

2026 CakePHP-Day2

CakePHP 概覽

M11305502 王朝威

CakePHP 概覽

- CakePHP 旨在讓常見的網頁開發任務變得簡單、輕鬆。
- 透過提供一個一體化(all-in-one)工具箱來幫助您入門，CakePHP 的各個部分可以很好地協同或單獨工作。
- 本概覽的目的是介紹 CakePHP 中的一般概念，並讓你快速概述這些概念是如何在 CakePHP 中實現的。
- 本概覽以 [CakePHP 5.x Chiffon Book](#) 為參考。
- 翻譯：王朝威。



約定優於配置

convention over configuration

- CakePHP 提供了一個基本的組織結構，涵蓋類別名稱、檔案名稱、資料庫表名和其他約定。
- 雖然這些約定需要一些時間來學習，但透過遵循 CakePHP 提供的約定，您可以避免不必要的配置並建立統一的應用程式結構，從而使處理各種項目變得簡單。

模型層 The Model Layer

- 模型層 (Model Layer) 代表應用程式中實作業務邏輯的部分。
- 它負責檢索資料並將其轉換為應用程式中主要的有意義的概念。
- 這包括處理、驗證、關聯或與處理資料相關的其他任務。
- 在社交網路的情況下，模型層將負責保存使用者資料、保存朋友的關聯、儲存和檢索使用者照片、尋找新朋友的建議等任務。
- 模型物件可以看作是「Friend」、「User」、「Comment」或「Photo」。
- 如果我們想從 `users` 表格中載入一些資料，我們可以這樣做：

```
use Cake\ORM\Locator\LocatorAwareTrait;

$users = $this->fetchTable('Users');
$resultset = $users->find()->all();
foreach ($resultset as $row) {
    echo $row->username;
}
```

模型層 The Model Layer

- 您會注意到，在開始處理資料之前，我們不必編寫任何程式碼。
- 通過使用約定，CakePHP 將對尚未定義的表格和實體類別使用標準類別。
- 如果我們想創建一個新使用者並保存它（通過驗證），我們可以這樣做：

```
use Cake\ORM\Locator\LocatorAwareTrait;

$users = $this->fetchTable('Users');
$user = $users->newEntity(['email' =>
    'mark@example.com']);
$users->save($user);
```

視圖層 The View Layer

- 視圖層 (View layer) 負責渲染模型資料 (modeled data) 的展示。
- 它與模型對象 (Model objects) 分離，主要任務是利用可用的資訊生成應用程式所需的任何展示介面。
- 例如，視圖可以使用模型資料來渲染包含該資料的 HTML 視圖模板，或者生成供他人使用的 XML 格式結果：

```
// 在視圖模板檔案中，我們將會為每一個使用者渲染一個「元素」。  
<?php foreach ($resultset as $user): ?>  
    <li class="user">  
        <?= $this->element('user_info', ['user' =>  
$user]) ?>  
    </li>  
<?php endforeach; ?>
```

視圖層 The View Layer

- 視圖層提供了多種擴展點，例如視圖模板（**View Templates**）、元素（**Elements**）和視圖單元（**View Cells**），讓你能夠重複使用你的展示邏輯。
- 視圖層不僅限於 **HTML** 或文本形式的資料表示，它還可以用來傳遞常見的資料格式，例如 **JSON**、**XML**，並且通過可插拔的架構，還可以支持任何其他你可能需要的格式，例如 **CSV**。

控制器層 The Controller Layer

- 控制器層 (Controller Layer) 負責處理使用者的請求，並在模型層 (Model) 和視圖層 (View) 的協助下渲染回應。
- 可以將控制器看作一個經理，它確保完成任務所需的所有資源都被正確地分配給合適的工作者。
- 控制器等待客戶端的請求，根據身份驗證或授權規則檢查請求的有效性，將資料的提取或處理委派給模型層，選擇客戶端接受的展示資料類型，最後將渲染過程交給視圖層處理。

控制器層 The Controller Layer

- 例如，以下是一個用於使用者註冊的控制器範例：
- 你可能會注意到，我們從未明確地渲染視圖（View）。CakePHP 的約定會自動處理選擇正確的視圖，並使用我們透過 `set()` 準備的視圖資料來進行渲染。

