

FAQ Zend Framework

Date de publication : 05 mai 2007

Dernière mise à jour : 20 avril 2009

Les questions redondantes se multipliant sur le forum **Zend Framework**, nous avons choisi de créer cette FAQ afin de regrouper les questions qui sont posées fréquemment.

Vos remarques et idées sont les bienvenues. Et pourquoi pas **votre aide** ! S'il vous prend l'envie de tailler votre plus belle plume et de retrousser vos manches, il y a de quoi faire !

Ont contribué à cette FAQ :


Guillaume Rossolini ([Tutoriels Web / SEO / PHP](#)) ([Blog](#)) - Matthew Weier O'Phinney - Janitrix - stalak - TheDrev - arnoweb - Rob Allen - byc_r - j.roc - Julien Pauli ([Cours, articles et tutoriaux PHP](#)) ([Blog](#)) - Reveur - websurfeur - vg33 - Yoteco - lecra - Gérard Ernaelsten ([Site perso](#)) -




1. Généralités (8)	4
2. Zend_Controller (4)	9
3. Zend_Db (15)	12
3.1. Zend_Db_Select (5)	18
4. Zend_Debug (2)	20
5. Zend_Form (2)	21
6. Zend_Layout (1)	23
7. Zend_Session (1)	24
8. Zend_Translate (3)	25
9. Zend_View (4)	27

[Sommaire > Généralités](#)

Qu'est-ce que ZF ?

Auteurs : [Guillaume Rossolini](#) ,

Zend Framework est un  **framework** de développement d'applications PHP. Il se situe au même niveau que symfony, CakePHP et d'autres frameworks. C'est un framework de type "glue" avec des tendances "full-stack".

Zend Framework est résolument orienté vers un développement à base de PHP5 et de ses capacités Objet. L'équipe de développement en construit tous les composants selon une procédure qualité exemplaire (validations, standards de codage, documentation, tests...). Les développeurs utilisent des  **design patterns** où leur utilisation est judicieuse (décision collégiale), par exemple  **MVC** (Zend_Controller),  **Singleton** (Zend_Registry) *etc.*

Pour contribuer à ZF, il faut signer un CLA (Contributor Licence Agreement) qui permet de protéger les contributions de tous les auteurs.

Pour utiliser Zend Framework, il faut avoir au moins PHP 5.1 et il est fortement recommandé d'avoir la main sur la configuration du serveur Web (serveur dédié).

Quelle arborescence adopter pour mon application ?

Auteurs : [Janitrix](#) ,

Le Zend Framework est souple sur l'arborescence des applications qui l'utilisent. Cependant, certaines solutions sont plus ou moins recommandées et recommandables.

La structure basique, que l'on utilise généralement pour débiter, est celle-ci :

```
/application // Comporte les fichiers relatifs à votre application
/config // Les fichiers de configuration de votre application
/controllers // Les contrôleurs de votre application
/models // Les modèles de votre application
/views // Les vues (.phtml)
/library // Les bibliothèques externes que votre application utilise
  /Zend // La bibliothèque du Zend Framework bien sûr
  /(autres bibliothèques)// Éventuellement d'autres bibliothèques
/public // Les ressources "publiques" de votre application
  /images // Les images
  /scripts // Les scripts (javascript)
  /styles // Les feuilles de style (CSS)
/tmp // Un dossier pour les données temporaires que va utiliser votre appli.
  /sessions // Les sessions PHP éventuellement
  /cache // Le cache pour les requêtes SQL ou pour les vues
  /view_compiles // Les vues compilés (pour un moteur de template par ex.)
index.php // Le fichier bootstrap de votre application
```

Bien que cette arborescence soit simple pour les petites applications, cela devient compliquer à gérer lorsque l'application devient conséquente. Généralement, on sépare dans ce cas les contrôleurs/modèles/vues en modules. La nouvelle arborescence devient donc :

```
/application // Comporte les fichiers relatifs à votre application
/config // Les fichiers de configuration de votre application
/models // Les modèles utilisés par plusieurs modules
/modules // Les modules de l'application
  /(module 1) // Le dossier pour le module 1 (par ex. forum)
  /config // Éventuellement les fichiers de configuration du module
```

```
    /controllers // Les contrôleurs spécifiques au module
    /models // Les modèles spécifiques au module
    /views // Les vues spécifique au module
/(module 2) // Le dossier pour le module 2 (par ex. blog)
    /controllers // Les contrôleurs spécifiques au module
    /models // Les modèles spécifiques au module
    /views // Les vues spécifique au module
/(module n) // De même pour tous les autres modules
    /controllers // Les contrôleurs spécifiques au module
    /models // Les modèles spécifiques au module
    /views // Les vues spécifique au module


/public
    /images
    /scripts
    /styles
/library
    /zend
    /(autres)
/tmp
    /sessions
    /cache
    /view_compiles
    index.php
```

Si vous utilisez une arborescence en modules, vous devrez spécifier les dossiers des modules dans le fichier bootstrap :

```
$controller->addModuleDirectory('path/to/application/modules/');
```

Zend Framework ne vous fige pas à une seule arborescence, vous pouvez très bien faire un mélange des différentes arborescences disponibles pour arriver à la structure la plus efficace pour votre application.

lien :  [La doc officielle sur les applications modulaires](#)

lien :  [Choisir son arborescence \(Wiki du Zend Framework Anglais\)](#)


lien :  <http://framework.zend.com/wiki/display/ZFPROP/Zend+Framework+Default+Project+Structure+-+Wil+Sinclair>

Comment utiliser le Zend Framework sur un hébergement mutualisé ?

Auteurs : [Guillaume Rossolini](#) ,

Le Zend Framework a besoin de PHP \geq 5.1.4, n'importe quel hébergeur mutualisé proposant cette version de PHP permet d'utiliser le Zend Framework.

À noter que ce framework est à vocation professionnelle et qu'il est préférable d'avoir la main sur son hébergement, sans quoi il n'est pas possible d'en profiter pleinement.

