CakePHP 3

Le framework PHP pour le développement de vos applications web

informatique technique

Fichiers complémentaires à télécharger

eni Editions

Collection EPSILON

Benoît GOYHENECHE
Benjamin LAMPÉRIER

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence de l'ouvrage **EP3CAK** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1
Avant de démarrer

Chapitre 2
Présentation de CakePHP

Chapitre 3
Installation et configuration

Chapitre 4
Les modèles

Chapitre 5
Les contrôleurs

Chapitre 6
Les vues

Chapitre 7
Tâches courantes

Chapitre 8
Utilisation avancée

Chapitre 9
Plugins indispensables

Chapitre 10
Cas pratique

Chapitre 5
Les contrôleurs

1. Introduction

Les contrôleurs sont les chefs d'orchestre de vos applications. À partir des requêtes des utilisateurs, les actions des contrôleurs sont exécutées, avec pour mission de retourner aux vues toutes les informations nécessaires à la construction d'une réponse.

Pour ce faire, les contrôleurs vont interagir avec les modèles, recevoir des données utilisateurs, manipuler des requêtes et des réponses HTPP, envoyer des informations aux vues, etc.

Dans les applications CakePHP, tous les contrôleurs héritent des méthodes et priorités d'un contrôleur principal nommé `AppController`.

Remarque

Rappel des conventions : les contrôleurs sont nommés en fonction du modèle qu'ils manipulent. Par exemple, le contrôleur en charge des utilisateurs est nommé `UtilisateursController`.

2. Le contrôleur App

La classe `AppController` étend la classe `Cake\Controller\Controller` qui est incluse dans le cœur de CakePHP.

Le contrôleur `App` est disponible dans **src/Controller/AppController.php** et se présente sous cette forme :

```
<?php
namespace App\Controller;
use Cake\Controller\Controller;
use Cake\Event\Event;
class AppController extends Controller
{
    /**
     * Initialization hook method.
     *
     * Use this method to add common initialization code like
     *loading components.
     *
     * e.g. `$this->loadComponent('Security');`
     *
     * @return void
     */
    public function initialize()
    {
        parent::initialize();
        $this->loadComponent('RequestHandler');
        $this->loadComponent('Flash');
    }
    /**
     * Before render callback.
     *
     * @param \Cake\Event\Event $event The beforeRender event.
     * @return void
     */
    public function beforeRender(Event $event)
    {
        if (!array_key_exists('_serialize', $this->viewVars) &&
            in_array($this->response->type(), ['application/json',
'application/xml'])
        ) {
            $this->set('_serialize', true);
        }
    }
}
```

Tous les contrôleurs déclarés dans une application CakePHP héritent de la classe `AppController`. Cela signifie que tous les attributs et méthodes définis dans ce contrôleur sont disponibles dans tous les contrôleurs de l'application.

3. Le contrôleur Pages

Le fichier **PagesController.php** est un contrôleur particulier qui permet d'afficher du contenu statique. Il est disponible dès l'installation de CakePHP, sans configuration ni développement particulier.

La page d'accueil visible juste après une installation de CakePHP est rendue grâce au contrôleur `Pages`.

Ce contrôleur contient uniquement une action `display()` qui affiche le fichier de vue qui lui est passé en paramètre.

Code de la fonction `display()` :

```
public function display()
{
    $path = func_get_args();

    $count = count($path);
    if (!$count) {
        return $this->redirect('/');
    }
    $page = $subpage = null;

    if (!empty($path[0])) {
        $page = $path[0];
    }
    if (!empty($path[1])) {
        $subpage = $path[1];
    }
    $this->set(compact('page', 'subpage'));

    try {
        $this->render(implode('/', $path));
    } catch (MissingTemplateException $e) {
        if (Configure::read('debug')) {
```

```
                throw $e;
            }
            throw new NotFoundException();
        }
    }
```

Pour créer une nouvelle page statique, il suffit de créer un fichier portant l'extension **.ctp** et de le sauvegarder dans le répertoire **src/Template/Pages**.

Par exemple, pour créer une page « Bonjour » :

- Créez le fichier **src/Template/Pages/bonjour.ctp**.
- Rendez-vous à l'adresse **http://mon-url.fr/pages/bonjour**.

4. Créer un contrôleur

Pour créer un contrôleur il suffit de créer une classe PHP qui est dans l'espace de noms `APP\Controller` et qui étend `AppController`.

Cette classe porte le nom du modèle à manipuler suivi de « Controller ».

Le fichier est à sauvegarder dans le dossier **src/Controller** et porte le même nom que la classe, suivi de l'extension **.php** .

Exemple :

Le code suivant représente le contrôleur `ArticlesController`, qui a pour objectif de traiter les articles de l'application stockés dans la table articles.

```
<?php
namespace App\Controller;

use App\Controller\AppController;

class ArticlesController extends AppController
{
}
```

5. Les actions du contrôleur

Les actions d'un contrôleur sont les méthodes publiques définies dans ce contrôleur.

Elles prennent en entrée les paramètres de la requête et renvoient en sortie la réponse à l'utilisateur dans la vue associée.

En suivant les conventions de CakePHP, la vue se trouve dans un fichier du nom de l'action, au sein d'un répertoire portant le nom du contrôleur.

Exemple :

Le code ci-dessous est celui d'un contrôleur simple permettant de lister, voir, ajouter, modifier et supprimer :

```
<?php
namespace App\Controller;

use App\Controller\AppController;

class ExemplesController extends AppController
{

    // Liste des exemples
    public function index(
    {
        $exemples = $this->paginate($this->Exemples);

        $this->set(compact('exemples'));
        $this->set('_serialize', ['exemples']);
    }

    // Voir un exemple
    public function view($id = null)
    {
        $exemple = $this->Exemples->get($id, [
            'contain' => []
        ]);

        $this->set('exemple', $exemple);
        $this->set('_serialize', ['exemple']);
    }

    // Ajouter un exemple
    public function add()
```

```
    {
        $exemple = $this->Exemples->newEntity();
        if ($this->request->is('post')) {
            $exemple = $this->Exemples->patchEntity($exemple,
$this->request->data);
            if ($this->Exemples->save($exemple)) {
                $this->Flash->success(__('The exemple has been saved.'));
                return $this->redirect(['action' => 'index']);
            } else {
                $this->Flash->error(__('The exemple could not be saved.
Please, try again.'));
            }
        }
        $this->set(compact('exemple'));
        $this->set('_serialize', ['exemple']);
    }

    // Modifier un exemple
    public function edit($id = null)
    {
        $exemple = $this->Exemples->get($id, [
            'contain' => []
        ]);
        if ($this->request->is(['patch', 'post', 'put'])) {
            $exemple = $this->Exemples->patchEntity($exemple,
$this->request->data);
            if ($this->Exemples->save($exemple)) {
               $this->Flash->success(__('The exemple has been saved.'));
                return $this->redirect(['action' => 'index']);
            } else {
                $this->Flash->error(__('The exemple could not be saved.
Please, try again.'));
            }
        }
        $this->set(compact('exemple'));
        $this->set('_serialize', ['exemple']);
    }

    // Supprimer un exemple
    public function delete($id = null)
    {
        $this->request->allowMethod(['post', 'delete']);
        $exemple = $this->Exemples->get($id);
        if ($this->Exemples->delete($exemple)) {
            $this->Flash->success(__('The exemple has been deleted.'));
        } else {
            $this->Flash->error(__('The exemple could not be deleted.
Please, try again.'));
```