10. Frameworks

M1 Outils de l'Internet
lundi 6 décembre 2010

victor.poupet@lif.univ-mrs.fr

Présentation

Qu'est-ce que c'est ?

- Environnement de travail
- Développer des sites dynamiques
- Développer des applications web
- Librairies, modèles, réutilisation de code...

Évolution

- Web statique : modifications à la main sur chaque page
- CGI permet la génération dynamique de contenu
- Apparition de langages spécialement pensés pour le web (PHP, ColdFusion, ASP)
- De nombreuses librairies utilisées pour le développement d'appli webs → frameworks

Architecture

- Model View Controller
- Deux principales façons de faire interagir le contrôle et la vue : *Push* et *Pull*.

Push

- Le contrôleur effectue une action et cette action modifie la vue
- Exemple : envoyer un message ou déclencher une action
- Frameworks : Django, Ruby on Rails

Pull

- La vue fait appel au contrôleur pour obtenir les informations qu'elle doit présenter
- Exemple : représentation d'informations contenues dans la BDD
- Frameworks : Struts2, Tapestry

Fonctionalités

Sécurité

- Authentification
- Autorisation
- Porte sur le contenu et les actions
- Ex. : Django utilise des *rôles* pour répartir les droits

Bases de données

- API unifiée pour communiquer avec différents formats de BDD
- Même code si on change de BDD
- Possibilité de grouper les échanges avec la BDD pour plus d'efficacité

Bases de données

- Gestion de *transactions*
- Outils de migration de données
- Object-Relational Mapping

Object-Relational Mapping

- ORM, O/RM, O/R mapping
- Conversion d'un objet complexe en un ensemble d'entrées dans une BDD ne contenant que des types simples
- Alternative : utiliser des BDD orientées objet (par exemple des BDD XML)

Transactions

- Objectif : maintenir l'intégrité des données
- Solution : gérer plusieurs opérations comme une opération indivisible
- Tester l'ensemble des opérations, puis les valider si aucune erreur

Structure des URL

- Utilisation d'expressions régulières et de conversions d'URL
- Permet d'alléger les URL

  ex. : `page.cgi?nom=guybrush&item=ship`

  → `page/guybrush/ship/`

Gabarits

- En anglais *Templates*
- La partie statique d'une page
- Description d'une page avec les emplacements des données (mais sans les données)
- Fait partie de la vue (donc pas de vraie programmation)

Cache

- Conserver des données qui seront réutilisées plus tard
- Certains frameworks permettent de gérer le fonctionnement du cache en séparant les données communes (p.ex. les *templates*)

Configuration automatique

- Exécution automatique de tâches courantes
- *Convention over configuration*
- Exemple : génération automatique de la structure de la BDD lors de l'exécution
- Utilisation de conventions de nommage

AJAX

- Asynchronous JavaScript and XML
- Il est assez compliqué d'écrire entièrement des pages en AJAX
- Il existe des frameworks spécialement pour AJAX
- Les frameworks gèrent automatiquement les communications AJAX en fonction des structures des objets

Langages

JAVA

- Utilisent en général Java EE
  - Spring Framework
  - Apache Struts
  - Apache Struts 2
  - Google Guice
  - Induction
  - Stripes
  - Tapestry
  - Hivemind

C# et VB.NET

- Langages utilisés avec ASP.NET (Microsoft)

Python

- Django
- Zope
- TurboGears
- Quixote
- Pylons
- web2py

Perl

- Catalyst
- Mojolicious

PHP

- Symfony
- CakePHP
- Zend Framework

Ruby

- Ruby on Rails
- Merb
- Sinatra

Principes et philosophies

Don't Repeat Yourself (DRY)

- Idée : Réduire au maximum les répétitions d'informations
  - Chaque information ne doit apparaître qu'une seule fois
  - Code source
  - Base de données

DRY

- Permet de maintenir la cohérence d'un système de grande taille
- Facilite la modification des données
- Présenté par Andy Hunt et Dave Thomas dans leur livre *The Pragmatic Programmer*.

"Every piece of knowledge must have a single, unambiguous, authoritative representation within a system."

