

CakePHP 3

Le framework PHP
pour le développement
de vos applications web



informatique technique

Fichiers complémentaires
à télécharger




Collection

epsilon

Benoît GOYHENECHÉ
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

- 1. Introduction 17
- 2. Public visé 18
- 3. Prérequis 18
- 4. Organisation du livre 18

Chapitre 1

Avant de démarrer

- 1. Introduction 21
- 2. La programmation orientée objet en PHP 21
 - 2.1 Pourquoi utiliser la programmation orientée objet ? 22
 - 2.2 Le vocabulaire 22
 - 2.2.1 Qu'est-ce qu'un objet ? 22
 - 2.2.2 Qu'est-ce qu'une classe ? 23
 - 2.2.3 Qu'est-ce qu'une instance ? 23
 - 2.3 Utiliser les objets 23
 - 2.3.1 Déclarer une classe. 23
 - 2.3.2 Instancier un objet. 24
 - 2.4 L'héritage en programmation orientée objet avec PHP 25
 - 2.4.1 L'héritage simple 25
 - 2.4.2 Les traits 25
 - 2.4.3 Les classes abstraites 26
 - 2.4.4 Les interfaces 28
- 3. Qu'est-ce qu'un ORM ? 29
- 4. Le pattern Modèle-Vue-Contrôleur. 29

- 5.5 Récapitulatif 45
- 6. Où obtenir de l'aide ? 46
 - 6.1 Le site officiel CakePHP 46
 - 6.2 Le CookBook 46
 - 6.3 L'API 47
 - 6.4 Stack Overflow 47
 - 6.5 Communauté française de CakePHP 47

Chapitre 3
Installation et configuration

- 1. Introduction 49
- 2. Prérequis 49
 - 2.1 Environnement de développement local 50
 - 2.2 Serveur HTTP et réécriture des URL 52
 - 2.2.1 Activer la réécriture des URL 53
 - 2.2.2 Désactiver la réécriture d'URL 54
 - 2.3 Vérifier la version de PHP 54
 - 2.3.1 Vérifier si le module PHP Mbstring est activé 55
 - 2.3.2 Vérifier si l'extension intl est activée 56
 - 2.3.3 Accéder au plugin DebugKit 57
- 3. Installer CakePHP grâce à Composer 58
 - 3.1 Installer Composer 58
 - 3.2 Installer CakePHP 58
- 4. Architecture des dossiers 61
- 5. Configuration 63
 - 5.1 Introduction 63
 - 5.2 Configuration générale 63
 - 5.2.1 Lire des données de configuration 64
 - 5.2.2 Écrire des données de configuration 65
 - 5.3 Activer le mode débogage 66
 - 5.4 Configurer la base de données 67

- 4.4 Accesseurs et mutateurs 92
 - 4.4.1 Accesseurs 92
 - 4.4.2 Mutateurs 93
- 4.5 Déterminer si une donnée a été modifiée 94
- 5. Sauvegarder des données 95
 - 5.1 Introduction 95
 - 5.2 Insérer des données 95
 - 5.3 Modifier des données 96
 - 5.4 Supprimer des données 96
- 6. Validation des données 98
 - 6.1 Validation à la construction de l'entité 98
 - 6.1.1 Ajouter des règles de validation 99
 - 6.1.2 Utiliser un autre ensemble de règles de validation 103
 - 6.2 Validation à la sauvegarde de l'entité 103
 - 6.2.1 Vérifier l'unicité d'un champ 104
 - 6.2.2 Vérifier la validité d'une clé étrangère 104
- 7. Les requêtes 104
 - 7.1 Constructeur de requête 105
 - 7.1.1 L'objet Query 105
 - 7.1.2 Parcourir les résultats de la requête 105
 - 7.1.3 Sélectionner des colonnes 108
 - 7.1.4 Définir des conditions simples 109
 - 7.1.5 Trier les données 109
 - 7.1.6 Limiter le nombre de résultats 109
 - 7.1.7 Fonctions SQL avancées 110
 - 7.1.8 Regrouper les résultats 111
 - 7.1.9 Désactiver l'hydratation 112
 - 7.1.10 Combinaisons « OU » et « ET » 112
 - 7.1.11 Récupérer le nombre de résultats 112
 - 7.2 Déboguer les requêtes 113
 - 7.3 Récupérer une entité à partir de la clé primaire 113