Si vous envisagez d'utiliser des URLs réécrites (URL Rewriting), assurez-vous que votre hébergeur mette à disposition `mod_rewrite` sous Apache. Son absence n'empêche pas l'utilisation de ZF mais complique largement l'utilisation du pattern  MVC.

lien : [FAQ Comment utiliser PHP5 avec un hébergeur qui ne le propose pas par défaut ?](#)

lien : [FAQ Comment utiliser la réécriture d'URL avec un hébergeur qui n'a pas mod_rewrite ?](#)

lien :  [Le modèle MVC et le contrôleur sous PHP](#)

lien :  [Récapitulatif] Zend Framework sur serveurs mutualisés

J'ai une erreur 404 en chargeant une page URL réécrite

Auteurs : [Julien Pauli](#) ,

Erreur 404 veut dire que le mod_rewrite d'apache n'est pas entré en fonction.

Il faut activer le mod_rewrite d'apache, ou alors utiliser l'URL `http://myhost.tld/index.php/admin/` (avec `AcceptPathInfo` à `On` dans la conf d'apache).

Comment utiliser Zend Framework sur un hébergement Free.fr ?

Auteurs : [TheDrev](#) ,

Il faut passer de PHP4 à PHP5 au moyen d'un fichier `.htaccess` et ajouter un répertoire "include" à la racine du site, en y copiant le Zend Framework.

À noter que Free.fr n'autorise pas le mod_rewrite sous Apache, ce qui empêche d'utiliser le MVC.

lien : [FAQ](#) Comment utiliser le Zend Framework sur un hébergement mutualisé ?

Comment gérer les erreurs serveur (404, 500, etc.) ?

Auteurs : [Janitrix](#) ,

S'il existe, Zend Framework va utiliser par défaut le contrôleur `ErrorController` et son action `errorAction()` pour gérer les erreurs (contrôleur introuvable, action introuvable, erreur 500, etc.).

Il vous suffit donc de créer ce contrôleur pour gérer ces erreurs.

controllers/ErrorController.php

```
class ErrorController
{
    public function errorAction()
    {
        $this->view->message = "Erreur !";
    }
}
```

Avec ce contrôleur et la vue qui va avec, vous aurez le message "Erreur !" si vous allez sur un lien inexistant.

Exemple de contrôleur :

```
<?php
/**
 * Cette classe gère les erreurs de navigation (page inexistante, etc).
 * Une partie du code est tiré d'un exemple de Julien Pauli (http://julien-pauli.developpez.com).
 */
class ErrorController
{
    private $_exception;
    private static $errorMessage;
    private static $httpCode;

    public function preDispatch()
    {
```

Exemple de contrôleur :

```
$this->_exception = $this->_getParam('error_handler');

switch ($this->_exception->type) {
    case Zend_Controller_Plugin_ErrorHandler::EXCEPTION_NO_CONTROLLER:
    case Zend_Controller_Plugin_ErrorHandler::EXCEPTION_NO_ACTION:
        self::$httpCode = 404;
        self::$errorMessage = 'Page introuvable';
        break;
    case Zend_Controller_Plugin_ErrorHandler::EXCEPTION_OTHER:
        switch (get_class($this->_exception->exception)) {
            case 'Zend_View_Exception' :
                self::$httpCode = 500;
                self::$errorMessage = 'Erreur de traitement d\'une vue';
                break;
            case 'Zend_Db_Exception' :
                self::$httpCode = 503;

        self::$errorMessage = 'Erreur de traitement dans la base de données';
        break;
            case 'Metier_Exception' :
                self::$httpCode = 200;
                self::$errorMessage = $this->_exception->exception->getMessage();
                break;
            default:
                self::$httpCode = 500;
                self::$errorMessage = 'Erreur inconnue : '. $this->_exception->exception->getMessage();
                break;
        }
        break;
    }
}

public function errorAction()
{
    $this->view->message = self::$errorMessage;
    $this->view->httpCode = self::$httpCode;

    // Affiche la vue.
    echo $this->view->render("erreur/erreur.tpl");
}
}
```

lien :  [Atelier "exceptions en MVC", par Julien Pauli](#)

Comment charger automatiquement les classes ?

Auteurs : Janitrix , Guillaume Rossolini ,

La solution la plus classique est d'utiliser la méthode statique registerAutoload() de la classe Zend_Loader :

```
require_once 'Zend/Loader.php';
Zend_Loader::registerAutoload();
```

Il est recommandé d'utiliser la technique ci-dessus. Pour information, voici le code utilisé en interne :

```
spl_autoload_register(array('Zend_Loader', 'autoload'));
```

Vos classes seront alors automatiquement chargées, et vous n'aurez plus besoin de le faire "manuellement", à partir d'appels à `require_once` par exemple.

Notez que l'autoloading du `Zend_Loader` suit la nomenclature recommandée par le Zend Framework concernant le nom des classes.

Ainsi, avec l'appel suivant :

```
$classe = new Application_Member_Abstract();
```

La classe sera recherchée dans le script `Application/Member/Abstract.php`.

À noter également que l'autoloading peut avoir un impact négatif sur les performances de l'application.

Depuis la version 1.8, la méthode à quelque peu changée, il faut maintenant appeler la méthode `getInstance()`;

```
require_once 'Zend/Loader/Autoloader.php';  
Zend_Loader_Autoloader::getInstance();
```

Où dois-je modifier ma variable d'environnement pour dire si je suis en dev ou en prod ?

Auteurs : Gérard Ernaelsten ,

Si vous avez utilisé le Quick Start de Zend Framework ou les outils permettant de créer une architecture de projet, cette variable d'environnement se trouve dans le fichier `.htaccess`

[Sommaire](#) > Zend_Controller

Comment récupérer le nom du module, du controller et de l'action ?