Limites de DRY

- Ne peut pas être automatisé (certaines informations sont égales mais pourraient devenir différentes)
- Application stricte du concept peut-être plus coûteux que les bénéfices
- Difficile à gérer dans les systèmes communautaires

Limites de DRY

- Informations générées automatiquement peuvent être dupliquées sans que ce soit un problème
- Les textes pour les humains contiennent des redondances et ne peuvent pas toujours être générés automatiquement
- Certains langages de programmation ne permettent pas la factorisation facilement

Principe d'abstraction

- Forme plus général de DRY
- Abstraire les éléments communs de différents objets
- Utilisation des possibilités des langages de programmation : objets/classes, fonctions de haut niveau, etc.

Principe d'abstraction

- Benjamin C. Pierce *Types and Programming Languages* (2002) :

Each significant piece of functionality in a program should be implemented in just one place in the source code. Where similar functions are carried out by distinct pieces of code, it is generally beneficial to combine them into one by abstracting out the varying parts.

Principe d'abstraction

- David A. Schmidt *The structure of typed programming languages* (1994) :

The phrases of any semantically meaningful syntactic class may be named.

Convention over Configuration

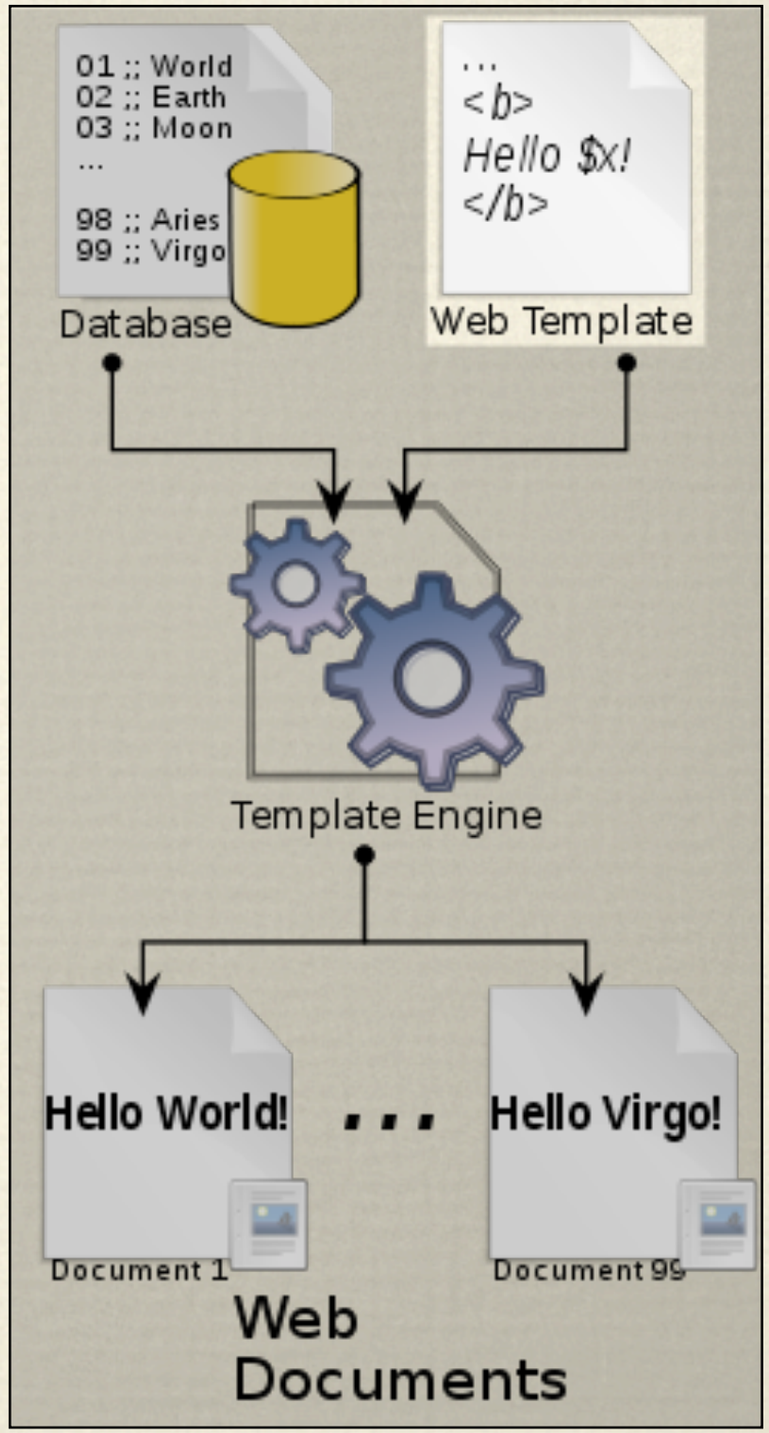
- Également *Coding by Convention*
- Idée : réduire le nombre de choix à spécifier par les programmeurs tout en conservant l'ensemble des possibilités
- Seul ce qui n'est pas conventionnel doit être décrit
- On ne perd pas en flexibilité

