

2026 CakePHP-Day4

CakePHP CMS 教學2

編輯文章

- 加入以下修改：

```
public function edit(?string $slug = null)
{
    $article = $this->Articles->findBySlug($slug)->firstOrFail();
    if ($this->request->is(['post', 'put'])) {
        $this->Articles->patchEntity($article, $this->request->getData());
        if ($this->Articles->save($article)) {
            $this->Flash->success(__('您的文章已經更新。'));
            return $this->redirect(['action' => 'view', $article->slug]);
        }
        $this->Flash->error(__('無法更新您的文章。'));
    }
    $this->set('article', $article);
}
```

ArticlesController.php

使用者 (Users)

- 完成了基本的文章建立功能後，我們需要讓多個作者能夠在 CMS 中協作。
- 之前，我們是手動建立所有的模型、視圖和控制器。這次，我們將使用 **Bake Console** 來生成骨架程式碼。
- **Bake** 是一個強大的程式碼生成命令列介面(CLI, Command Line Interface)，利用 **CakePHP** 的約定，能非常高效地建立增刪查改(CRUD, Create Read Update Delete)應用程式的框架程式碼。
- 我們將使用 **bake** 來建立 **Users** 的程式碼：

```
cd /CMS
```

```
bash
```

```
# 可以覆寫任何已存在的檔案。
```

```
bin/cake bake model users
```

```
bin/cake bake controller users
```

```
bin/cake bake template users
```

使用者 (Users)

```
cd /CMS
```

```
bash
```

```
# 可以覆寫任何已存在的檔案。
```

```
bin/cake bake model users
```

```
bin/cake bake controller users
```

```
bin/cake bake template users
```

這三條指令將生成：

1. 資料表類別、實體類別和測試資料檔案：

- 包括資料表 (Table)、實體 (Entity)、測試資料檔案 (Fixture)。

2. 控制器：

- 自動生成 UsersController，包含基本的 CRUD 操作。

3. CRUD 範本模板：

- 生成基本的 Create、Read、Update 和 Delete 模板檔案。

4. 每個生成類別的測試案例：

- 包含自動生成的單元測試，用於測試模型、控制器和模板功能是否正常。

Bake 的優點

1. 自動推斷關聯與驗證規則：

- Bake 會根據 CakePHP 的慣例，自動推斷模型之間的關聯（如一對多、多對多等），以及模型的驗證規則。

2. 高效生成基礎程式碼：

- 適合快速構建應用程式骨架，然後手動調整生成的程式碼以符合特定需求。

透過 Bake Console，我們可以快速建立 Users 的相關功能，後續將進一步整合標籤 (Tags) 功能並完成整體系統的開發。

標籤 (Tags)

新增範例標籤

完成程式碼生成後，可以通過瀏覽以下網址，新增一些範例標籤：
<http://localhost:8080/tags/add>

- 現在多個使用者可以訪問我們的小型 CMS，我們可以新增一種分類文章內容的方法。
- 透過使用標籤 (Tags) 功能，使用者可以為內容自由建立分類標籤。
- 我們將再次使用 Bake 快速生成一些程式碼骨架：

```
# 一次生成所有相關程式碼。  
bin/cake bake all tags
```

bash

執行以上指令後，Bake 將會生成：

1. 模型 (Model) : TagsTable 與 Tag 實體類別。
2. 控制器 (Controller) : TagsController，處理標籤的 CRUD 操作。
3. 模板 (Template) : 包含 add、edit、view 和 index 模板檔案，用於顯示和操作標籤。
4. 測試 (Tests) : 測試案例檔案，用於驗證標籤功能。

標籤 (Tags)

更多資訊

如需詳細了解關聯設定，請參考官方文件：[Associations - Linking Tables Together](#)