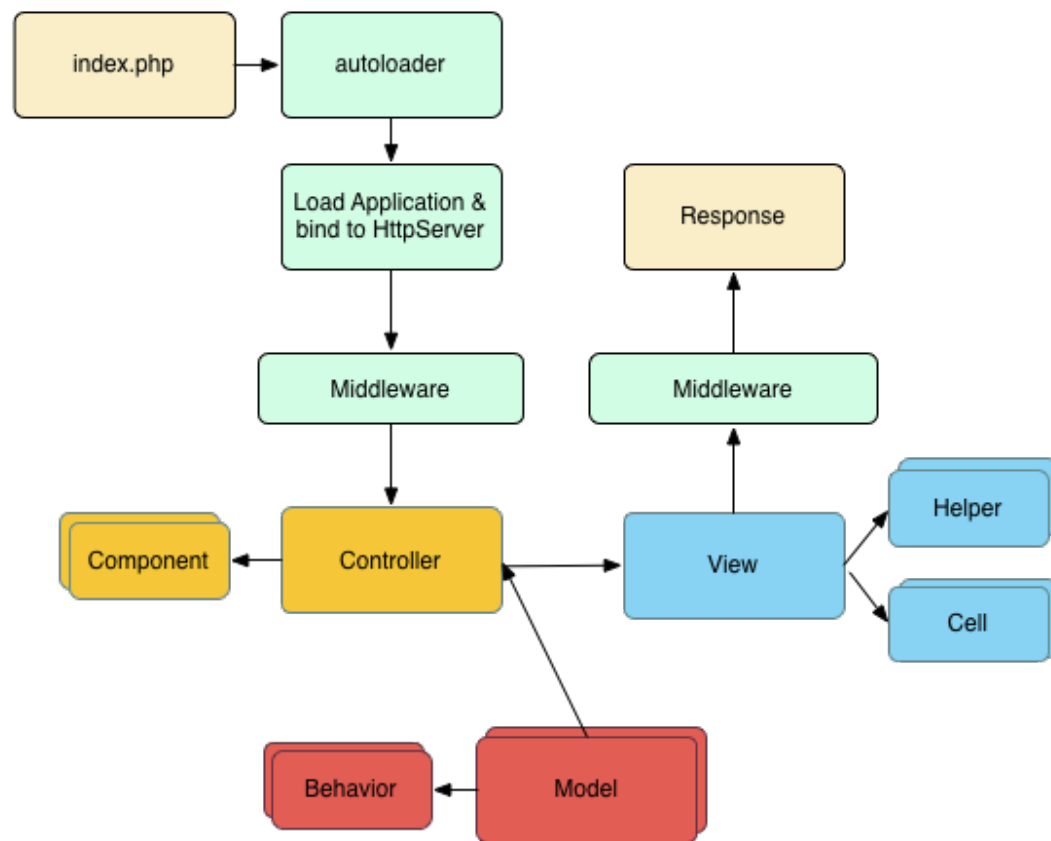
```
public function add() {  
    $user = $this->Users->newEmptyEntity();  
    if ($this->request->is('post')) {  
        $user = $this->Users->patchEntity(  
            $user, $this->request->getData());  
        if ($this->Users->save(  
            $user, ['validate' => 'registration'])) {  
            $this->Flash->success(__(  
                '您現在已註冊。'));  
        } else {  
            $this->Flash->error(__(  
                '存在一些問題。'));  
        }  
    }  
    $this->set('user', $user);  
}
```

CakePHP 請求循環

典型的 CakePHP 請求循環始於使用者請求應用程式中的某個頁面或資源。在高層次上，每個請求會經過以下步驟：

1. 網頁伺服器的重寫規則將請求導向至 webroot/index.php。
2. 你的應用程式被加載並綁定到一個 HttpServer。
3. 應用程式的中介軟體（**middleware**）被初始化。
4. 請求和回應通過應用程式使用的 **PSR-7 中介軟體** 進行分派。通常包括錯誤捕捉和路由。
5. 如果中介軟體未回傳回應，且請求包含路由資訊，則會選擇一個控制器（**Controller**）和動作（**Action**）。
6. 呼叫控制器的動作，控制器與所需的模型（**Model**）和元件（**Component**）進行交互。
7. 控制器將回應的生成委派給視圖（**View**），以根據模型資料生成輸出。
8. 視圖使用輔助工具（**Helpers**）和單元（**Cells**）來生成回應主體和標頭。
9. 回應通過中介軟體發送回去。
10. **HttpServer** 將回應傳送給網頁伺服器。