Convention over Configuration

- Les fichiers de configuration sont bien plus simples
- Exemples de frameworks :
  Spring, Ruby on Rails, CakePHP, Zend Framework
- Classes → Tables en BDD (attributs → colonnes)

Gabarits

- Séparer les données de la présentation
- Similaire au principe des formulaires (papiers)
- Très fréquemment utilisés pour la génération automatique de pages



Gabarits

- Développer des applications faciles à modifier
- Séparation de la vue et des données
- Principes flous, difficile à formaliser entièrement
- Les gabarits peuvent être récursifs
- Utilisation de la notion d'héritage

Quelques Exemples

Django

- Écrit en *Python*
- *Open Source*
- Tire son nom de Django Reinhardt

Django : composants

- Object Relational Mapper (classes Python → BDD)
- Traitement des URL par expressions régulières
- Système de *templates*

Django : fonctionnalités

- Serveur web pour tester
- Système de validation des entrées
- Gestion des caches
- Système de *sérialisation* en XML ou JSON
- Le système de templates est facilement extensible

Django : fonctionnalités

- Système d'authentification extensible
- Interface dynamique d'administration
- Gestion de flux RSS et Atom

Django : compatibilité

- *Apache* avec `mod_python` (ou `mod_wsgi`)
- FastCGI
- PostgreSQL, MySQL, SQLite et Oracle

Ruby on Rails

- Programmé en Ruby
- Basé sur le projet *Basecamp* de *37signals*
- Publié en open source en 2004
- Inclus dans Mac OS X 10.5 (2007)

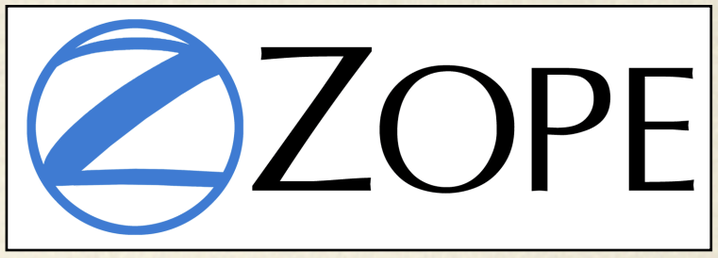
RoR : fonctionalités

- Fonctionne par échafaudage (*scaffolding*) : spécification de l'utilisation de la BDD permet de produire une base d'application (MVC)
- Serveur web en Ruby (WEBrick)
- Fonctionne avec *Apache* (*Passenger*, *FastCGI*, `mod_ruby`)
- Moteur de production : *Rake*

RoR : philosophie

- Convention over Configuration
- Don't Repeat Yourself

Zope



- *Z Object Publishing Environment*
- Open source
- Orienté objet
- Écrit en Python
- Interface web

Zope : fonctionnalités

- Objets dans une BDD (pas de fichiers)
- Encapsulation facile
- Conversion URL → objets
- Utilise la *Zope Objects Database*

What a long, strange trip it's been...

Internet : vue d'ensemble

- Matériel
- Encodage/Formats
- Protocoles
- Langages
- Applications
- Sécurité

Matériel

- Vu rapidement au premier cours
- Voir avec Emmanuel...

Encodage

- Beaucoup de formats sur Internet
- Les normes évoluent
- Réutilisation de normes (ex. MIME, URI)

Protocoles

- DNS, mail, web, sécurité, etc.
- Différentes structures possibles (P2P, client-serveur, Radius, mail, etc.)

Langages

- Web : *ML, CSS, JS, PHP, Python, LDAP, etc.
- Liés aux protocoles, applications et à l'architecture (p.ex. MVC)

Sécurité

- À tous les niveaux
  - physique
  - réseau
  - protocoles
  - langages
  - applications
  - humain

Applications

- S'appuient sur les autres points
- De plus en plus diversifiées
- Motivent l'évolution des formats et protocoles
- Applications web ou locales (avec fonctionnalités en ligne)