透過這種方式，我們可以輕鬆實現文章的標籤功能，讓使用者能夠更好地組織與分類內容。

建立文章與標籤的關聯

- 現在，我們已經有了 **Tags** 資料表，接下來需要將 **Articles** 和 **Tags** 建立關聯。只需在 **ArticlesTable** 的 **initialize** 方法中新增以下程式碼：

```
// 加入到現有的 src/Model/Table/ArticlesTable.php 檔案
```

ArticlesTable.php

```
public function initialize(array $config): void
{
    $this->addBehavior('Timestamp');
    $this->belongsToMany('Tags'); // 新增這行
}
```

如何運作

此關聯能夠正常運作的原因是我們遵循了 CakePHP 的資料表命名慣例：

- 1. 聯結資料表 (Join Table)：** CakePHP 自動假設我們有一個命名為 **articles_tags** 的聯結資料表，用於表示多對多關聯。
- 2. 欄位命名：** **articles_tags** 資料表包含 **article_id** 和 **tag_id** 欄位，指向 **Articles** 和 **Tags** 資料表的主鍵。

標籤 (Tags)

// 加入到現有的 `src/Controller/ArticlesController.php` 檔案

`ArticlesController.php`

```
namespace App\Controller;
class ArticlesController extends ApplicationController {
    // 一些程式碼...
    public function add() {
        // 一些程式碼...

        // 取得標籤清單
        $tags = $this->Articles->Tags->find('list')->all();
        // 將標籤傳遞到視圖上下文
        $this->set('tags', $tags);

        $this->set('article', $article);
    }
    // 其他方法
}
```

更新文章功能以啟用標籤

- 現在我們的應用程式已經有了標籤功能，接下來需要讓使用者能夠為文章新增標籤。
- 將 `add` 方法更新為此頁內容。

標籤 (Tags)

更新文章功能以啟用標籤

- 現在我們的應用程式已經有了標籤功能，接下來需要讓使用者能夠為文章新增標籤。以下是具體步驟：
- 在 `templates/Articles/add.php` 的 PHP 區塊中，新增以下程式碼：

```
<!-- 加入到現有的 templates/Articles/add.php 檔案 -->
```

add.php

```
<h1>新增文章</h1>
```

```
<?php
```

```
    echo $this->Form->create($article);
```

```
    // 暫時寫死使用者。
```

```
    echo $this->Form->control('user_id', ['type' => 'hidden', 'value' => 1]);
```

```
    echo $this->Form->control('title', ['label' => '標題']);
```

```
    echo $this->Form->control('tags._ids', ['label' => '標籤', 'options' => $tags]);
```

```
    echo $this->Form->control('body', ['label' => '內文', 'rows' => '3']);
```

```
    echo $this->Form->button(__('儲存文章'));
```

```
    echo $this->Form->end();
```

```
?>
```

- 這段程式碼會生成一個多選下拉框，使用 `$tags` 變數來生成選項。

標籤 (Tags)

// 加入到現有的 `src/Controller/ArticlesController.php` 檔案

`ArticlesController.php`

```
namespace App\Controller;
class ArticlesController extends ApplicationController {
    // 一些程式碼...
    public function edit($slug) {
        $article = $this->Articles
            ->findBySlug($slug)->contain('Tags')->firstOrFail(); // 載入關聯的標籤

        // 一些程式碼...

        $tags = $this->Articles->Tags->find('list')->all();
        $this->set('tags', $tags);

        $this->set('article', $article);
    }
    // 其他方法
}
```

更新文章功能以啟用標籤

- 現在我們的應用程式已經有了標籤功能，接下來需要讓使用者能夠為文章新增標籤。
- 為了在編輯文章時支援標籤，將 `edit` 方法更新如此頁。

標籤 (Tags)

