CakePHP Cookbook Documentation

Publicación 3.x

Cake Software Foundation

ene. 21, 2017

Índice general

1. CakePHP at a Glance **1**
Additional Reading . . . 1

2. Quick Start Guide **5**
Bookmarker Tutorial . . . 5
Bookmarker Tutorial Part 2 . . . 5

3. 3.0 Migration Guide **7**

4. Tutoriales y Ejemplos **9**
Bookmarker Tutorial . . . 9
Bookmarker Tutorial Part 2 . . . 9
Tutorial de desarrollo del Blog . . . 10
Tutorial de desarrollo del Blog - Añadiendo una capa . . . 14
Tutorial de desarrollo de Blog - Parte 3 . . . 26
Tutorial de aplicación Blog - Autenticación y Autorización . . . 31

5. Contribuye **41**
Documentación . . . 41
Tickets . . . 42
Code . . . 42
Coding Standards . . . 42
Backwards Compatibility Guide . . . 43

6. Instalación **45**
Requisitos . . . 45
Licencia . . . 46
Instalando CakePHP . . . 46
Permisos . . . 47
Configuración . . . 47

Desarrollo . . . 47
Producción . . . 48
A rodar! . . . 49
URL Rewriting . . . 49

7. Configuration **55**

8. Routing **57**
Connecting Routes . . . 57

9. Request & Response Objects **59**

10. Controllers **61**
More on Controllers . . . 61

11. Views **65**
More About Views . . . 65

12. Acceso a la base de datos & ORM **71**
Ejemplo rápido . . . 71
Más información . . . 73

13. Bake Console **85**
Code Generation with Bake . . . 85
Extending Bake . . . 87

14. Caching **89**

15. Shells, Tasks & Console Tools **91**
More Topics . . . 91

16. Debugging **95**

17. ES - Deployment **97**

18. Email **99**

19. Error & Exception Handling **101**

20. Events System **103**

21. Internationalization & Localization **105**

22. Logging **107**

23. Modelless Forms **109**

24. Plugins **111**

25. REST **113**

26. Security **115**
Security . . . 115

27. Sessions **117**

28. Testing **119**

29. Validation **121**

30. App Class **123**

31. Collections **125**

32. Folder & File **127**

33. Hash **129**

34. Http Client **131**

35. Inflector **133**

36. Number **135**

37. Registry Objects **137**

38. Text **139**

39. Time **141**

40. Xml **143**

41. Constants & Functions **145**

42. Debug Kit **147**

43. Migrations **149**

44. Apéndices **151**
Guía de Migración a 3.x . . . 151
Información General . . . 157

PHP Namespace Index **159**

Índice **161**

CAPÍTULO 1

CakePHP at a Glance

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[1] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Additional Reading

Where to Get Help

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[2] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

CakePHP Conventions

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

[1] https://github.com/cakephp/docs
[2] https://github.com/cakephp/docs

Por favor, siéntase libre de enviarnos un pull request en Github[3] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

CakePHP Folder Structure

Después de haber descargado y extraido la aplicación CakePHP, estos son los archivos y directorios que podrás ver:

- bin
- config
- logs
- plugins
- src
- tests
- tmp
- vendor
- webroot
- .htaccess
- composer.json
- index.php
- README.md

Notarás unos cuantos directorios de primer nivel:

- La carpeta *bin* contiene los ejecutables por consola de Cake.
- La carpeta *config* contiene los documentos de *Configuration* que utiliza CakePHP. Detalles de la conexión a la Base de Datos, bootstrapping, arhivos de configuración del core y otros, serán almacenados aquí.
- La carpeta *plugins* es donde se almacenan los *Plugins* que utiliza tu aplicación.
- La carpeta de *logs* contiene normalmente tus archivos de log, dependiendo de tu configuración de log.
- La carpeta *src* será donde tu crearás tu mágia: es donde se almacenarán los archivos de tu aplicación.
- La carpeta *tests* será donde pondrás los test para tu aplicación.

[3] https://github.com/cakephp/docs

- La carpeta *tmp* es donde CakePHP almacenará temporalmente la información. La información actual que almacenará dependerá de cómo se configure CakePHP, pero esta carpeta es normalmente utilizada para almacenar descripciones de modelos y a veces información de sesión.
- La carpeta *vendor* es donde CakePHP y otras dependencias de la aplicación serán instaladas. Comprométete a **no** editar los archivos de esta carpeta. No podremos ayudarte si modificas el core.
- El directorio *webroot* es la raíz de los documentos públicos de tu aplicación. Contiene todos los archivos que quieres que sean accesibles públicamente.

Asegúrate de que las carpetas *tmp* y *logs* existen y permiten escritura, en caso contrario el rendimiento de tu aplicación se verá gravemente perjudicado. En modo debug, CakePHP te avisará si este no es el caso.

La carpeta src

La carpeta *src* de CakePHP es donde tú harás la mayor parte del desarrollo de tu aplicación. Observemos más detenidamente dentro de la carpeta *src*.

Console Contiene los comandos de consola y las tareas de consola de tu aplicación. Para más información mirar *Shells, Tasks & Console Tools*.

Controller Contiene los controladores de tu aplicación y sus componentes.

Locale Almacena los ficheros de string para la internacionalización.

Model Contiene las tablas, entidades y funcionamiento de tu aplicación.

View Las clases de presentación se ubican aquí: cells, helpers y templates.

Template Los archivos de presentación se almacenan aquí: elementos, páginas de error, layouts, y templates.

