



The PHP Company

# Introduction à Zend Framework 2

Mickaël Perraud

# Mickael Perraud

---



- Contributeur ZF depuis 2007
- Responsable documentation française
- Travaille sur l'aide à la traduction et propose les versions déconnectées de la documentation PDF / CHM



@mikaelkael / <http://mikaelkael.fr>

# Zend Framework 2 : pourquoi ?

# Zend Framework jusqu'à aujourd'hui

---

- Annoncé en octobre 2005
- Mars 2006 : v0.1
- Juillet 2007 : v1.0 : MVC, Db, Acl, Auth
- Mars 2008 : v1.5 : Form, Layout, Context
- Avril 2009 : v1.8 : Application, Tool, Nav
- Juin 2010 : formation CRTeam
- Octobre 2010 : v1.11 : Cloud, UserAgent

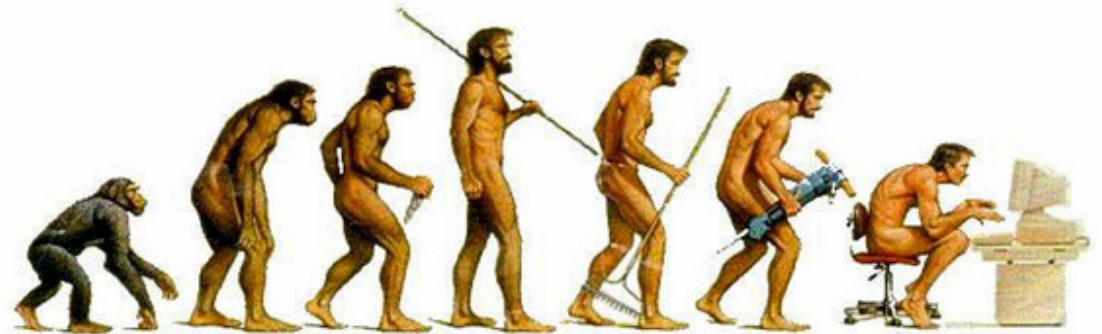
# Zend Framework 2.0

---

Don't think



think



# Zend Framework 2.0

---

« The primary thrust of ZF 2.0 is to make a more consistent, well-documented product, improving developer productivity and runtime performance. »

*Matthew Weier O'Phinney*

# Zend Framework 2.0 : buts principaux

---

- Améliorer l'extensibilité
- Améliorer les performances
- Améliorer la courbe d'apprentissage
- Réécrire quand cela est nécessaire

# Zend Framework 2 : améliorer l'extensibilité

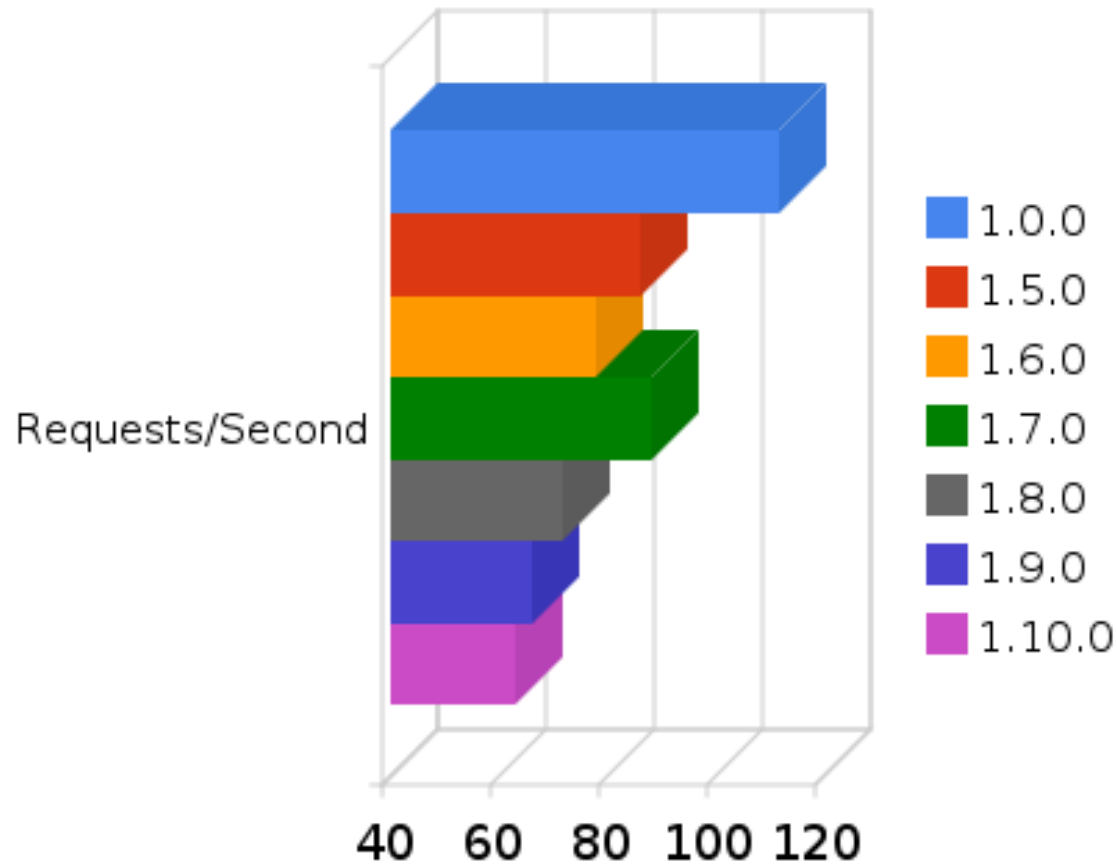
# Zend Framework 2.0 : améliorer l'extensibilité

