Framework PrototypePHP

Comment l'utiliser ?

Éric Quinton

2 septembre 2016

Table des matières

Chapitre 1

Présentation

Historique

Au début des années 2000, PHP commençait à être largement utilisé pour créer des applications web. Certains frameworks étaient déjà présents, mais ils présentaient souvent des difficultés pour les appréhender et n'étaient pas forcément adaptés aux besoins de l'époque (performance souvent insuffisante en raison d'un chargement systématique de toutes les classes, fonctionnement exclusivement objet, etc.). De plus, ils ne permettaient que difficilement de remplacer certains composants par d'autres.

Des outils comme Smarty, un moteur de templates qui permet de séparer le code HTML du code PHP commençaient à se faire une place. On trouvait également des bibliothèques assez élaborées comme PHPGACL pour gérer les droits de manière particulièrement pertinente.

La gestion des bases de données n'était pas des plus optimales, et un souvent un peu trop conceptuelle.

PrototypePHP a été créé pour assembler divers outils disponibles, selon la conception qu'en avait l'auteur à l'époque. Il était loin d'être parfait et a évolué de multiples fois, pour intégrer une approche mvc, puis des contraintes de sécurité, etc. Toutefois, les fondements de départ sont restés quasiment identiques, même si certaines évolutions ont été intégrées :

- des actions décrites dans un fichier xml, qui est utilisé pour générer le menu en fonction des droits détenus par l'utilisateur ;
- une gestion des droits basée sur PHPGACL. Si le produit initial a été abandonné, sa philosophie a été conservée ;
- une séparation du code PHP et HTML avec l'utilisation de SMARTY ;
- un accès aux tables de la base de données réalisé par l'intermédiaire d'une classe dédiée à cet usage, ObjetBDD, qui contient des fonctions très simples à manipuler, comme ecrire($data), lire($id), supprimer($id). La connexion à la base de données, à l'époque réalisée en utilisant la bibliothèque ADODB, a été remplacée par PDO ;
- un support de l'identification selon quatre modalités : base de données, annuaire LDAP, annuaire LDAP puis base de données, et connexion via un serveur CAS ;
- un souci permanent de la performance, lié au passé de son concepteur [1].

La première version publiée l'a été en 2008, dans sourceforge (`https://sourceforge.net/projects/prototypephp/`). Depuis quelques années, elle est disponible dans github (`https://github.com/equinton/prototypephp`), la branche active étant la branche *bootstrap*, créée au moment du basculement de l'affichage en utilisant les fonctionnalités de ce produit.

Si le principe général d'une conception MVC a prévalu depuis plusieurs années, des améliorations récentes, notamment dans la gestion des vues, a été apportée. À partir de septembre 2016, une

1. il a commencé sa carrière à une époque où les ressources informatiques étaient rares, chères, et dont la puissance était limitée

meilleure gestion des droits a été implémentée, notamment dans les contextes de travail avec un annuaire d'entreprise LDAP. Il n'est pas impossible également que le support de Shibboleth puisse être intégré dans le futur, notamment quand des bibliothèques prêtes à l'emploi seront disponibles.

Gestion des versions

Le framework est mis à jour en parallèle aux développements de logiciels bâtis à partir de celui-ci. Le code disponible reflète donc les retranscriptions des modifications apportées au gré des évolutions envisagées par son concepteur.

Il n'existe ainsi plus depuis plusieurs mois de gestion de version : le plus simple est de se référer à la date du commit, en utilisant la branche *bootstrap*, qui est celle de travail actuel.

plugins utilisés

Les bibliothèques suivantes sont installées dans le framework :

- pour le code PHP :
  - ObjetBDD (conçu par le développeur du framework), qui gère l'interface avec la base de données ;
  - SMARTY (`http://www.smarty.net`), le moteur de templates ;
  - phpCAS (`https://wiki.jasig.org/display/CASC/phpCAS`), pour la connexion par l'intermédiaire d'un serveur CAS ;
  - et d'autres bibliothèques disponibles dans le framework, mais utilisées uniquement si nécessaire, comme tcpdf (`https://sourceforge.net/projects/tcpdf/files/`), odtphp (`https://sourceforge.net/projects/odtphp/`), openoffice_generation, phpExcelReader (`sourceforge.net/projects/phpexcelreader/`)...
- pour l'affichage et la conception des pages web, le recours au javascript est omniprésent :
  - JQuery, JQueryUI, et des plugins pour les sélections des dates ;
  - DataTables et ses plugins ;
  - OpenLayers pour l'affichage des cartes ;
  - bootstrap pour la prise en compte de l'affichage sur le mode *responsive* ;
  - ...

Elles sont mises à jour régulièrement, mais il est préférable de vérifier si de nouvelles versions sont disponibles avant de procéder à une mise en production.

Modèle MVC

Le framework est basé sur un modèle MVC, qui présente les caractéristiques suivantes :

- le contrôleur est unique, les actions et les droits associés sont décrits dans un fichier unique ;
- les vues sont héritées d'une classe non instanciable, avec des classes dédiées à l'usage (html via Smarty, ajax, csv pour le moment) ;
- le modèle est constitué de deux types d'objets : des classes héritées d'ObjetBDD pour gérer les échanges avec la base de données, et des fichiers de script exécutant les modules (ou actions) demandés.

Le framework n'a pas une philosophie « tout objet », comme peuvent l'être d'autres, pour tirer parti de la souplesse du php. De nombreuses fonctions permettent de faciliter et limiter le code à écrire.

Quelques classes génériques sont utilisées (une classe Message, par exemple), et l'application recourt fortement aux variables de session.

Terminologie

Dans le framework, l'appel à une action, depuis le navigateur, se nomme un module. Un module se caractérise par un fichier appelé, des droits nécessaires à son exécution, et une action particulière (list, display, change...) qui est traitée spécifiquement dans le fichier appelé.

Cette approche présente des avantages de clarté au niveau du code (relativement peu de fichiers), mais aussi une moindre visibilité au niveau des éditeurs comme Eclipse : comme les actions ne sont pas déclarées comme des fonctions de classe, la navigation dans le code est parfois une peu plus complexe.

Chapitre 2

Le contrôleur

Chapitre 3

Le modèle

L'accès aux données : la classe ObjetBDD

Les modules exécutés