Auteurs : arnoweb ,

Dans mon controller, je récupère le nom du module, de l'action, du controller et je l'affecte à mes variables de vue pour pouvoir les afficher ou les tester :

```
$this->view->module = $this->getRequest()->getModuleName();  
// recupere le module  
$this->view->controller = $this->getRequest()->getControllerName();  
// recupere le controller  
$this->view->action = $this->getRequest()->getActionName();  
// recupere l'action
```

Comment gérer les exceptions avec le ZF ?

Auteurs : Julien Pauli ,

Le plugin ErrorHandler est fait pour vous : Zend_Controller_Plugin_ErrorHandler.

lien :  [Atelier ZF : Gestions des exceptions intégrées dans MVC : le plugin ErrorHandler, par Julien Pauli](#)

Comment fonctionne le système de routes ?

Auteurs : Janitrix ,

Le Zend Framework intègre un système de routage, complémentaire au module mod_rewrite du serveur web Apache. Il est important de comprendre le fonctionnement du routage par défaut inclut dans le framework.

Le premier routage n'est pas dû au Zend Framework, mais au fichier htaccess. En effet, le fichier htaccess doit diriger la plupart des requêtes vers le fichier "bootstrap". Voici le fichier htaccess proposé par la documentation du Zend Framework :

```
RewriteEngine on  
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} !-d  
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} !-f  
RewriteRule .* /index.php
```

Donc, cette première redirection mène les toutes les requêtes (autres que les fichiers images, CSS et Javascript) vers le fichier index.php, le fichier amorce de l'application.

Dans ce même fichier, le Zend Framework entre en jeu lorsque vous dispatchez l'action, à l'aide du code suivant :

```
$frontController->dispatch();
```

Le "Front Controller" va, selon un ordre déterminé, chercher le contrôleur correspondant à la requête de l'utilisateur. Tout cela grâce aux "routes".

Par défaut, le Zend Framework définit plusieurs routes génériques. Voici quelques exemples de "routes" gérées par défaut (traduits de la documentation officielle) :

```
// Si on ajoute quelques modules
```

```
$frontController->setControllerDirectory(
    array(
        'default' => '/path/to/default/
controllers', // Les contrôleurs par défaut, lorsque le module n'est pas spécifié
        'forum'   => '/path/to/forum/controllers', // Un module forum
        'blog'    => '/path/to/blog/controllers' // Un module blog
    )
);

// Exemple avec seulement le module
http://exemple/forum
    => module == forum

/*
Dans ce cas, le dispatcher va chercher un contrôleur IndexController
dans le module 'test'. Hors, ce module n'existe pas. Il va alors chercher un contrôleur 'test', dans le dossier
*/
http://exemple/test
    => controller == test

/*
On spécifie ici le module et le contrôleur.
*/
http://exemple/blog/article
    => module == blog
    => controller == article

/*
Dans ce cas, on spécifie à la fois le module, le contrôleur, et l'action.
*/
http://exemple/blog/archive/list
    => module == blog
    => controller == archive
    => action == list

/*
En plus du module, du contrôleur et de l'action, on spécifie ici des paramètres.
*/
http://exemple/blog/archive/list/sort/alpha/date/desc
    => module == blog
    => controller == archive
    => action == list
    => sort == alpha
    => date == desc
```

Ces quelques exemples nous montrent que le système de routage du Zend Framework est assez puissant par défaut. Cependant, il vous sera peut être nécessaire de créer un routage plus spécifique. Il est heureusement possible de configurer vos propres routes, à l'aide du routeur.

Comment créer une route statique ?

Auteurs : [Janitrix](#) ,

Bien que la plupart du temps, on utilise des routes dites "dynamiques" (donc avec une partie du lien qui est variable), il se peut que vous ayez besoin de routes statiques. Par exemples, si votre contrôleur s'appelle `ContactController`, avec une action `sendAction`, par défaut, on accéderait à cette action avec le lien `z-f/contact/send`, ce qui n'est pas forcément souhaitable. Vous préféreriez peut être y accéder avec un lien du type `z-f/Nous-Contacter`.

Dans ce genre de situation, on peut facilement créer une route personnalisée :

```
// On récupère le gestionnaire de routes
$routeur = $frontController->getRouter();
```

```
// On créé la route statique
$route = new Zend_Controller_Router_Route_Static(
    'Nous-Contactez',
    array('controller' => 'contact', 'action' => 'send')
);
// On ajoute la route
$route->addRoute('contact', $route);
```

Ce code est relativement simple : on récupère le gestionnaire de routeur à partir de l'instance du Front Controller, puis, on crée un objet `Zend_Controller_Router_Route_Static`, qui représente une route statique. Le premier paramètre est le lien d'accès à cette route, donc concrètement, c'est ce paramètre qui représente la route "virtuelle" par laquelle le visiteur accédera à l'action. Le deuxième paramètre est un tableau d'options, comprenant le contrôleur et l'action cibles de cette route.

Enfin, on ajoute la route à la suite de celles existantes, en spécifiant un identifiant de route et l'objet route associé. Ce nom devrait être unique, mais le Zend Framework ne réalise aucune vérification. Si vous utilisez deux fois le même nom, la dernière entrée écrasera les précédentes. Il faut noter que les listes sont ajoutées dans une pile, par conséquent, la dernière ajoutée sera la première récupérée pour déterminer le contrôleur à utiliser.

[Sommaire > Zend_Db](#)

Comment construire l'objet \$db ?

Auteurs : Guillaume Rossolini , Gérard Ernaelsten ,

Pour construire un objet \$db permettant d'utiliser la base de données, une solution est d'utiliser la méthode abstraite `Zend_Db::factory()`.