更新文章功能以啟用標籤

- 現在我們的應用程式已經有了標籤功能，接下來需要讓使用者能夠為文章新增標籤。以下是具體步驟：
- 在 `templates/Articles/edit.php` 模板中，新增與新增文章相同的標籤多選下拉框：

```
<!-- 加入到現有的 templates/Articles/edit.php 檔案 -->
```

edit.php

```
<h1>編輯文章</h1>
```

```
<?php
```

```
    echo $this->Form->create($article);
```

```
    echo $this->Form->control('user_id', ['type' => 'hidden']);
```

```
    echo $this->Form->control('title', ['label' => '標題']);
```

```
    echo $this->Form->control('tags._ids', ['label' => '標籤', 'options' => $tags]);
```

```
    echo $this->Form->control('body', ['label' => '內文', 'rows' => '3']);
```

```
    echo $this->Form->button(__('儲存文章'));
```

```
    echo $this->Form->end();
```

```
?>
```

測試功能

- 現在可以嘗試新增或編輯文章，同時為文章添加標籤。

標籤 (Tags)

根據標籤篩選文章

- 當使用者為內容分類後，他們可能希望透過標籤來篩選文章。為此，我們將實作一個路由、控制器動作和 `Finder` 方法，實現這個功能。

設置路由

- 理想情況下，我們希望能使用以下類似的 URL 格式：
`http://localhost:8080/articles/tagged/funny/cat/gifs`
- 這樣可以找到所有具有「`funny`」、「`cat`」或「`gifs`」標籤的文章。

標籤 (Tags)

```
<?php
// 檔案: config/routes.php
use Cake\Routing\Route\DashedRoute;
use Cake\Routing\RouteBuilder;

$routes->setRouteClass(DashedRoute::class);

$routes->scope('/', function (RouteBuilder $builder) {
    $builder->connect('/', ['controller' => 'Pages', 'action' => 'display', 'home']);
    $builder->connect('/pages/*', ['controller' => 'Pages', 'action' => 'display']);

    // 新增以下路由
    // 新的路由，用於實作 tagged 動作。
    // 尾部的 `*` 表示該動作會接收參數。
    $builder->scope('/articles', function (RouteBuilder $builder) {
        $builder->connect('/tagged/*', ['controller' => 'Articles', 'action' => 'tags']);
    });

    $builder->fallbacks();
});
```

routes.php

- 首先，更新 config/routes.php，修改後的內容如此頁

標籤 (Tags)

- 首先，更新 `config/routes.php`，修改後的內容如下：

```
// 新增以下路由
// 新的路由，用於實作 tagged 動作。
// 尾部的 `*` 表示該動作會接收參數。
$builder->scope('/articles', function (RouteBuilder $builder) {
    $builder->connect('/tagged/*', ['controller' => 'Articles', 'action' => 'tags']);
});
```

`routes.php`

路由功能解釋

- 以上路由定義了一個新的路由，將 `/articles/tagged/*` 路由對應到 `ArticlesController::tags()` 動作。
- 定義路由的好處是可以將 URL 的外觀與其實際實作分離。如果訪問 `http://localhost:8080/articles/tagged`，CakePHP 會提供一個錯誤頁面，提示控制器動作不存在。

標籤 (Tags)

新增 tags() 方法

- 接下來，在 `src/Controller/ArticlesController.php` 中新增以下方法：

```
public function tags()  
{  
    // `pass` 是 CakePHP Request 中保存已傳遞路徑參數的 key，  
    // 內容為 URL 中未命名的 positional arguments。  
    $tags = $this->request->getParam('pass');  
  
    // 使用 ArticlesTable 搜尋具有指定標籤的文章。  
    $articles = $this->Articles->find('tagged', tags: $tags)->all();  
  
    // 將變數傳遞給視圖模板上下文。  
    $this->set([  
        'articles' => $articles,  
        'tags' => $tags  
    ]);  
}
```

`ArticlesController.php`

- 請參考 [CakePHP 文件](#) 以了解如何訪問其他請求數據：
[Request 文件](#)。

標籤 (Tags)

