



Frameworks Web next- generation



Frameworks web next-gen

- **Introduction**
 - La vision Old-School
 - Le web actuel
- **Frameworks classiques**
 - Principes fondateurs
 - Vulnérabilités & Faiblesses
 - Problématique



Frameworks web next-gen

- **Frameworks "next-gen"**
 - Dérivation des langages scriptés
 - Intégration aux serveurs actuels
 - Optique de développement et sécurisation
- **Frameworks reconnus**
 - Django
 - Zope/Plone
 - Ruby on Rails



Frameworks web next-gen

- **CaT: framework web python**
 - Présentation
 - Fonctionnalités
 - Exemples
- **Conclusion**
 - Avantages
 - Inconvénients
 - Mot de la fin



Frameworks web next-gen

Introduction





- **Framework web:**
 - Environnement de travail permettant le développement et l'intégration d'applications web
 - Ensemble de logiciels permettant la conception, le développement et la publication d'applications web
 - Par extension, les différents éléments architecturaux du serveur d'hébergement



- **La vision "Old-School":**
 - Le serveur web offre un ensemble de ressources
 - Chaque ressource dynamique est un programme unique, exécuté par le serveur
 - Le visiteur se fraye un chemin dans les ressources disponibles

RESPECT
THE OLD SCHOOL





- **Le web actuel:**

- C'est le web 2.0 de M. Lefebvre !
- Intégration de la technologie AJAX (Javascript + XMLHttpRequest)
- Transformation du navigateur en interface graphique applicative multi-usage
- Rafraîchissement zoné et asynchrone (basé sur AJAX)
- Externalisation des applicatifs métiers
- Apparition de services web, d'APIs, ...





- **Problématiques:**
 - Est-ce que les frameworks web classiques conviennent aux applications actuelles ?
 - Y-a-t-il un moyen d'améliorer le quotidien des développeurs ?
 - Peut-on proposer aux internautes des sites à haute interactivité (HI), basés sur ces technologies ?



Frameworks classiques



Frameworks classiques

- **Frameworks "classiques":**
 - Des frameworks employés depuis le début des serveurs web (ou presque),
 - Des frameworks limités, qui n'ont pas pris en compte l'évolution des technologies
- **Exemples:**
 - Apache 2.0 / PHP5
 - IIS 6.0 / ASP .Net



Frameworks classiques

- **Principes fondateurs:**
 - La racine du serveur web correspond à un emplacement physique réel
 - Le serveur fait l'abstraction entre la ressource web (URI) et la ressource physique (fichier)
 - Le serveur réagit différemment selon le type de fichier



Frameworks classiques

- **Vulnérabilités**

- **Le serveur accède aux ressources sans validation de l'application:**

- Attaques par force brute sur les fichiers et répertoires
- Fuites d'informations



- **Les scripts sont interprétés comme des programmes:**

- Inclusion de fichier local (faille « include », ...)
- Lecture/écriture dans le répertoire racine



Frameworks classiques

- **Faiblesses**

- Pas de système de cache natif
- Pas beaucoup de fonctionnalités en standard (PHP par exemple)
- Trop rigide (nommage des ressources, type des fichiers)
- Trop d'exposition (fichiers et répertoires)



Frameworks classiques

- **Problématique:**
 - Fuites d'informations: extensions, version du serveur, fichiers accessibles, ...
 - Exposition de l'arborescence: le serveur fait office de couche d'abstraction
 - L'application passe au second plan



Frameworks next-gen





- **Dérivation des langages scriptés**
 - Les langages comme python ou ruby possèdent une bibliothèque standard fournie
 - Ces langages sont multi-plateformes
 - Ils sont majoritairement orientés objet
 - Ils respectent le modèle MVC (Modèle/View/Contrôleur)