Exemple pour MySQL avec PDO

```
$params = array
(
    'host'      => 'localhost',
    'username' => 'yogui',
    'password' => '1234',
    'dbname'   => 'developpez'
);

try
{
    $db = Zend_Db::factory('PDO_MYSQL', $params);
    $db->getConnection();
}
catch (Zend_Db_Adapter_Exception $e)
{
    echo $e->getMessage();
}
```

`Zend_Db::factory()` utilise des adaptateurs (`Zend_Db_Adapter_*`) pour initialiser la connexion. Cela signifie que vous pouvez utiliser un pilote direct comme MySQLi, DB2 ou Oracle, mais aussi passer par PDO pour avoir plus de flexibilité.

Depuis la version 1.8 et `Zend_application`, vous pouvez construire un objet \$db de la façon suivante: dans le fichier `config.ini`

Exemple de fichier `application.ini`

```
; connexion à une base de données
resources.db.adapter = "pdo_mysql"
resources.db.params.host = "localhost"
resources.db.params.username = "mon_user"
resources.db.params.password = "mon_passe"
resources.db.params.dbname = "ma_db"
resources.db.isDefaultTableAdapter = true
```

lien :  [Présentation du Zend Framework - Zend_Db](#)

lien :  [Premiers pas avec Zend_Application](#)

lien : [FAQ](#) [Quels adaptateurs sont disponibles ?](#)

Comment travailler avec plusieurs base de données ?

Auteurs : Gérard Ernaelsten ,

Il est possible de mettre en place un système assez simple pour travailler sur plusieurs base de données, voir même différent SGBDR.

Voici une ébauche de solution.

1) on crée un fichier XML (un tableau est également possible)

```

    <?xml version="1.0" encoding="UTF-8"?>
<config>
<acceptance>
    <drivername>DB2</drivername>
    <host>adresse ip</host>
    <port>50572</port>
    <username>login</username>
    <password>passwor</password>
    <dbname>maDb</dbname>
    <database>MaBase</database>
</acceptance>
<integration extends="acceptance"><
    <host>AutreIp</host>
    <port>50571</port>
    <username>autreLogin</username>
    <password>MonPasswd</password>
</integration>
<MySQL>
    <drivername>pdo_mysql</drivername>
    <host>localhost</host>
    <port>3306</port>
    <username>aityahia</username>
    <password>monpasse</password>
    <dbname>zf-project</dbname>
    <database>MaBase</database>
</MySQL>
<MySQL2 extends="MySQL"><!-- It's the name choosen in the application -->
    <host>localhost</host>
    <port>3307</port>
    <username>Gg</username>
    <password>kmljmlhmk1</password>
</MySQL2>
</config>

```

Ensuite dans la classe model, on crée un objet \$db

```

public function __construct($base='MySQL') {
    $config = new Zend_Config_Xml('vers/fichier/xml',$base);
    $driverName = $config->drivername;
    $params = array
    (
        'host' => $config->host,
        'username' => $config->username,
        'password' => $config->password,
        'dbname' => $config->dbname,
        'port' => $config->port,
    );
    try {
        $this->_db = Zend_Db::factory($driverName,$params);
        $this->_db->getConnection();
        $this->_db->setFetchMode(Zend_Db::FETCH_OBJ);
    }
    catch ( Zend_Db_Adapter_Exception $e) {
        echo $e->getMessage ();
    }
}

```

Enfin dans le controller

```
$db = new Model_Db('DB2');
```

lien : [FAQ](#) Comment ne plus écrire de requêtes SQL ?

Zend_Db est-il sensible à la casse ?

Auteurs : [Reveur](#) ,

Lorsque vous utilisez PDO pour les accès à la base de données, l'extension PDO est configurée par ZF avec la constante PDO::CASE_NATURAL par défaut, ainsi les noms des tables et des champs peuvent être précisés indifféremment en majuscules ou minuscules.

Si vous souhaitez modifier ce comportement, utilisez le code suivant :

```
$db = Zend_Db::factory('PDO_MYSQL', $params);  
$db->setParams(PDO::CASE_LOWER);
```

Comment définir la méthode de récupérations des données ?

Auteurs : [Gérard Ernaelsten](#) ,

Dans Zf il est possible de récupérer les données de différentes façons !

Dans la classe abstract class Zend_Db_Adapter_Abstract, on trouve que le mode est par défaut

```
protected $_fetchMode = Zend_Db::FETCH_ASSOC;
```

Les possibilités sont :

- 1 Zend_Db::FETCH_ASSOC
- 2 Zend_Db::FETCH_NUM
- 3 Zend_Db::FETCH_BOTH
- 4 Zend_Db::FETCH_COLUMN
- 5 Zend_Db::FETCH_OBJ

Pour changer de méthode, il faut dans la création de l'objet \$db appelé la méthode setFetchMode() ;

```
$db = Zend_Db::factory('PDO_MYSQL', $params);  
$db->setFetchMode(Zend_Db::FETCH_OBJ)//récupération sous forme d'objet  
$db->getConnection();
```

On vous renvoie sur la documentation pour voir la définition de chaque méthode .

lien :  [Documentation Zend Framework](#)

Comment ne plus écrire de requêtes SQL ?

Auteurs : Julien Pauli , Guillaume Rossolini ,

Zend_Db permet de s'affranchir du code SQL, mais les fonctionnalités de mapping relationnel objet (ORM) sont encore plus utiles. Le Zend Framework propose cette l'ORM grâce à la classe Zend_Db_Table_Abstract.

Exemple :

```
<?php
class Membres extends Zend_Db_Table_Abstract
{
    protected $_name = 'Membres';

    protected $_primary = 'num';

    protected $_dependentTables = array('Emprunts');

    public function findByNom($nom){
        $where = $this->getAdapter()->quoteInto('nom = ?',(string)$nom);
        return $this->fetchRow($where);
    }
}
?>
```

```
<?php
$tableMembre = new Membres();
$julien = $tableMembre->findByNom('julien');
$julien->date_naissance = '1982-12-08';
$julien->save();
?>
```

lien :  [Présentation du Zend Framework - #Zend_Db](#), par Julien Pauli

lien :  [EZPDO: Object-Relational Mapping en PHP](#), par Pierre-Nicolas Mougel

Faire une requête classique avec Zend_DB ?

