (BD) et de la forme (CSS)
- conformité aux standards comme (HTML5, WAI-ARIA...)
- contrôle de version

Inconvénients

- lourdeur de la prise en main
- manque d'adaptabilité
- performance
- maintenance (màj)

CMS : quelques exemples

Joomla

- créé en 2005 ; version actuelle = 3.6
- PHP, PostgreSQL/MySQL/MSSQL/SQLite

Wordpress

- créé en 2003 ; version actuelle = 4.7
- PHP, MySQL

Drupal

- créé en 2001 ; version actuelle = 8.2.6
- PHP (dont librairies Symfony), PDO

CMS : quels sites ?

facilité d'utilisation : Wordpress $>$ Joomla $>$ Drupal
facilité de personnalisation : Wordpress $<$ Joomla $<$ Drupal

Exemples de sites

- ENSIIE : SPIP
- LIMSI : Joomla
- Université Paris-Saclay : Drupal
- sites WordPress

Frameworks en détails

Framework (cadriciel) ?

- boîte à outils logicielle
  - visée générique
  - architecture logicielle, patrons de conception

Dans le cadre web

- Objectif = création d'applications web
- destiné aux développeurs $\neq$ CMS
  - librairies pour fonctionnalités standard (authentification des utilisateurs, connexion aux bases de données, templates, pagination, envoi d'emails, sessions, droits, formulaires...)
- MVC
- POO
- inversion de contrôle (patron d'architexture) : le flot d'exécution d'un logiciel est dicté par le framework et non plus par le développeur

Frameworks : fonctionnalités courantes

- templates (gabarits) : fichiers html
- URL mapping : routage (expressions régulières ou réécriture d'URL) qui autorise à utiliser des URL plus conviviales
- cache
- accès au données, par exemple avec ORM (*Object Relationnal Mapping*)
- scaffolding (échaffaudage) : génération du code d'accès aux données
- conventions

Frameworks

Avantages

- organisation du code évolutive et facile à maintenir
- utilisation de composants existants robustes
- gain de temps de développement, surtout sur les aspects génériques de l'application
- partage du développement facilité
- communauté
- standards de programmation

Inconvénients

- courbe d'apprentissage
- choix du framework

Frameworks : différents types

- côté serveur : tout langage (PHP, Python, JS) – Django, Zend, Symfony...
- côté client : (single page applications) AngularJS, EmberJS...
- push-based (ou action-based) : traitements nécessaires, puis les données sont envoyées (push) à la partie vue pour affichage
  - ex : Django, Ruby on Rails, Symfony, CodeIgniter...
- pull-based (ou component-based) : partie principale = vue, qui fait appel (pull) aux contrôleurs pour obtenir les données nécessaires

Créer du CSS

- pré-processeurs générant du CSS (SASS, LESS)
- frameworks front-end responsive comme Bootstrap