使用可變參數的替代方法

- 由於傳遞的參數會以方法參數的形式傳遞，你也可以使用 PHP 的可變參數語法 (variadic argument) 來撰寫此動作：

```
public function tags(...$tags)
{
    // 使用 ArticlesTable 搜尋具有指定標籤的文章。
    $articles = $this->Articles->find('tagged', tags: $tags)
        ->all();

    // 將變數傳遞給視圖模板上下文。
    $this->set([
        'articles' => $articles,
        'tags' => $tags
    ]);
}
```

ArticlesController.php

標籤 (Tags)

建立 Finder 方法

- 在 CakePHP 中，我們傾向讓控制器的動作簡潔，並將應用程式的大部分邏輯放在模型層。如果現在訪問 `/articles/tagged` URL，會看到一個錯誤，提示 `findTagged()` 方法尚未實作。我們來實作這個方法。
- 在 `src/Model/Table/ArticlesTable.php` 中，新增以下程式碼：

標籤 (Tags)

ArticlesTable.php

```
// 在命名空間聲明後，新增以下 use 語句以引入 Query 類別
use Cake\ORM\Query\SelectQuery;

// $query 參數是一個查詢構建器實例。
// $tags 陣列會包含我們在控制器動作中傳遞給 find('tagged') 的 'tags' 選項。
public function findTagged(SelectQuery $query, array $tags = []): SelectQuery
{
    $columns = [
        'Articles.id', 'Articles.user_id', 'Articles.title', 'Articles.body', 'Articles.published',
        'Articles.created', 'Articles.slug',
    ];

    $query = $query->select($columns)->distinct($columns);

    if (empty($tags)) {
        // 如果沒有提供標籤，則查詢沒有標籤的文章。
        $query->leftJoinWith('Tags')->where(['Tags.title IS' => null]);
    } else {
        // 查詢包含一個或多個提供標籤的文章。
        $query->innerJoinWith('Tags')->where(['Tags.title IN' => $tags]);
    }

    return $query->groupBy(['Articles.id']);
}
```

標籤 (Tags)

方法說明

以上程式碼實作了一個自訂的 `Finder` 方法。這是 `CakePHP` 中非常強大的概念，允許將可重用的查詢封裝起來。以下是關鍵點：

1. `Finder` 方法的參數：

`$query` 是查詢構建器實例，提供了查詢操作的所有方法。

`$tags` 是選項陣列，包含從控制器傳遞的標籤數據。

2. 查詢操作：

使用 `distinct()` 確保回傳的文章是唯一的。

使用 `leftJoinWith()` 查找沒有標籤的文章。

使用 `innerJoinWith()` 查找具有指定標籤的文章。

3. 條件處理：

如果沒有提供任何標籤 (`empty($tags)`)，則查詢所有沒有標籤的文章。

如果提供了標籤，則查詢符合任一提供標籤的文章。

4. 回傳結果：

最後，回傳分組 (`groupBy()`) 後的查詢物件，以避免重複的文章出現在結果中。

標籤 (Tags)

Finder 方法的應用

Finder 方法使查詢邏輯更具模組化和可重用性。完成實作後，控制器中的 `find('tagged')` 調用將自動執行此查詢邏輯，無需在控制器中手動處理條件。

透過此方法，使用者現在可以根據標籤篩選文章，例如：

- 查找沒有標籤的文章。
- 查找包含一個或多個指定標籤的文章（例如 `funny`、`cat` 或 `gifs`）。

標籤 (Tags)

建立 tags View Template

現在，如果再次訪問 `/articles/tagged`，CakePHP 會顯示一個新錯誤，提示尚未建立 `view` 檔案。接下來，我們將為 `tags()` 動作建立 `view` 檔案：

```
<!-- 位於 templates/Articles/tags.php -->
<h1>標籤為 <?= $this->Text->toList(h($tags), '或') ?> 的文章</h1>

<section>
<?php foreach ($articles as $article): ?>
    <article>
        <!-- 使用 HtmlHelper 來建立連結 -->
        <h4><?= $this->Html->link(
            $article->title,
            ['controller' => 'Articles', 'action' => 'view', $article->slug]
        ) ?></h4>
        <span><?= h($article->created) ?></span>
    </article>
<?php endforeach; ?>
</section>
```

tags.php

標籤 (Tags)