6 **CakePHP 3**

Le framework PHP pour le développement de vos applications web

7.4	Utilisation des finders	114
7.4.1	Retourner les résultats au format clé/valeur	114
7.4.2	Créer des finders personnalisés	115
7.4.3	Finders dynamiques	115
7.5	Récupérer les données associées	116
7.5.1	Charger les données associées avec <code>contain()</code>	116
7.5.2	Filtrer par les données associées avec <code>matching()</code>	117
8.	Les comportements (behaviors)	118
8.1	Utiliser un comportement	118
8.2	CounterCache	119
8.2.1	Utilisation basique	119
8.2.2	Utilisation avancée	120
8.3	Timestamp	121
8.3.1	Introduction	121
8.3.2	Modifier les champs	121
8.3.3	Mettre à jour les timestamps	122
8.4	Translate	122
8.4.1	Introduction	122
8.4.2	Création de la table <code>i18n</code> dans la base de données	122
8.4.3	Attacher le comportement à une table	123
8.4.4	Sélectionner une traduction	124
8.4.5	Récupérer les traductions	124
8.4.6	Sauvegarder des traductions	125
8.5	Tree	125
8.5.1	Introduction	125
8.5.2	Prérequis	125
8.5.3	Attacher le comportement à une table	126
8.5.4	Rechercher des données hiérarchisées	127
8.5.5	Réorganiser des données hiérarchisées	128
8.5.6	Sauvegarder des données hiérarchisées	129
8.6	Créer un comportement	129

Chapitre 5 Les contrôleurs

1. Introduction	131
2. Le contrôleur App	132
3. Le contrôleur Pages	133
4. Créer un contrôleur	134
5. Les actions du contrôleur	135
6. Les objets Request et Response	137
6.1 Request	137
6.1.1 Accéder aux paramètres de la requête	138
6.1.2 Accéder aux données de la requête	138
6.1.3 Accéder aux variables d'environnement	139
6.1.4 Informations sur le chemin	139
6.1.5 Vérifier la requête	139
6.2 Response	140
6.2.1 Définir les en-têtes	140
6.2.2 Définir l'encodage	141
6.2.3 Gérer les types de contenu	141
6.2.4 Envoyer des fichiers	142
6.2.5 Gestion du cache	143
7. Interagir avec les vues	143
7.1 Définir les variables de vue	143
7.2 Afficher une vue	145
8. Les redirections	146
9. Les méthodes de rappel (callback)	147
10. Les composants	148
10.1 Auth	149
10.2 Cookie	150
10.2.1 Configurer les cookies	150
10.2.2 Utiliser les cookies	152
10.3 Csrf	153

10.4 Flash	155
10.5 Security	155
10.5.1 Forcer l'utilisation du SSL	156
10.5.2 Restreindre les requêtes entre contrôleurs	156
10.5.3 Empêcher la falsification des formulaires	157
10.6 Pagination	157
10.7 RequestHandler	158
10.7.1 Obtenir des informations sur les requêtes	158
10.7.2 Répondre aux requêtes	160
10.8 Créer un composant	160

Chapitre 6

Les vues

1. Introduction	163
2. Utiliser les variables	165
3. La mise en page (layout)	166
4. Les vues	168
5. Les éléments	169
6. Les assistants (helpers)	170
6.1 Assistants disponibles	171
6.2 Charger un assistant	171
6.3 Html	172
6.3.1 Définir le jeu de caractères	172
6.3.2 Lier des fichiers CSS	172
6.3.3 Lier des fichiers JavaScript	173
6.3.4 Afficher une image	173
6.3.5 Créer des liens	174
6.3.6 Créer des listes imbriquées	175
6.4 Form	176
6.4.1 Créer un formulaire	176
6.4.2 Créer des éléments de formulaire standards	177

- 6.4.3 Créer des boutons 179
- 6.4.4 Fermer le formulaire 180
- 6.5 Flash..... 180
- 6.6 Number..... 181
 - 6.6.1 Le formatage des nombres 181
 - 6.6.2 Le formatage des pourcentages..... 182
 - 6.6.3 Le formatage des nombres à virgule..... 183
 - 6.6.4 Le formatage des devises..... 183
- 6.7 Text..... 185
 - 6.7.1 Convertir du texte..... 185
 - 6.7.2 Extraire du texte 185
 - 6.7.3 Mise en valeur d'un extrait..... 188
 - 6.7.4 Concaténer un tableau 189
 - 6.7.5 Gestion des liens et des e-mails..... 190
- 6.8 Time 191
 - 6.8.1 Tester les données temporelles..... 192
 - 6.8.2 Formater les données temporelles 195
- 6.9 Url 196
- 6.10 Session..... 198
- 7. Les cellules..... 199
- 8. Les assistants personnalisés 201
 - 8.1 Créer un assistant..... 201
 - 8.2 Utiliser un assistant créé 202