---

- Suppression des Singletons
- Programmation par contrat : des interfaces au dessus des classes abstraites
- Suppression des dépendances codées en dur

# Zend Framework 2 : améliorer les performances

# Zend Framework 2.0 : améliorer les performances



# Zend Framework 2.0 : améliorer les performances

---

- « Kill the magic! »
- Autoloading
- Chargement des plugins

# Zend Framework 2.0 : « kill the magic! »

## Exemple d'un view helper classique :

```
echo $this->headLink()->appendStylesheet('foo.css');  
/**  
 * Lance Zend_View::__call()  
 * Appelle Zend_View::getHelper()  
 * Appelle Zend_View::_getPlugin()  
 * Appelle Zend_Loader_PluginLoader::load()  
 * Appelle Zend_Loader::isReadable()  
 * Appelle call_user_func (hits autoloader...)  
 * qui appelle Zend_Loader::loadClass  
 * qui appelle Zend_Loader::loadFile  
 * qui appelle include_once  
 * Instanciation de l'helper  
 * Appelle la méthode de l'helper via call_user_func_array()  
 * Retourne l'instance de l'helper  
 * Appelle la méthode de l'instance (lance __call...)  
 */
```

# Zend Framework 2.0 : « kill the magic! »

---

Enlevons la magie :

```
$this->broker('head_link')
    ->appendStylesheet('foo.css');
/**
 * Lance PhpRenderer::broker()
 * Appelle HelperBroker::load()
 * Appelle HelperLoader::load()
 * Lance autoloader
 * qui réalise un include_once
 * Instanciation de l'helper
 * Appelle la méthode de l'helper
 */
```

**=> La magie dégrade les performances**

# Zend Framework 2.0 : « kill the magic! »

---

Questions légitimes du débutant :

```
class MonController extends Zend_Controller_Action
{
    public function monAction()
    {
        $this->view->maVariable = 'maValeur';
    }
}
```

- C'est quoi view ? C'est défini où ?
- Où est le rendu dans tout ça ?
- Et les layouts ?