1. Html 和 Text Helpers

`HtmlHelper`：用於生成 HTML 元素，如超連結。

`TextHelper`：用於處理文本內容。例如，`toList()` 方法可以將陣列轉換為可讀格式。

2. 防止 HTML 注入

使用 `h()` 函數對輸出進行 HTML 編碼，避免 HTML 注入攻擊。

3. View 檔案的命名規範

根據 `CakePHP` 的規範，`view` 模板檔案的名稱應為控制器動作名稱的小寫並使用底線命名法（例如 `tags()` 的檔案名為 `tags.php`）。

4. 變數的使用

`$tags` 和 `$articles` 變數是在控制器中透過 `set()` 方法傳遞的，這些變數會自動在 `view` 模板中以區域變數的形式可用。

標籤 (Tags)

預期結果

- 現在，當您訪問 `/articles/tagged/funny` 時，應該能看到所有被標籤為「`funny`」的文章。

標籤 (Tags)

改善標籤使用體驗

- 目前，新增標籤的流程較為繁瑣，因為作者需要事先建立所有想要使用的標籤。我們可以通過以逗號分隔的文字輸入欄位來改善標籤選擇的使用者介面。這不僅能提供更好的使用體驗，還能利用 **ORM** 的更多強大功能。

標籤 (Tags)

新增計算欄位

為了能夠簡單地取得實體的格式化標籤，我們可以在實體中新增一個虛擬 / 計算屬性 (computed property)。在 `src/Model/Entity/Article.php` 中新增以下程式碼：

此程式碼讓我們能夠使用 `$article->tag_string` 計算屬性。我們稍後將在控制項中使用此屬性。

```
// 在 namespace 宣告下方新增以下 use 語句來匯入 Collection 類別
use Cake\Collection\Collection;

// 更新 accessible 屬性以包含 `tag_string`
protected array $_accessible = [
    // 其他欄位...
    'tag_string' => true
];

protected function _getTagString(): string
{
    if (isset($this->_fields['tag_string'])) {
        return $this->_fields['tag_string'];
    }
    if (empty($this->tags)) {
        return '';
    }
    $tags = new Collection($this->tags);
    $str = $tags->reduce(function ($string, $tag) {
        return $string . $tag->title . ', ';
    }, '');

    return trim($str, ', ');
}
```

標籤 (Tags)

更新 View

現在我們已經更新了實體，可以為標籤新增一個新的控制項。在 `templates/Articles/add.php` 和 `templates/Articles/edit.php` 中，將現有的 `tags._ids` 控制項替換為以下程式碼：

```
<!-- 檔案: templates/Articles/add.php & edit.php --> add.php | edit.php  
echo $this->Form->control('tag_string', ['label' => '標籤', 'type' => 'text']);
```

我們還需要更新文章檢視模板。在 `templates/Articles/view.php` 中，新增以下程式碼：

```
<!-- 檔案: templates/Articles/view.php --> view.php  
  
<h1><?= h($article->title) ?></h1>  
<p><?= h($article->body) ?></p>  
// 新增以下行  
<p><b>標籤:</b> <?= h($article->tag_string) ?></p>
```

標籤 (Tags)

更新 view

此外，還需要更新 `view()` 動作，讓查詢一併載入既有 `Tags` 關聯資料。：

```
// 檔案：src/Controller/ArticlesController.php
```

ArticlesController.php

```
public function view($slug = null)
{
    // 使用 contain() 檢索標籤
    $article = $this->Articles
        ->findBySlug($slug)
        ->contain('Tags')
        ->firstOrFail();
    $this->set(compact('article'));
}
```