現在你已經熟悉了 CakePHP 的不同層次，我們來回顧一下 CakePHP 中的請求循環是如何運作的：



從何處獲得幫助

- CakePHP 官方網站 <https://cakephp.org>
- Cookbook <https://book.cakephp.org>
- Bakery <https://bakery.cakephp.org>
- API <https://api.cakephp.org/>
- 測試案例 `tests/TestCase/`
- Slack <https://cakesf.slack.com/?redir=%2Fmessages%2Fgerman%2F>
- Discord <https://discord.com/invite/k4trEMPebj>
- 官方 CakePHP 論壇 <https://discourse.cakephp.org/>
- Stackoverflow <https://stackoverflow.com/>

CakePHP 約定

- 我們是約定優於配置 (**convention over configuration**) 的忠實粉絲。
- 雖然學習 CakePHP 的約定需要一點時間，但從長遠來看，可以**節省時間**。
- 通過遵循約定，可以獲得免費功能，並將自己從追蹤配置檔案的維護噩夢中解放出來。
- 約定還提供了非常統一的開發體驗，允許讓其他開發人員能夠參與並提供協助。

控制器約定

- 控制器類別的名稱使用複數形式、CamelCase (大駝峰命名法)，並以 `Controller` 結尾。
- `UserController` 和 `MenuLinksController` 都是傳統控制器名稱的示例。
- 控制器中的 `public` 方法通常會被作為「actions」透過網頁瀏覽器訪問，這些方法名稱採用 `camelBack` (小駝峰命名法)。
- 例如，`/users/view-me` 會對應到 `UserController` 的 `viewMe()` 方法 (假設路由使用預設的 `dashed inflection` 轉換方式)。
- `protected` 或 `private` 方法無法通過路由進行訪問。

控制器名稱的 URL 注意事項

- 正如你剛才看到的，單字的控制器名稱會對應簡單的小寫 URL 路徑。
- 例如，`UserController` (定義於檔案 `UserController.php`) 可以透過 `http://example.com/users` 訪問。
- 雖然您可以自由定義多字控制器的路由，但慣例是 URL 使用小寫並以短橫線分隔，這通常使用 `DashedRoute` 類別來實現。
- 因此，`/menu-links/view-all` 是訪問 `MenuLinksController::viewAll()` 方法的正確形式。

控制器名稱的 URL 注意事項

當您使用 `$this->Html->link()` 創建鏈接時，可以遵循以下 URL 陣列的命名慣例：

```
$this->Html->link('link-title', [  
    'prefix' => 'MyPrefix', // 大駝峰命名法 (CamelCased)  
    'plugin' => 'MyPlugin', // 大駝峰命名法 (CamelCased)  
    'controller' => 'ControllerName', // 大駝峰命名法 (CamelCased)  
    'action' => 'actionName' // 小駝峰命名法 (camelBacked)  
])
```

如需了解更多關於 CakePHP 的 URL 和參數處理，請參考 [Connecting Routes](#) 的相關文件。

檔案名稱和類別名稱的約定

一般而言，檔案名稱與類別名稱相符，並遵循 **PSR-4 標準** 進行自動載入。以下是一些類別名稱及其對應檔案名稱的範例：

- 控制器類別：`LatestArticlesController` 應位於名為 **LatestArticlesController.php** 的檔案中。
- 元件類別：`MyHandyComponent` 應位於名為 **MyHandyComponent.php** 的檔案中。
- 資料表類別：`OptionValuesTable` 應位於名為 **OptionValuesTable.php** 的檔案中。
- 實體類別：`OptionValue` 應位於名為 **OptionValue.php** 的檔案中。
- 行為類別：`EspeciallyFunkableBehavior` 應位於名為 **EspeciallyFunkableBehavior.php** 的檔案中。
- 視圖類別：`SuperSimpleView` 應位於名為 **SuperSimpleView.php** 的檔案中。
- 輔助類別：`BestEverHelper` 應位於名為 **BestEverHelper.php** 的檔案中。
- 每個檔案都應放置在應用程式資料夾中對應的目錄或命名空間內。

資料庫約定

表格名稱：

- 與 CakePHP 模型對應的資料表名稱採用複數形式並以底線分隔。例如：`users`、`menu_links` 和 `user_favorite_pages`。
- 對於包含多個單詞的資料表名稱，僅將最後一個單詞複數化，例如：`menu_links`。

欄位名稱：

- 包含兩個或更多單詞的欄位名稱應以底線分隔，例如：`first_name`。

外鍵：

- 在 `hasMany`、`belongsTo` 或 `hasOne` 關係中，外鍵預設為相關資料表的單數名稱後接 `_id`。
- 例如，如果 `Users` 有多個 `Articles`，則 `articles` 資料表會透過 `user_id` 外鍵引用 `users` 資料表。
- 對於像 `menu_links` 這種多單詞的資料表，其外鍵名稱會是 `menu_link_id`。

