

# Zend Framework - Le front Contrôleur

par JYT ([Les expériences Zend de Sekaijin](#)) (Blog)

Date de publication : 22 octobre 2007

Dernière mise à jour :

J'ai abordé précédemment la problématique du **BootStrap** et de la **configuration**. Pour la configuration, l'essentiel est fait. Il sera peut être nécessaire d'adapter un peu la chose mais la structure est suffisamment solide pour supporter de telles évolutions. Quant au BootStrap, il est resté aussi simple que je le souhaitais. Facile me direz-vous, j'ai botté en touche. J'ai donc un démarrage simplifié, un chargeur de configuration, mais dans tout ça rien qui dise à mon application comment l'utiliser.

- I - Le chef d'orchestre
- II - Un exemple

## I - Le chef d'orchestre

Le front contrôleur est un élément central dans le modèle MVC. C'est lui le grand chef d'orchestre. Son rôle est de distribuer le travail. Il va préparer le travail de l'application, distribuer les activités, vérifier la cohérence du tout et s'assurer que la demande aboutit bien à une réponse correcte. En tant que chef d'orchestre, il est le mieux placé pour prendre en compte les paramètres de configuration générale et préparer les éléments nécessaires pour les rendre accessibles à l'application.

Le front contrôleur de Zend\_Framework est particulièrement adaptable. Mais pour cela, il est nécessaire d'écrire beaucoup de code, ce que je cherche justement à éviter. Je pourrais écrire mon propre front contrôleur comme je l'ai fait par le passé en PHP4. Mais celui de ZF est plutôt bien fait et je n'ai pas envie de m'en priver. Comme toujours, la programmation orientée objet permet d'adapter une classe à mon besoin. Un petit retour sur le BootStrap vous montrera que, finalement, mon front contrôleur est simplement un Zend\_Controller\_Front. Il ne fera rien d'autre que tenir compte de la configuration avant de lancer l'application.

Tout ce que vous trouvez comme paramétrage du contrôleur dans le BootStrap, s'y trouve parce que dépendant de l'application. L'application peut être chargée automatiquement dans la méthode **run**, il suffit pour cela de définir un attribut de paramétrage.

## II - Un exemple

Par exemple l'authentification. Vos applications utilisent une authentification des utilisateurs, d'autres pas. Vous pouvez placer un paramètre **auth** à **true** ou **false** dans le fichier de paramètres. Et dans la méthode **run**, en fonction de sa valeur, utiliser un **Zend\_Auth** : il est nécessaire pour cela d'avoir les paramètres du système d'authentification. Ajoutons alors une section **auth** dans le fichier de configuration.

La méthode run ressemblera alors à ceci :

```
public static function run($controllerDirectory)
{
    // choix du moteur de
    $config = Fast_Registry::getConfiguration();
    $parameters = Fast_Registry::getParameters();
    if ($parameters&&$config) {
        $controller = self::getInstance();
        $controller->throwExceptions(true);
        $controller->setControllerDirectory($controllerDirectory);
        $controller->setRequest('Fast_Controller_Request_Http');
        $controller->_setDispatcher('Fast_Controller_Dispatcher');

        Zend_Loader::loadClass('Zend_Session');
        Zend_Session::start();

        $auth = $parameters->fast->get('auth', false);
        if($auth) {
            Zend_Loader::loadClass('Zend_Db_Table');
            Zend_Loader::loadClass('Zend_Auth');
            if (!$config->get($parameters->fast->db)) {
                Zend_Loader::loadClass('Fast_Exception_Db');
                throw new Fast_Exception_Db('No configuration found for database:
' . $parameters->fast->db);
            }
            $params = $config->get($parameters->fast->db->toArray();
            // connexion
            $dbAdapter = Zend_Db::factory($parameters->fast->db, $params);
            Zend_Db_Table::setDefaultAdapter($dbAdapter);
            Zend_Registry::set('dbAdapter', $dbAdapter);
        }
    }
}
```

etc.

Encore une fois, dès qu'une nouvelle option du front contrôleur est utilisée par une application, la reporter ici permet de la mettre à disposition de toutes les applications futures.

Vous trouverez dans mon code des classes dont je n'ai pas encore évoqué la fonctionnalité. Supprimez les lignes ou remplacez-les par les classes Zend correspondantes. Je reviendrai dessus au fur et à mesure. Elles sont simplement la preuve que l'on peut capitaliser beaucoup de développements tout en gardant une facilité de mise en #uvre.

[Source](#) **Fast\_Controller\_Front**

A+JYT