Chapitre 7
Tâches courantes

- 1. Introduction 203
- 2. Authentification..... 203
 - 2.1 Introduction 203
 - 2.2 Ajouter et configurer le composant Auth..... 204
 - 2.3 Créer la table Users dans la base de données 205

2.4	Créer le contrôleur	206
2.4.1	Introduction	206
2.4.2	Identifier les utilisateurs et les connecter	206
2.4.3	Rediriger les utilisateurs après la connexion	207
2.4.4	Déconnecter les utilisateurs	208
2.5	Créer les vues	209
2.5.1	Connexion	209
2.5.2	Ajout d'un utilisateur	210
2.6	Créer le modèle	211
2.6.1	Ajouter des règles de validation	211
2.6.2	Crypter le mot de passe	212
3.	Les autorisations	213
3.1	Introduction	213
3.2	Gérer les autorisations	213
3.3	Autoriser des actions	215
4.	Envoyer des e-mails	215
4.1	Fonctionnement de base	215
4.2	Configuration des transports	216
4.3	Envoyer des e-mails en utilisant les vues	217
4.4	Envoyer des pièces jointes	219
4.5	Créer des e-mails réutilisables	220
5.	Pagination	222
5.1	Introduction	222
5.2	Mise en place du composant Paginator	223
5.3	Mise en place de l'assistant PaginatorHelper	225
5.3.1	Créer des liens triés	226
5.3.2	Créer des liens de page numérotés	226
5.3.3	Créer des liens de saut de page	227
5.3.4	Créer un compteur de pages	229
5.3.5	Vérifier l'état de la pagination	229

6. Gérer les messages Flash	230
6.1 Introduction	230
6.2 Utilisation du composant FlashComponent	231
6.3 Utilisation de l'assistant FlashHelper	232
7. Créer un formulaire sans modèle	233
7.1 Définir le formulaire	233
7.2 Traiter les données	235
7.3 Créer la vue du formulaire	236
8. Requêtes Ajax	236
8.1 Introduction	236
8.2 Appel de la requête depuis une vue	237
8.3 Utiliser la mise en page Ajax	238

Chapitre 8 Utilisation avancée

1. Introduction	239
2. Routage	239
2.1 Introduction	239
2.2 Les scopes	240
2.3 Créer une route	241
2.3.1 Les éléments de route	242
2.3.2 Passer les paramètres à l'action	243
2.3.3 Nommer les routes	244
2.3.4 Préfixer les routes	244
2.3.5 Routes des plugins	245
2.3.6 Routes et SEO	245
2.3.7 Extensions de fichiers	246
2.4 Définir les routes par défaut	246
3. Sécurité	246
3.1 Introduction	246
3.2 Chiffrer et déchiffrer des données	247

3.3	Hacher des données	248
4.	Événements système	249
4.1	Introduction	249
4.2	Événements (events)	249
4.2.1	Introduction	249
4.2.2	Événement du cœur	249
4.2.3	Déclarer des événements	251
4.3	Écouteurs (listeners)	252
5.	Internationalisation	254
5.1	Introduction	254
5.2	Rendre les chaînes traduisibles	254
5.2.1	Chaîne de caractères simple	254
5.2.2	Regrouper les messages par domaine	255
5.2.3	Définir un contexte	255
5.2.4	Utiliser des variables	255
5.2.5	Gérer les pluriels	256
5.3	Fichiers de traduction	256
5.4	Définir la localisation par défaut	257
5.5	Localiser les dates et les nombres	258
6.	Interface de programmation (API)	259
6.1	Introduction	259
6.2	Créer des routes RESTful	259
6.3	Mise en place du contrôleur	260
6.4	Lire une API	262
6.4.1	Les types de requêtes	263
6.4.2	Exécuter une requête	264
6.4.3	Récupérer le résultat	266
6.4.4	Paramétrer une connexion	269