資料庫約定

聯結 (或「交集」) 資料表：

- 在模型的 `BelongsToMany` 關係中，聯結資料表用於連結兩個資料表。
- 聯結資料表的名稱應根據所連結的資料表名稱命名，並且名稱必須為複數形式且按字母順序排列。例如：`articles_tags`，而非 `tags_articles` 或 `article_tags`。
- 如果不遵循此命名規範，`bake` 命令將無法正常運作。
- 如果聯結資料表中包含除了連結外鍵之外的其他資料，則應為該資料表建立具體的實體類別或資料表類別。

主鍵：

- 除了使用自動遞增的整數作為主鍵外，也可以使用 `UUID` 欄位作為主鍵。
- 使用 `Table::save()` 方法保存新記錄時，CakePHP 會自動透過 ([Cake\Utility\Text::uuid\(\)](#)) 生成 `UUID` 值。

模型層約定

資料表類別名稱：

- 資料表類別名稱為複數形式、CamelCase (大駝峰命名法)，並以 Table 結尾。
- 例如：UsersTable、MenuLinksTable 和 UserFavoritePagesTable 分別對應資料表 users、menu_links 和 user_favorite_pages。

實體(Entity)類別名稱：

- 實體類別名稱為單數形式、CamelCase (大駝峰命名法)，且不帶任何後綴。
- 例如：User、MenuLink 和 UserFavoritePage 分別對應資料表 users、menu_links 和 user_favorite_pages。

列舉類別名稱：

- 列舉類別名稱應使用 {Entity}{Column} 的命名方式，列舉案例名稱則應使用 CamelCase (大駝峰命名法)。

視圖層約定

視圖模板檔案：

- 視圖模板檔案的名稱基於其所對應控制器函式的名稱，使用底線分隔形式。
- 例如：`ArticlesController` 的 `viewAll()` 函式會尋找位於 `templates/Articles/view_all.php` 的視圖模板。

基本命名模式：

- `templates/Controller/underscored_function_name.php`



CakePHP 預設使用英語的單複數轉換規則。如果您的資料表或欄位使用其他語言，則需要自定義單複數轉換規則。您可以使用 [Cake\Utility\Inflector](#) 定義自訂的轉換規則。有關更多資訊，請參閱 [Inflector](#) 的官方文件。

套件(Plugins)約定

命名建議：

- 在套件名稱前加上「cakephp-」是有用的，這樣可以讓名稱與其所依賴的框架語義相關聯。

命名空間規則：

- 不要使用 CakePHP 的命名空間 (cakephp) 作為廠商名稱，因為這是保留給 CakePHP 官方套件的。
- 命名時應使用小寫字母，並以短橫線 (dash) 作為分隔符號：

更多資訊：

- 有關詳細建議，請參閱 [awesome list recommendations](#)。

// 錯誤範例

cakephp/foo-bar

// 正確範例

your-name/cakephp-foo-bar

總結

通過使用 **CakePHP** 約定來命名應用程式的各個部分，您可以在不需額外配置的情況下，獲得自動化的功能並減少維護負擔。以下是一個將約定聯繫在一起的完整範例：

- 資料庫表：“articles”，“menu_links”
- 表格類別：ArticlesTable, found at `src/Model/Table/ArticlesTable.php`
- 實體(Entity)類別：Article, found at `src/Model/Entity/Article.php`
- 控制器類別：ArticlesController, found at `src/Controller/ArticlesController.php`
- 視圖模板：found at `templates/Articles/index.php`

使用這些命名約定，**CakePHP** 能夠自動辨識對 `http://example.com/articles` 的請求，並將其對應到 `ArticlesController` 的 `index()` 方法，同時自動載入 `Articles` 模型。這些關聯關係完全不需要額外配置，只需建立必要的類別和檔案即可完成。