**=> La magie rend l'apprentissage plus difficile**

# Zend Framework 2.0 : autoloading

---

- Suppression des `require_once`
- Option 1 : ClassMapAutoloader - fichier de mapping
  - Pré-compile un tableau de mapping pour tout ZF et par composant
  - Outils pour générer le mapping
  - Le plus efficace en terme de performances

# Zend Framework 2.0 : autoloading

---

- Option 1 :

```
php classmap_generator.php Some/Directory/
```

```
// .classmap.php
return array(
    'Foo\SomeController' => __DIR__ . '/foo/controllers/SomeController.php',
    'Foo\Model\Bar'      => __DIR__ . '/foo/models/Bar.php',
);
```

```
// ClassMapAutoloader
require_once 'Zend/Loader/ClassMapAutoloader.php';
$loader = new Zend\Loader\ClassMapAutoloader('./.classmap.php');
$loader->register();
```

```
$bar = new Foo\Model\Bar();
```

# Zend Framework 2.0 : autoloading

---

- Option 2 : StandardAutoloader - un tableau explicite de paires namespaces => dossier
  - Pas d'utilisation de l'`include_path`
  - Flexibilité et performance au cours du développement
  - Peut aussi agir en autoloader par défaut (s'appuie sur `include_path`)

# Zend Framework 2.0 : autoloading

---

- Option 2 :

```
require_once 'Zend/Loader/StandardAutoloader.php';
$loader = new Zend\Loader\StandardAutoloader();

$loader->registerNamespace('Mkk', APPLICATION_PATH . '/../library/Mkk');
$loader->registerPrefix('Doctrine', APPLICATION_PATH .
    '/../library/Doctrine');

// ceci n'est pas recommandé mais reste toutefois possible
$loader->setFallbackAutoloader(true);

$loader->register();

$foo = new Mkk\Model\Foo();
```

# Zend Framework 2.0 : plugin loading

---

- Problèmes :
  - L'autoloading basé sur une pile de chemins est lent
  - La casse est importante
  - Les solutions actuelles ne s'intéressent qu'à l'aspect chargement des classes
  - L'instanciation est différente suivant les composants
  - La persistance est différente suivant les composants

# Zend Framework 2.0 : plugin loading

---

- Solution
  - Mapping avec alias par défaut

```
namespace My\Plugins;
use Zend\Loader\PluginClassLoader;

class ComponentLoader extends PluginClassLoader
{
    protected $plugins = array(
        'foo'      => 'My\Plugins\Foo',
        'bar'      => 'My\Plugins\Bar',
        'foobar' => 'My\Plugins\FooBar',
    );
}
```

# Zend Framework 2.0 : plugin loading

- Loaders couplés avec des Brokers
  - ◆ Agissant comme un registre
  - ◆ Permettant l'instanciation (avec arguments)

```
namespace My\Component;
use Zend\Loader\PluginClassLoader,
    Zend\Loader\PluginBroker;
class ComponentBroker extends PluginBroker
{
    protected $defaultClassLoader = 'My\Component\ComponentLoader';
    protected function validatePlugin($plugin)
    {
        if (!$plugin instanceof Adapter) {
            throw new Exception\RuntimeException();
        }
        return true;
    }
}
```

# Zend Framework 2.0 : plugin loading

---

- Utilisation

```
namespace My\Component;
class Factory
{
    /* Méthodes setBroker() and broker() */
    public function get($adapter, array $options)
    {
        return $this->broker()
            ->load($adapter, $options);
    }
}
```

# Zend Framework 2 : améliorer la courbe d'apprentissage

# Zend Framework 2.0 : création d'une meilleure documentation

---

- Reproches :
  - Options pas toujours documentées
  - Méthodes non présentées
  - Inconsistance entre les différents composants
  - Exemple souvent spécifiques au composant
  - Pas d'exemple pour une application complète

# Zend Framework 2.0 : création d'une meilleure documentation

---

- Proposition :
  - Introduction
  - Démarrage rapide
  - Options
  - Méthodes
  - Exemples

# Zend Framework 2.0 : revoir la consistance de l'API

---

- Reproches
  - Les méthodes magiques sont difficiles à appréhender
  - On ne sait pas toujours comment et quand étendre les composants
  - Certains concepts ne sont pas évidents à comprendre (décorateurs `Zend_Form`)

# Zend Framework 2.0 : revoir la consistance de l'API

---

- Proposition
  - Refactorisation quand les modèles d'utilisation diffèrent des modèles de design
  - Réduire le nombre d'appels magiques
  - Revoir certaines API non claires