Auteurs : Gérard Ernaelsten ,

Il est tout à fait possible de faire des requêtes écrites à la main et de récupérer le résultat.

```
public function unRequeteAvecParametre($name){
    $sql = 'Select nom,prenom FROM maTbale WHERE nom = ?';
    return $this->_db->fetchAll($sql,$name);
}
```

Quels adaptateurs sont disponibles ?

Auteurs : [Guillaume Rossolini](#) ,

Utilisez toujours  la doc en ligne pour obtenir la réponse à ce genre de questions.

À ce jour, vous pouvez utiliser les adaptateurs suivants :

- db2
- mysqli
- oracle
- pdo_mssql
- pdo_mysql
- pdo_oci
- pdo_pgsql
- pdo_sqlite

lien : [FAQ](#) Comment construire l'objet \$db ?

Est-ce que ZF peut s'occuper de la liaison entre les tables ?

Auteurs : [Julien Pauli](#) , [Guillaume Rossolini](#) ,

Oui, avec un peu de bonne volonté ;)

Les tuples des tables sont identifiés par un numéro automatique. Zend Framework permet de faire des requêtes en chaîne, de la forme (pour trouver le message #1 d'un forum) :

```
$message = $messages->find(1)->current();
```

C'est pratique ; néanmoins, à l'issue de cette requête, j'obtiens l'identifiant de l'auteur `$message->id_auteur` plutôt que son nom. Je suis donc obligé de faire une seconde requête pour trouver le nom de l'auteur à partir de son identifiant :

```
$auteur = $auteurs->find($message->id_auteur)->current();
```

Une solution pour éviter tout ce travail est d'étendre `Zend_Db_Table_Abstract` et d'utiliser cette nouvelle classe dans nos développements.

Plus de détails dans l'atelier par Julien.

lien :  [Atelier ZF : Des résultats pertinents avec les tables liées : l'ORM en mode FullLoading, par Julien Pauli](#)

Peut-on faire de l'héritage de tables avec le ZF ?

Auteurs : [Guillaume Rossolini](#) , [Julien Pauli](#) ,

Ce n'est pas prévu par le framework de départ mais c'est possible en utilisant la classe de Réflexion introduite depuis PHP5.

lien :  [Documentation sur la Réflexion](#)

lien :  [Atelier : Implémentation de l'héritage de tables SQL sous Zend Framework, par Julien Pauli](#)

Comment récupérer les erreurs retournées par la base de données ?

Auteurs : [Janitrix](#) , [Guillaume Rossolini](#) ,

Dans le cas d'une erreur de la base de données lors de la connexion ou d'une requête, une exception `Zend_Db_Exception` est levée. Ainsi, il vous suffit de capturer cette exception pour récupérer le message d'erreur retourné par la base de données.

```
try {
    // $db est de type Zend_Db_Adapter, $query Zend_Db_Select
    $db->fetchRow($select);
} catch (Zend_Db_Exception $e) {
    die($e->getMessage());
}
```

Si vous utilisez une transaction au moment du déclenchement l'exception, elle est mise en erreur : il vous faut exécuter `$db->commit()` ou `$db->rollback()`.

```
$this->_db->beginTransaction();
$row = $this->createRow();
try {
    $row->save();
    $this->_db->commit(); // n'est pas exécuté si save() lance une exception
} catch (Zend_Db_Exception $e) {
    $this->_db->rollback();
    // traiter l'exception
}
```

lien :  [Tutoriel : Les exceptions en PHP 5 par Guillaume Affringue](#)

[Sommaire](#) > [Zend_Db](#) > [Zend_Db_Select](#)

Comment faire un SELECT ?

Auteurs : Guillaume Rossolini ,

Zend Framework utilise PDO pour exécuter les requêtes SQL. Il y a plusieurs méthodes simples pour lancer une requête SELECT :

Méthode 1

```
$select = $db->query('SELECT champ FROM table');
```

Méthode 2

```
$select = $db->select();  
$select->from('table', 'champ');
```

La récupération du résultat se fait de la même manière pour les deux méthodes :

```
$rows = $select->fetchAll();
```

\$rows contient tous les résultats de la requête sous forme de tableau (Array). Il ne reste plus qu'à appeler foreach dessus.

lien : [FAQ](#) Comment ne plus écrire de requêtes SQL ?

Comment faire un SELECT MAX() ?

Auteurs : lecra ,

Il suffit d'appeler la fonction MAX() dans la liste des champs :

Solution 1 :

```
$select = $db->query('SELECT MAX(champ) FROM table');  
$max = $db->fetchOne($select);
```

Solution 2 :

```
$select = $db->select();  
$select->from('table', 'MAX(champ)');
```

Solution 2 :

```
$max = $db->fetchOne($select);
```

lien :  [Tutoriel Zend Framework : Sélections avec Zend_Db](#)

Comment appliquer ORDER BY aux fonctions find<TableClass> ?

Auteurs : j.roc , Julien Pauli ,

Il est possible d'implémenter sa propre logique, mais il faut à ce moment là réécrire les méthodes en les surchargeant.

Comment savoir combien d'enregistrements sont retournés par ma requête ?

Auteurs : byc_r ,

Après avoir appelé la méthode fetchAll(), le résultat est placé dans un tableau PHP. Il suffit donc d'appeler la fonction PHP count() sur ce résultat pour savoir combien de tuples sont retournés par la requête.

```
$select = $db->select();  
$select->from('table', 'champ');  
$rows = $select->fetchAll();  
echo count($rows);
```

SQLServer : Syntaxe incorrecte (General error 10007)

Auteurs : websurfeur ,

Le Zend Framework, en construisant les requêtes avec Zend_Db_Select, ajoute des guillemets pour protéger les noms des tables et des champs. SQL Server 7 n'aime pas du tout ce genre de requête, il les préfère sans guillemet.

