2025/12/11 10:35 1/6 ORM IgnitedRecord

ORM IgnitedRecord

<

ORM Doctrine >>

- 1. Introduction à Codelgniter
- 2. Helpers
- 3. Bibliothèques
- 4. Sessions Codelgniter
- 5. Validation des formulaires
- 6. ORM IgnitedRecord
- 7. Doctrine
- 8. Javascript et codelgniter

IgnitedRecord est un ORM qui peut être associé à CodeIgniter. Mais le projet semble abandonné... →Utilisation déconsillée Lien de téléchargement : IgnitedRecord download

L'aide est disponible à l'intérieur du zip téléchargé.

Installation

Dézipper l'archive. Copier les fichiers du dossier **application/libraries/** dans le dossier du même nom de votre application web.

ignitedrecord doit être chargé:

• soit automatiquement avec autoload.php:

```
$autoload['libraries'] = array('database','ignitedrecord/ignitedrecord');
```

Le chargement de la bibliothèque database est indispensable

• Soit dans le code d'un contrôleur par exemple :

```
$this->load->library('ignitedrecord/ignitedrecord');
```

ou

```
$this->load->orm();
```

Création des classes métier

Une classe métier correspond à la notion de model dans codelgniter.

Considérons la base de données suivante :

Last update: 2019/08/31 14:21



La base de données sera composée de 2 models : utilisateur et categorie. La relation de type CIF entre utilisateurs et categories peut s'exprimer de la façon suivante :

- Chaque utilisateur appartient à 1 catégorie (belongs to)
- Dans chaque categorie, on peut compter de 0 à n utilisateurs (has many)

Le model utilisateur

Dans le dossier **application/models**, créer le fichier utilisateur.php :

|h application/models/utilisateur.php

```
<?php
class Utilisateur extends IgnitedRecord {
    public $table='utilisateurs';
    public $belongs_to ='categorie';
}
?>
```

- Un **model** est un fichier contenant une classe dont le nom commence par une majuscule.
- Le nom du fichier doit être le même que celui de la classe, mais en minuscule.
- Le fichier doit être enregistré dans le dossier application/models/
- La table de la base de données associée au model est nommée modelNames

Le model categorie

Dans le dossier application/models, créer le fichier categorie.php :

|h application/models/categorie.php

```
<?php
class Categorie extends IgnitedRecord{
    public $table='categories';
    public $has_many='utilisateurs';
}
</pre>
```

Chargement des models

Le chargement peut être automatique, par le biais de application/config/autoload.php

2025/12/11 10:35 3/6 ORM IgnitedRecord

```
$autoload['model'] = array('categorie','utilisateur');
```

ou bien se faire dans un contrôleur :

```
$this->load->model('utilisateur');
```

Gestion des utilisateurs

Contrôleur utilisateurs

Ajouter un contrôleur utilisateurs dans controllers :

- la méthode **all** charge tous les utilisateurs, et leur catégorie correspondante.
- Elle appelle ensuite la vue v_utilisateurs et lui passe les utilisateurs chargés (users)

|h application/controllers/utilisateurs.php

Vues

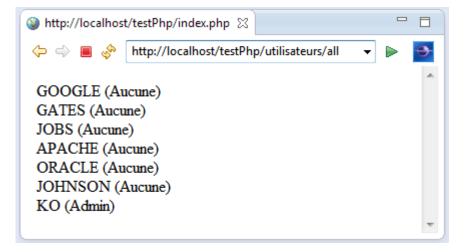
Liste des utilisateurs

Créer la vue v_utilisateurs pour afficher la liste des utilisateurs : La variable \$utilisateurs est récupérée par la méthode all du contrôleur utilisateurs

|h application/views/v utilisateurs.php

```
<?php
foreach ($utilisateurs as $user){
    echo($user->nom." (".$user->cat->nom.")<br>};
}
```

Tester en allant à l'adresse http://localhost/testPhp/utilisateurs/all/



Ajout d'utilisateur

Modification du contrôleur

Modifier le contrôleur utilisateurs :

- La méthode **add** permet d'afficher un formulaire **v_utilisateur_add** permettant d'ajouter un utilisateur en saisissant son nom.
- La méthode **submit_add** effectue la validation du formulaire en cas de succès de la validation puis appelle la vue **v success_add**

|h application/controllers/utilisateurs.php

```
<?php
class Utilisateurs extends CI_Controller{
    public function add(){
        $this->load->helper(array('form', 'url'));
        $this->load->library('form validation');
        $this->form validation->set rules('username', 'Username',
'trim|required|min length[5]|max length[12]|xss clean');
        if ($this->form validation->run() == FALSE)
            $this->load->view('v utilisateur add');
        else
            $this->submit add($ POST["username"]);
    public function submit add($name){
        $new_user = $this->utilisateur->new_record();
        $new user->nom=$name;
        $new user->save();
        $this->load->view('v_success_add',array('user'=>$new_user));
    public function all(){
```

2025/12/11 10:35 5/6 ORM IgnitedRecord

Ajout des vues

La vue v_utilisateur_add sera appelée par l'intermédiaire du contrôleur utilisateurs/add

|h application/views/v_utilisateur_add.php

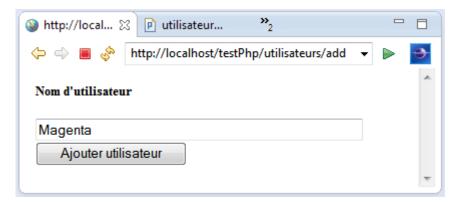
```
<hre><html>
<head>
<title>Ajout utilisateur</title>
</head>
<body>
</php echo validation_errors(); ?>
</php echo form_open('utilisateurs/add/'); ?>
<h5>Nom d'utilisateur</h5>
<input type="text" name="username" value="<?php echo set_value('username'); ?>" size="50" />
<div><input type="submit" value="Ajouter utilisateur" /></div>
</form>
</body>
</html>
```

La vue **v_success_add** sera appelée après soumission du formulaire par le contrôleur **utilisateurs/submit_add**

|h application/views/v_success_add.php

```
<?php
echo($user->nom." ajouté");
?>
```

Tester en allant à l'adresse : http://localhost/testPhp/utilisateurs/add/



Vérifier l'insertion dans la base de données du nouvel utilisateur.

Sur le même principe que pour les utilisateurs, en respectant MVC :

- Créer un contrôleur categories
- Afficher la liste des catégories, et les utilisateurs correspondants
- Créer la fonctionnalité d'ajout de catégorie
- Créer la fonctionnalité de modification d'une catégorie existante
- Créer la fonctionnalité de suppression d'une catégorie

ORM Doctrine >>

From:

http://slamwiki2.kobject.net/ - SlamWiki 2.1

Permanent link:

http://slamwiki2.kobject.net/slam4/php/codeigniter/orm

Last update: 2019/08/31 14:21