完成後，新增和編輯文章時可以使用簡潔的文字欄位來輸入標籤，並在文章詳細檢視頁面中顯示標籤列表。

標籤 (Tags)

儲存標籤字串

- 現在我們可以將現有標籤顯示為字串，但我們還需要將這些資料儲存起來。
- 因為我們已經將 `tag_string` 標記為可存取，ORM 會自動將請求中的資料複製到實體中。
- 我們可以使用 `beforeSave()` 鉤點方法(Hook Method)來解析標籤字串並找到/建立相關的實體。

將以下程式碼新增到 `src/Model/Table/ArticlesTable.php` 中：

標籤 (Tags)

ArticlesTable.php

```
// 檔案: src/Model/Table/ArticlesTable.php

public function beforeSave(EventInterface $event, $entity,
$options): void
{
    if ($entity->tag_string) {
        $entity->tags = $this->_buildTags(
            $entity->tag_string);
    }

    // 其他程式碼
}

protected function _buildTags($tagString): array
{
    // 修剪標籤
    $newTags = array_map('trim', explode(',', $tagString));
    // 移除所有空的標籤
    $newTags = array_filter($newTags);
    // 去除重複的標籤
    $newTags = array_unique($newTags);

    $out = [];
    $tags = $this->Tags->find()
        ->where(['Tags.title IN' => $newTags])
        ->all();

    // 從新標籤列表中移除已存在的標籤
    foreach ($tags->extract('title') as $existing) {
        $index = array_search($existing, $newTags);
        if ($index !== false) {
            unset($newTags[$index]);
        }
    }
    // 添加已存在的標籤
    foreach ($tags as $tag) {
        $out[] = $tag;
    }
    // 添加新標籤
    foreach ($newTags as $tag) {
        $out[] = $this->Tags->newEntity(['title' => $tag]);
    }

    return $out;
}
```

標籤 (Tags)

功能說明

現在，當您建立或編輯文章時，可以使用逗號分隔的標籤列表儲存標籤。系統會自動建立標籤以及連結的關聯記錄。

儘管這段程式碼比我們之前的程式碼稍微複雜，但它展示了 **CakePHP ORM** 的強大功能。您可以利用集合方法來操作查詢結果，並輕鬆處理需要在執行過程中動態建立 **Entity** 的情境。

標籤 (Tags)

自動填入標籤字串

在完成之前，我們需要一個機制，在載入文章時自動載入標籤並產生 `tag_string` (如果有的話)。

在 `src/Model/Table/ArticlesTable.php` 中，修改以下內容：

```
// 檔案 : src/Model/Table/ArticlesTable.php
```

ArticlesTable.php

```
public function initialize(array $config): void
{
    $this->addBehavior('Timestamp');
    // 修改這一行
    $this->belongsToMany('Tags', [
        'joinTable' => 'articles_tags',
        'dependent' => true
    ]);
}
```

此設定告訴 `Articles` 表模型存在與標籤相關聯的中間表。`dependent` 選項表示，刪除 `Article` 時，同步清除 `Join Table` 中與該 `Article` 相關的關聯記錄。

標籤 (Tags)

更新 findBySlug() 方法

最後，在 `src/Controller/ArticlesController.php` 中更新 `findBySlug()` 方法的呼叫：

```
// 檔案：src/Controller/ArticlesController.php
public function edit($slug)
{
    // 更新這一行
    $article = $this->Articles
        ->findBySlug($slug)
        ->contain('Tags')
        ->firstOrFail();
    ...
}

ArticlesController.php
public function view($slug = null)
{
    // 更新這一行
    $article = $this->Articles
        ->findBySlug($slug)
        ->contain('Tags')
        ->firstOrFail();
    $this->set(compact('article'));
}
```

功能說明

`contain('Tags')` 告訴 ORM 在載入 `Article` 時一併載入 `Tags` 關聯資料。現在當 `Article` 實體的 `tag_string` 被呼叫時，將有足夠的資料來生成標籤字串！