Ce code PHP :

```
$select = $db->select();  
$select->select("ma_table", array("colonne1" ) );
```

Est traduit en SQL :

```
SELECT "ma_table"."colonne1" FROM "ma_table";
```

La 1^{ère} solution consiste à faire un remplacement des quotes doubles par rien avant d'exécuter la requête.

La 2^{ème} consiste à activer l'option quoted identifier sur la table que vous souhaitez requêter dans MS SQL Server 7.

[Sommaire](#) > [Zend_Debug](#)

Comment accéder au système de debug ?

Auteurs : arnoweb ,**Dans mon controller :**

```
Zend_Debug::dump($this->_request);
```

Comment utiliser PHPUnit dans un projet MVC ?

Auteurs : Matthew Weier O'Phinney ,**Il est possible d'utiliser PHPUnit avec le Zend Framework et son composant MVC. Voici un exemple :**

```
class FooControllerTest extends PHPUnit_Framework_TestCase
{
    public function setUp()
    {
        $this->front = Zend_Controller_Front::getInstance();
        $this->front->addModuleDirectory('/path/to/
modules'); // le chemin des modules et contrôleurs
        $this->front->resetInstance(); // réinitialise l'instance du "front" contrôleur
        $this->front->returnResponse(true); // désactive l'envoi automatique de la réponse
    }

    public function testIndexPageContents()
    {
        // L'URL est utilisée principalement pour pouvoir définir l'URI demandée
        // et les informations qui lui sont liées :
        $request = new Zend_Controller_Request_Http('http://localhost/');

        // On a désactivé l'envoi automatique, on peut donc utiliser l'objet réponse de cette façon
        $response = $this->front->dispatch($request);

        // On réalise le test
        $this->assertFalse($response->isException());

        // on teste que le contenu contient certaines strings
        $this->assertContains('index page', $response->getBody());

        // etc...
    }
}
```

Sommaire > Zend_Form

Mon code n'est pas valide XHTML

Auteurs : stalak ,

Les champs *input* générés par `Zend_Form_Element_Text` ou `Zend_Form_Element_Submit` n'ont pas de balise fermante.

Pour y remédier, il faut mettre ce code dans le *bootstrap* :

```
// setup view
$view = new Zend_View();
$viewRenderer = Zend_Controller_Action_HelperBroker::getStaticHelper('ViewRenderer');
$viewRenderer->setView($view);
$viewRenderer->view->doctype('XHTML1_TRANSITIONAL');
```

Comment traduire les messages d'erreur de validation ?

Auteurs : vg33 ,

Il est conseillé d'utiliser `Zend_Translate`.

Pour un site unilingue, le plus simple est d'étendre `Zend_Form` avec une classe contenant les traductions de messages. Exemple fonctionnel avec des traductions perso des principaux messages :

```
class My_Form extends Zend_Form
{
    /**
     * construction du formulaire
     *
     * @param mixed $options
     * @return void
     */
    public function __construct($options = null)
    {
        parent::__construct($options);

        // traduction des messages d'erreur de validation
        $french = array(
            'notAlnum' => "'%value%' ne contient pas que des lettres et/ou des chiffres.",
            'notAlpha' => "'%value%' ne contient pas que des lettres.",
            'notBetween' => "'%value%' n'est pas compris entre %min% et %max% inclus.",
            'notBetweenStrict' => "'%value%' n'est pas compris entre %min% et %max% exclus.",
            'dateNotYYYY-MM-DD' => "'%value%' n'est pas une date au format AAAA-MM-JJ (exemple : 2000-12-31).",
            'dateInvalid' => "'%value%' n'est pas une date valide.",
            'dateFalseFormat' => "'%value%' n'est pas une date valide au format JJ/MM/AAAA (exemple : 31/12/2000).",
            'notDigits' => "'%value%' ne contient pas que des chiffres.",
            'emailAddressInvalid' => "'%value%' n'est pas une adresse mail valide selon le format adresse@domaine.",
            'emailAddressInvalidHostname' => "'%hostname%' n'est pas un domaine valide pour l'adresse mail '%value%'.",
            'emailAddressInvalidMxRecord' => "'%hostname%' n'accepte pas l'adresse mail '%value%'.",
            'emailAddressInvalidDotAtom' => "'%localPart%' ne respecte pas le format dot-atom.",
            'emailAddressQuotedString' => "'%localPart%' ne respecte pas le format quoted-string.",
            'emailAddressInvalidLocalPart' => "'%localPart%' n'est pas une adresse individuelle valide.",
            'notFloat' => "'%value%' n'est pas un nombre décimal.",
```

```
'notGreaterThan' => "'%value%' n'est pas strictement supérieur à '%min%'.",
'notInt'=> "'%value%' n'est pas un nombre entier.",
'notLessThan' => "'%value%' n'est pas strictement inférieur à '%max%'.",
'isEmpty' => "Ce champ est vide : vous devez le compléter.",
'stringEmpty' => "Ce champ est vide : vous devez le compléter.",
'regexNotMatch' => "'%value%' ne respecte pas le format '%pattern%'.",
'stringLengthTooShort' => "'%value%' fait moins de %min% caractères.",
'stringLengthTooLong' => "'%value%' fait plus de %max% caractères."

);

$translate = new Zend_Translate('array', $french, 'fr');
$this->setTranslator($translate);
}
}
```

[Sommaire](#) > [Zend_Layout](#)

Comment passer une variable au layout ?

Auteurs : [vg33](#),**Insérez le code suivant où vous le souhaitez (controller, plugin, action_helper...) :**

```
//Récupère l'instance du layout en cours
$layout= Zend_Layout::getMVCInstance();
//Assigne les données au segment
$layout->assign('foo', 'test');
```

Et dans le layout :

```
echo $this->layout()->foo;
```

[Sommaire > Zend_Session](#)

Comment stocker une valeur dans une session ?