總結

每個檔案都應放置於應用程式資料夾中對應的目錄/命名空間下。

範例	articles	menu_links	備註
資料庫表 (Database Table)	articles	menu_links	與 CakePHP 模型對應的表名稱是複數且帶底線。
檔案(File)	ArticlesController.php	MenuLinksController.php	
表格(Table)	ArticlesTable.php	MenuLinksTable.php	表格類別名稱是複數的，用大駝峰命名法，並且以 Table 結尾。
實體(Entity)	Article.php	MenuLink.php	實體類別名稱是單數的，大駝峰命名法：Article 和 MenuLink。
類別(Class)	ArticlesController	MenuLinksController	
控制器 (Controller)	ArticlesController	MenuLinksController	複數，大駝峰命名法，以 Controller 結尾。
模板(Templates)	Articles/index.php Articles/add.php Articles/get_list.php	MenuLinks/index.php MenuLinks/add.php MenuLinks/get_list.php	視圖模板檔案的名稱基於其所對應的控制器函式，並採用底線分隔形式命名。
行為(Behavior)	ArticlesBehavior.php	MenuLinksBehavior.php	
視圖(View)	ArticlesView.php	MenuLinksView.php	
助手(Helper)	ArticlesHelper.php	MenuLinksHelper.php	
元件(Component)	ArticlesComponent.php	MenuLinksComponent.php	
套件(Plugin)	you/cakephp-articles	you/cakephp-articles	建議在套件名稱前加上「cakephp-」，且不要使用 CakePHP 的命名空間 (cakephp) 作為供應商名稱。套件名稱應使用小寫字母並以短橫線分隔。

總結

資料庫約定	
外鍵(Foreign keys) hasMany belongsTo/ hasOne BelongsToMany	關聯關係預設會被識別為相關資料表的單數名稱加上 <code>_id</code> 。例如， <code>Users</code> 與 <code>Articles</code> 是一對多的關係， <code>articles</code> 資料表會透過外鍵 <code>user_id</code> 來參照 <code>users</code> 資料表。
多個單詞(Multiple Words)	對於名稱包含多個單字的資料表，例如 <code>menu_links</code> ，其外鍵將會是 <code>menu_link_id</code> 。
自動遞增(Auto Increment)	除了使用自動遞增的整數作為主鍵之外，您也可以使用 UUID 欄位作為主鍵。當您使用 <code>Table::save()</code> 方法儲存新紀錄時，CakePHP 會自動透過 (Cake\Utility\Text::uuid()) 生成 UUID 值。
聯接表格(Join tables)	應以所連接的模型資料表命名，並按字母順序排列（例如 <code>articles_tags</code> 而非 <code>tags_articles</code> ），否則 <code>bake</code> 指令將無法運作。 如果交集表中包含額外的欄位，則應為該表建立一個獨立的實體/資料表類別。

CakePHP 資料夾結構

確保 **tmp** 和 **logs** 資料夾存在且可寫入，否則會嚴重影響應用程式的效能。在除錯模式下，如果這些資料夾無法寫入，CakePHP 會發出警告。

在您下載 CakePHP 的應用程式骨架後，您會看到幾個頂層資料夾：

- **bin**：存放 CakePHP 的控制台可執行檔。
- **config**：存放 CakePHP 使用的[設定檔](#)，例如資料庫連線詳細資訊、啟動設定、核心設定檔等。
- **plugins**：存放應用程式使用的[外掛程式/套件](#)。
- **logs**：通常存放您的日誌檔案，具體內容取決於日誌設定。
- **src**：應用程式的原始碼檔案將存放於此。
- **templates**：存放展示用的檔案，例如元素檔、錯誤頁面、佈局和視圖模板檔案。
- **resources**：包含各類資源檔案的子資料夾。**locales** 子資料夾用於儲存國際化的語言檔案。
- **tests**：存放應用程式的測試案例。
- **tmp**：CakePHP 存放臨時資料的地方，具體存放內容取決於您的設定，通常用於翻譯訊息、模型描述以及有時的會話資訊。
- **vendor** 資料夾：Composer 安裝的 CakePHP 和其他應用程式相依性檔案存放於此。建議不要編輯這些檔案，因為 Composer 更新時會覆蓋您的更改。
- **webroot** 資料夾：應用程式的公開文件根目錄，包含所有希望公開訪問的檔案。

src 資料夾

CakePHP 的 `src` 資料夾是您進行大部分應用程式開發的地方。以下是 `src` 資料夾內的子資料夾說明：

- **Command**：包含應用程式的控制台指令。詳情請參考 [Command Objects](#)。
- **Console**：包含由 `Composer` 執行的安裝腳本。
- **Controller**：存放應用程式的[控制器](#)和其元件。
- **Middleware**：儲存應用程式的[中介軟體](#)。
- **Model**：包含應用程式的資料表、實體和行為。
- **View**：展示類別存放於此，包括視圖、細胞、輔助工具等。



預設情況下，**Command** 資料夾並不存在。當您需要時可以自行新增。