- **Intégration aux serveurs actuels**
 - Les frameworks next-gen peuvent être placés comme extensions des serveurs standards:
 - mod_python (Python + Apache)
 - Urlrewrite (RoR + Apache)
 - Ils peuvent intégrer un serveur applicatif:
 - WEBrick (RoR)
 - Mise en cache des applicatifs pseudo-compilés



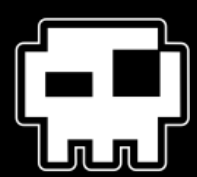
- **Intégration (suite)**
 - L'application supplante le serveur classique:
 - Gestion des accès aux ressources
 - Nommage paramétrable
 - Répertoire racine virtuel
 - L'application est uniforme
 - Pas de scripts éparses
 - Respect du modèle MVC
 - Applicatif standard, méthodes de développement standard



- **Optique de développement et de sécurisation**
 - **Plus de sécurité: L'application filtre les accès aux ressources, les ressources peuvent être purement virtuelles.**
 - **Cloisonnement plus stricte**
 - **Développement facilité**
 - Tout est concentré dans un applicatif unique
 - Modélisation UML possible, et implémentation type objet



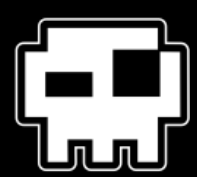
Frameworks reconnus



Frameworks reconnus:

- **Django**

- Basé sur python
- Implémentation du modèle MVC
- Gestion multi-vhosts
- Intègre tous les outils de gestion de base de données
- Interface d'administration générée automatiquement



Frameworks reconnus:

- **Zope/Plone**

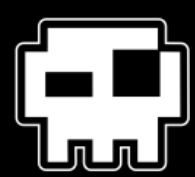
- Basé sur python
- Serveur applicatif dédié
- Très bonne communauté
- Employé par un grand nombres de sociétés



Frameworks reconnus:

- **Ruby on Rails (RoR)**
 - Basé sur ruby
 - Très bonne communauté
 - Serveur applicatif dédié (WEBrick)
 - Gestion multi-vhosts
 - Implémentation selon le modèle MVC





CaT: Framework python



CaT: Framework python

- **Présentation**

- Framework léger et évolutif
- Fonctionne avec mod_python sur Apache
- Etend les fonctionnalités Apache
- Intégration des Virtual Hosts
- Emploie un handler Python strict
- Implémente le système de templates de Django





CaT: Framework python

- **Fonctionnalités**
 - **Arborescence purement virtuelle: le framework crée une arborescence virtuelle, dans laquelle sont mappées les ressources accessibles**
 - **Les ressources sont liées implicitement à des méthodes et des classes**
 - **Possibilité de servir des fichiers médias, mais référencés dans l'arborescence virtuelle**



- **Exemple d'implémentation**
 - **Scripts côté serveur: manipulation de BDD via un CRUD**
 - **Scripts côté navigateur: Pur AJAX (Interface dynamique)**
 - **L'interface AJAX utilise les ressources mises à disposition pour manipuler la BDD**
 - **Le serveur web se transforme en service web**



- **Avantages de l'implémentation**
 - **Peu de templates à gérer:**
 - Le serveur implémente la logique (Modèle/Controlleur)
 - Le navigateur utilise cette logique, et conçoit l'interface (Vue/Controlleur)
 - **On transforme le navigateur en pure application web**
 - **La sécurité du serveur est accrue grâce au framework**



Conclusion





- **Avantages:**

- Les frameworks web next-gen réduise le champ des vulnérabilités
- Ils permettent un développement plus rapide, et concentré
- Ils autorisent une implémentation de type MVC

- **Inconvénients**

- Ils ne suppriment pas toutes les vulnérabilités existantes (XSS, SQL Injection, ...)
- Ils nécessitent un temps d'adaptation



Questions ?



- **Sources:**

- Wikipédia: Ruby-on-Rails, Apache

- Site officiel mod_python:
<http://www.modpython.org>

- Site officiel Rails:
<http://www.rubyonrails.org>

- **Remerciements divers:**

- CrashFr, 4ine, Dvrasp