# Zend Framework 2 : réécrire quand cela est nécessaire

# Zend Framework 2.0 : gestion des exceptions

---

- Basé sur les exceptions SPL
- Chaque composant possède son interface
- Permet d'utiliser `\Exception` dans tous les cas

# Zend Framework 2.0 : gestion des exceptions

---

```
// Interface
namespace Zend\Barcode;
interface Exception{}

// Exception spécifique
namespace Zend\Barcode\Exception;
use Zend\Barcode\Exception;
class InvalidArgumentException extends \InvalidArgumentException implements
Exception{}

// Utilisation
use Zend\Barcode\Exception;
try {
    // Faire qqch
} catch (Exception\InvalidArgumentException $e) {
    // Exception spécifique
} catch (Exception $e) {
    // Interface du composant
} catch (\Exception $e) {
    // Exception générale
}
```

# Zend Framework 2.0 : réécrire Zend\Session

---

- Problèmes :
  - Boîte noire non testable
  - Stockage des espaces de noms incompatible avec `$_SESSION`
  - Incompatibilités avec `ext/session`
- Nouveau composant :

```
use Zend\Session\SessionManager,  
    Zend\Session\Container as SessionContainer;  
  
$manager = new SessionManager(array('//mes options));  
$container = new SessionContainer('MonEspace', $manager);  
$container['maCle'] = 'maValeur';  
$container->setExpirationSeconds(60);
```

# Zend Framework 2.0 : réécrire Zend\Db

---

- Problèmes :
  - Difficile de récupérer la couche de connexion pour la partager entre les instances et les classes
  - Difficile de récupérer les données du schéma
  - Difficile d'étendre
  - Difficile d'ajouter des tâches pre/post
- Proposition :
  - ▶ [http://framework.zend.com/wiki/display/ZFDEV2/..](http://framework.zend.com/wiki/display/ZFDEV2/)

# Zend Framework 2.0 : réécrire Filtrés et Validateurs

---

- Problèmes :
  - Utilisation statique et en chaîne sont mixés au sein du même objet
  - N'ont pas tous la même méthodologie dans le chargement des plugins
  - Certains aspects requis par `Zend_Filter_Input` ou `Zend_Form` ne sont pas toujours supportés

# Zend Framework 2.0 : réécrire Filtrés et Validateurs

---

- Nouveaux composants :

```
namespace Zend\Validator;

if (StaticValidator::execute($valeur, 'int')) {
    // validation statique
}

$chaine = new ValidatorChain();
$chaine->addValidator(new StringLength(5, 5), true)
    ->addValidator(new PostCode('fr_FR'));

if ($chaine->isValid($valeur)) {
    // validation en chaîne
}
```

# Zend Framework 2 : où en est-on aujourd'hui ?

# Zend Framework 2.0 : les étapes

---

- Migration vers Git
- Suppression `require_once`
- Ajout des interfaces (sur classes abstraites)
- Migration vers les espaces de noms
- Additions SPL, `SignalSlot`
- Réécriture de `Zend\Session`

Zend Framework 2.0.0dev1

<http://devzone.zend.com/article/12385-First-Development-Milestone-of-ZF-2.0-Released>

# Zend Framework 2.0 : les étapes

---

- Refactorisation de l'autoloading et du chargement des plugins
- Refactorisation des exceptions
- Réécriture de `Zend\View`

Zend Framework 2.0.0dev2

<http://framework.zend.com/announcements/2010-11-03-zf2dev2>

# Zend Framework 2.0 : la suite

---

- Refactorisation MVC
- Refactorisation I18N et L10N
- Amélioration des tests
- Documentation
- Packaging
- Outils de migration

# Zend Framework 2.0 : comment contribuer ?

---

- Wiki ZF2 : <http://bit.ly/zf2wiki>
- Liste des contributeurs ZF :  
[zf-contributors-subscribe@lists.zend.com](mailto:zf-contributors-subscribe@lists.zend.com)
- IRC : #zftalk.dev on Freenode
- Git
  - ▶ <git://git.zendframework.com/zf.git>
  - ▶ <http://github.com/zendframework/zf2>

**Merci**  
**Questions ?**

Talks : <http://joind.in/event/view/518>  
Slideshare : <http://www.slideshare.net/mikaelkael>