Auteurs : [Yoteco](#) ,

Dans le Zend Framework, les sessions fonctionnent avec un système de « namespaces ». Ce système permet de bien séparer les sessions entre les différents modules. Pour stocker une valeur dans une session, il faut utiliser la classe `Zend_Session_Namespace` :

```
<?php
require_once('Zend/loader.php') ;
Zend_Loader::loadClass('Zend_Session_Namespace') ;

// Insertion d'une valeur
$defaultNamespace = new Zend_Session_Namespace('Default') ;
$defaultNamespace->myKey = $myValue ;

// Récupération d'une valeur
$defaultNamespace = new Zend_Session_Namespace('Default') ;
$myValue = $defaultNamespace->myKey ;

?>
```

lien :  [Tutoriel : Les sessions en PHP, par julp et Mathieu Lemoine](#)

Sommaire > Zend_Translate

Comment initialiser un système de translation ?

Auteurs : Gérard Ernaelsten ,

Dans le Bootstrap.php :

```
$translate = new Zend_Translate('array','/My/lan/');
Zend_Registry::set('Zend_Translate', $translate);
//si nous voulons fixer une langue par défaut
Zend_Registry::set('Zend_Translate', $translate,'fr');
```

Dans le fichier .ini [A partir de 1.8]

```
resources.translate.registry_key = "Zend_Translate"
resources.translate.adapter = "array"
resources.translate.options.scan = "directory"
resources.translate.data = APPLICATION_PATH "/My/lan/"
//si nous voulons fixer une langue par défaut
resources.translate.default = fr
```

Exemple de fichier (fr.php)

```
<?php
/**
 *
 * @copyright 2009 MaitrePylos Technologies
 * @author Ernaelsten Gerard <info@formatux.be>
 * @license GPL
 * @version Release: 0.1
 * www.formatux.be
 */
return array(
'notAlnum' => "'%value%' ne contient pas que des lettres et/ou des
chiffres.",
'notAlpha' => "'%value%' ne contient pas que des lettres.",
'notBetween' => "'%value%' n'est pas compris entre %min% et %max%
inclus.",
'notBetweenStrict' => "'%value%' n'est pas compris entre %min% et %max
% exclus.",
'dateNotYYYY-MM-DD'=> "'%value%' n'est pas une date au format AAAA-MMJJ
(exemple : 2000-12-31).",
'dateInvalid' => "'%value%' n'est pas une date valide.",
'dateFalseFormat' => "'%value%' n'est pas une date valide au format
JJ/MM/AAAA (exemple : 31/12/2000).",
'notDigits' => "'%value%' ne contient pas que des chiffres.",
'emailAddressInvalid' => "'%value%' n'est pas une adresse mail valide
selon le format adresse@domaine.",
'emailAddressInvalidHostname' => "'%hostname%' n'est pas un domaine
valide pour l'adresse mail '%value%'.",
'emailAddressInvalidMxRecord' => "'%hostname%' n'accepte pas l'adresse
mail '%value%'.",
'emailAddressDotAtom' => "'%localPart%' ne respecte pas le format dotatom.",
'emailAddressQuotedString' => "'%localPart%' ne respecte pas le format
quoted-string.",
'emailAddressInvalidLocalPart' => "'%localPart%' n'est pas une adresse
individuelle valide.",
'notFloat' => "'%value%' n'est pas un nombre décimal.",
'notGreaterThan' => "'%value%' n'est pas strictement supérieur à '%min
%'.",
'notInt'=> "'%value%' n'est pas un nombre entier.",
'notLessThan' => "'%value%' n'est pas strictement inférieur à '%max
%'.
```

```
'isEmpty' => "Ce champ est vide : vous devez le compléter.",  
'stringEmpty' => "Ce champ est vide : vous devez le compléter.",  
'regexNotMatch' => "%value% ne respecte pas le format '%pattern%'.",  
'stringLengthTooShort' => "%value% fait moins de %min% caractères.",  
'stringLengthTooLong' => "%value% fait plus de %max% caractères."  
);
```

Comment récupérer son Objet translate?

Auteurs : Gérard Ernaelsten ,

Suivant la convention Il faut mettre dans le registry une entrée 'Zend_Translate'

```
Zend_Registry::set('Zend_Translate', $translate);  
//ou depuis la 1.8 dans le fichier .ini  
resources.translate.registry_key = "Zend_Translate"
```

Ensuite si vous voulez étendre un classe Zend Framework, vous retrouverez dans la plupart des cas une méthode qui s'appelle getTranslator() ou getDefaultTranslator(), qui combiner avec le registry vous rend la main sur votre objet translate.

Dans les autres cas vous rappeler voter objet se trouvant dans votre registry

```
$translate = Zend_Registry::get('Zend_Translate');  
$this->view->uneVariable = $translate->translate('notAlnum');  
//ou  
$this->view->uneVariable = $translate->_('notAlnum');  
//vous pouvez-bien entendu créer une classe pour vous faciliter la vie  
class My_Translate  
{  
    public static function translate($message) {  
        $translate = Zend_Registry::get('Zend_Translate');  
        return $translate->_($message);  
    }  
}  
//et l'utilisez comme suit  
My_Translate::translate('notAlnum')
```

L'encodage de mes fichiers TMX est incorrect

Auteurs : Yoteco ,

J'utilise un fichier .tmx pour faire mes traductions. Je ne peux pas utiliser des caractères "Français" typiquement un "ç" ou un "é".

Solution : vérifier dans les options de votre éditeur que l'encodage correspond bien à celui donné dans l'en-tête du fichier XML :

```
<?xml version="1.0" encoding="utf-8"?>
```

[Sommaire](#) > [Zend_View](#)

Pourquoi mes caractères sont-ils mal encodés ("?" au lieu d'accents) avec le Zend Framework ?

Auteurs : [Guillaume Rossolini](#) ,

Il faut appeler la méthode `setEscape()` de la Vue, dans la méthode `init()` du contrôleur :

```
class IndexController extends Zend_Controller_Action
{
    ...

    function init()
    {
        ...
        $this->view->setEscape('utf8_encode');
        ...
    }

    ...
}
```

Peut-on utiliser un moteur de gabarits avec Zend_View ?