Chapitre 9
Plugins indispensables

- 1. Introduction 271
- 2. Ajouter un plugin à son application 272
 - 2.1 Installer un plugin avec Composer 272
 - 2.2 Charger un plugin dans l'application 272
- 3. Migrations 272
 - 3.1 Créer un fichier de migration 273
 - 3.2 Appliquer une migration 277
 - 3.3 Annuler une migration 278
 - 3.4 Connaître l'état des migrations 279
- 4. Générer du code avec Bake 280
 - 4.1 Introduction 280
 - 4.2 Générer un comportement 281
 - 4.3 Générer une cellule 282
 - 4.4 Générer un composant 283
 - 4.5 Générer un contrôleur 284
 - 4.6 Générer des données d'essai 287
 - 4.7 Générer un formulaire 289
 - 4.8 Générer un assistant 290
 - 4.9 Générer une classe générique de mail 291
 - 4.10 Générer une capture de la base de données 291
 - 4.11 Générer un modèle 293
 - 4.12 Générer un plugin 295
 - 4.13 Générer une commande 297
 - 4.14 Générer l'aide d'une commande 298
 - 4.15 Générer les vues 299
 - 4.16 Générer les cas de test 302
 - 4.17 Générer une application complète 304
- 5. Inspecter avec Debug Kit 307
 - 5.1 Installation de Debug Kit 307
 - 5.2 Utilisation de la barre de débogage 308

6.	Tester avec PHPUnit	314
6.1	Installation de PHPUnit	314
6.2	Les premiers tests	315
6.3	Les fixtures	317
6.4	Tester les classes Table	320
6.5	Tests d'intégration sur les contrôleurs	321
6.5.1	Simuler une requête	322
6.5.2	Les méthodes d'assertion	322
7.	Vérifier la syntaxe avec CodeSniffer	325
7.1	Introduction	325
7.2	Installation du plugin CakePHP CodeSniffer	326
7.3	Utilisation	326
8.	Gérer les suppressions logiques avec SoftDelete	327
8.1	Introduction	327
8.2	Installer le plugin SoftDelete	327
8.3	Configurer le plugin	327
8.4	Utilisation	328
8.4.1	Restaurer des données supprimées logiquement	328
8.4.2	Supprimer définitivement les données	329
9.	Utiliser Bootstrap dans ses formulaires	329
9.1	Installer le plugin Bootstrap Helper	330
9.2	Utiliser le plugin Bootstrap Helper	331
9.2.1	Utiliser l'assistant Html	331
9.2.2	Utiliser l'assistant Form	334
9.2.3	Utiliser l'assistant Modal	339
10.	Exporter des données au format CSV	342
10.1	Installation de CsvView	342
10.2	Utilisation	342
10.2.1	Export simple	342
10.2.2	Exporter plusieurs variables	344
10.2.3	Définir les propriétés à exporter	344
10.2.4	Définir les lignes d'en-tête et de pied de tableau	345

- 10.2.5 Modifier le nom du fichier 346
- 10.2.6 Personnaliser la vue de l'export. 346
- 11. Créer des fichiers PDF. 348
 - 11.1 Installation de CakePDF 348
 - 11.2 Utilisation. 349

Chapitre 10
Cas pratique

- 1. Introduction 351
- 2. Conception 351
- 3. Installation et paramétrage de CakePHP 352
- 4. Mise en place de la base de données 353
 - 4.1 Les familles de produits 354
 - 4.2 Les produits. 355
- 5. Génération de code 357
 - 5.1 Générer les modèles 358
 - 5.1.1 Générer la table et l'entité du modèle Familles 358
 - 5.1.2 Générer la table et l'entité du modèle Produits 361
 - 5.2 Générer les contrôleurs. 364
 - 5.2.1 Générer le contrôleur Familles 364
 - 5.2.2 Générer le contrôleur Produits 367
 - 5.3 Générer les vues 370
 - 5.3.1 Générer les vues pour la table Familles 370
 - 5.3.2 Générer les vues pour la table Produits 377
- 6. Personnaliser l'affichage de l'application 384
 - 6.1 Installation de Bootstrap Helper 384
 - 6.2 Utilisation. 388
 - 6.2.1 Les vues gérant les familles 388
 - 6.2.2 Les vues gérant les produits 392

16 _____ CakePHP 3

Le framework PHP pour le développement de vos applications web

7. Gestion des utilisateurs	396
7.1 Créer la table des utilisateurs.	397
7.2 Générer le code	398
7.3 Mettre en place l'authentification.	402
7.4 Personnaliser l'affichage en fonction des droits	407
8. Exporter en CSV	408
8.1 Installer le plugin CsvView	408
8.2 Mettre en place l'export	409
9. Utiliser une fenêtre modale	410
10. Mise en place de règles métier	413
10.1 La gestion des quantités produites	413
10.2 Affichage des familles.	413
Index	417



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 HTTP, 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')) {
```

Le framework PHP pour le développement de vos applications web

```
        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.));
        }
    }

```