Auteurs : [Guillaume Rossolini](#) ,

Oui, il est possible d'utiliser un moteur de gabarits (template engine) avec `Zend_View`. La Vue (du point de vue du Zend Framework) est donc un script en PHP pur, qui utilise un gabarit en HTML pur (selon la syntaxe du moteur choisi).

Quels moteurs de gabarits peut-on utiliser avec le Zend Framework ?

Auteurs : [Guillaume Rossolini](#) ,

N'importe quel moteur de gabarits peut convenir. C'est à vous de choisir celui qui vous convient le mieux.

lien :  [Comparatif : Les principaux moteurs de template en PHP, par Guillaume Rossolini](#)

Comment utiliser un gabarit principal pour tout son site ?

Auteurs : [Rob Allen](#) , [vg33](#) ,

Si votre site a un header (doctype, css, menu...) et un footer (menu de bas de page, infos...) commun à toutes les pages, une façon simple de l'implémenter est d'intégrer le code suivant à chaque script de vue :

```
<?php echo $this->render('header.phtml'); ?>

// ici le contenu de votre page

<?php echo $this->render('footer.phtml'); ?>
```

Mais ce n'est pas du tout DRY (Don't Repeat Yourself, ne vous répétez pas). Nous vous proposons donc d'utiliser un template principal qui affiche en bonne place le contenu du script d'action appelé.

Notre but est non pas de faire un `render()` de l'header et du footer dans chaque script de vue, mais plutôt d'utiliser un template principal qui affiche le contenu du script d'action appelé.

Le `Zend_Controller_Action_Helper_ViewRenderer` est un code qui automatise le `render` d'un template fondé sur l'action appelée. Il est très utile, mais il fait un `render` du template d'action, pas du template principal. Il s'agit donc d'étendre le `Zend_Controller_Action_Helper_ViewRenderer` pour qu'il intègre notre template principal. On en profitera pour modifier automatiquement le suffixe par défaut des scripts de vue (`.phtml`) en `.tpl.php`.

Le template principal du site s'appelle `site.tpl.php` et se situe dans `views/scripts`. Il pourrait ressembler à ceci :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr" xml:lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title><?php echo $this->escape($this->pageTitle);?></title>
<link rel="stylesheet" href="<?php echo $this->baseUrl; ?>/css/site.css"
    type="text/css" media="screen" />
</head>
<body>
    <div id="header">
        <?php echo $this->menu(); /* menu view helper */ ?>
    </div>
    <div id="content">
        <?php echo $this->render($this->actionScript); ?>
    </div>
    <div id="footer">
        Copyright 2007 Rob Allen
    </div>
</body>
</html>
```

`$this->actionScript` est le script associé à l'action courante qui est automatiquement déterminée par `Zend_Controller_Action_Helper_ViewRenderer`. Par exemple, pour l'action `index` du contrôleur `index`, le script d'action est `views/scripts/index/index.tpl.php`.

Nous devons étendre le `Zend_Controller_Action_Helper_ViewRenderer` :

`application/(default)/views/helpers/ViewRenderer.php`

```
<?php
class Controller_Action_Helper_ViewRenderer
    extends Zend_Controller_Action_Helper_ViewRenderer
{
    /**
     * Nom du template principal. Par défaut : 'site.tpl.php'.
     *
     * @var string
     */
    protected $_layoutScript = 'site.tpl.php';

    /**
     * Constructeur
     *
     * Configure le suffixe des scripts de vue à "tpl.php" sauf si
     * un autre suffixe est passé dans les paramètres $options
     *
     * @param Zend_View_Interface $view
     * @param array $options
     * @return void
     */
}
```

application/(default)/views/helpers/ViewRenderer.php

```
public function __construct(Zend_View_Interface $view = null,
                           array $options = array())
{
    if (!isset($options['viewSuffix'])) {
        $options['viewSuffix'] = 'tpl.php';
    }
    parent::__construct($view, $options);
}

/**
 * Configure le template principal
 *
 * @param string $script
 */
public function setLayoutScript($script)
{
    $this->_layoutScript = $script;
}

/**
 * Renvoie le nom du template principal
 *
 * @return string
 */
public function getLayoutScript()
{
    return $this->_layoutScript;
}

/**
 * Renvoie le script d'action et l'assigne à la vue pour l'utiliser
 * dans le template principal. Fait un render() du template principal
 * et complète le body de l'objet Response.
 *
 * @param string $script
 * @param string $name
 */
public function renderScript($script, $name = null)
{
    $this->view->baseUri = $this->_request->getBaseUrl();
    if (null === $name) {
        $name = $this->getResponseSegment();
    }

    // affecte le nom du script d'action à la vue
    $this->view->actionScript = $script;

    // Fait un render() du template principal
    // et complète le body de l'objet Response.
    $layoutScript = $this->getLayoutScript();
    $layoutContent = $this->view->render($layoutScript);
    $this->getResponse()->appendBody($layoutContent, $name);

    $this->setNoRender();
}
}
```

Tout ce qu'il nous reste à faire est de modifier notre bootstrap (fichier d'amorce) pour que notre nouveau ViewRenderer soit utilisé à la place de celui par défaut. On ajoute donc les lignes suivantes dans index.php avant la première instantiation du FrontController :

bootstrap

```
<?php
```

bootstrap

```
// Utilise notre ViewRenderer action helper
require_once('../application/(default/)views/helpers/ViewRendererer.php');
$viewRendererer = new Controller_Action_Helper_ViewRendererer();
Zend_Controller_Action_HelperBroker::addHelper($viewRendererer);
```

C'est tout ! Notre script d'action contient juste le code HTML spécifique à l'action, et notre template principal est rendu automatiquement avec le code HTML de l'action au bon emplacement !