CAPÍTULO 2

Quick Start Guide

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[4] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Bookmarker Tutorial

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[5] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Bookmarker Tutorial Part 2

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

[4] https://github.com/cakephp/docs
[5] https://github.com/cakephp/docs

Por favor, siéntase libre de enviarnos un pull request en Github[6] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[6] https://github.com/cakephp/docs

CAPÍTULO 3

3.0 Migration Guide

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[7] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[7] https://github.com/cakephp/docs

CAPÍTULO 4

Tutoriales y Ejemplos

En esta sección puedes encontrar varias aplicaciones completas construidas en CakePHP que te ayudarán a comprender el framework y ver cómo se relacionan todas las piezas.

También puedes ver otros ejemplos en: CakePackages[8] y en Bakery[9] encontrarás también componentes listos para usar.

Bookmarker Tutorial

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[10] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Bookmarker Tutorial Part 2

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[11] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

[8] https://plugins.cakephp.org/
[9] https://bakery.cakephp.org/
[10] https://github.com/cakephp/docs
[11] https://github.com/cakephp/docs

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Tutorial de desarrollo del Blog

Bienvenido a CakePHP. Probablemente estás consultando este tutorial porque quieres aprender más sobre cómo funciona CakePHP. Nuestro objetivo es potenciar tu productividad y hacer más divertido el desarrollo de aplicaciones. Esperamos que puedas comprobarlo a medida que vas profundizando en el código.

Este tutorial te guiará en la creación de una aplicación sencilla de blog. Obtendremos e instalaremos CakePHP, crearemos y configuraremos la base de datos y añadiremos suficiente lógica como para listar, añadir, editar y eliminar artículos del blog.

Esto es lo que necesitarás:

1. Servidor web funcionando. Asumiremos que estás usando Apache, aunque las instrucciones para otros servidores son similares. Igual tendremos que ajustar un poco la configuración inicial, pero la mayoría pueden poner en marcha CakePHP sin configuración alguna. Asegúrate de tener PHP 5.5.9 o superior así como tener las extensiones `mbstring`, `intl` y `mcrypt` activadas en PHP.
2. Servidor de base de datos. Usaremos MySQL en este tutorial. Necesitarás saber cómo crear una base de datos nueva. CakePHP se encargará del resto. Dado que utilizamos MySQL, asegúrate también de tener `pdo_mysql` habilitado en PHP.
3. Conocimientos básicos de PHP.

¡Vamos allá!

Obtener CakePHP

La manera más sencilla de ponerse en marcha es utilizando Composer. Composer te permite instalar fácilmente CakePHP desde tu terminal o consola. Primero, debes descargar e instalar Composer si todavía no lo has hecho. Si tienes cURL instalado, es tan fácil como ejecutar lo siguiente:

curl -s https://getcomposer.org/installer | php

O puedes descargar `composer.phar` desde la página web de Composer[12].

Instalando Composer de manera global evitarás tener que repetir este paso para cada proyecto.

Luego, simplemente escribe la siguiente línea en tu terminal desde tu directorio de instalación para instalar el esqueleto de la aplicación de CakePHP en el directorio [nombre_app].

```
php composer.phar create-project --prefer-dist cakephp/app [nombre_app]
```

O si tienes Composer instalado globalmente:

```
composer create-project --prefer-dist cakephp/app [nombre_app]
```

[12] https://getcomposer.org/download/

La ventaja de utilizar Composer es que automáticamente completará algunas tareas de inicialización, como aplicar permisos a ficheros y crear tu fichero config/app.php por ti.

Existen otros modos de instalar CakePHP si no te sientes cómodo con Composer. Para más información revisa la sección *Instalación*.

Dejando de lado cómo has descargado e instalado CakePHP, una vez ha terminado la configuración, tu directorio de instalación debería tener la siguiente estructura:

```
/directorio_raiz
    /config
    /logs
    /src
    /plugins
    /tests
    /tmp
    /vendor
    /webroot
    .gitignore
    .htaccess
    .travis.yml
    README.md
    composer.json
    phpunit.xml.dist
```

Quizás sea buen momento para aprender algo sobre cómo funciona esta estructura de directorios: echa un vistazo a la sección *CakePHP Folder Structure*.

Permisos de directorio en tmp

También necesitarás aplicar los permisos adecuados en el directorio `/tmp` para que el servidor web pueda escribir en él. El mejor modo de hacer esto es encontrar con qué usuario corre tu servidor web (`<?= \`whoami\`; ?>`) y cambiar la propiedad del directorio `tmp` hacia dicho usuario. El comando final que ejecutarás (en *nix) se parecerá al siguiente:

```
$ chown -R www-data tmp
```

Si por alguna razón CakePHP no puede escribir en ese directorio, serás informado mediante una alerta mientras no estés en modo producción.

A pesar de que no se recomienda, si no eres capaz de aplicar la propiedad del directorio al mismo usuario que el servidor web, puedes simplemente aplicar permisos de escritura al directorio ejecutando un comando tipo:

```
$ chmod 777 -R tmp
```

Creando la base de datos del Blog

Vamos a crear una nueva base de datos para el blog. Puedes crear una base de datos en blanco con el nombre que quieras. De momento vamos a definir sólo una tabla para nuestros artículos ("posts"). Además crearemos

algunos artículos de test para usarlos luego. Una vez creada la tabla, ejecuta el siguiente código SQL en ella:

```
/* Primero, creamos la tabla artículos: */
CREATE TABLE articles (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(50),
    body TEXT,
    created DATETIME DEFAULT NULL,
    modified DATETIME DEFAULT NULL
);


/* Luego insertamos algunos artículos para probar */
INSERT INTO articles (title,body,created)
    VALUES ('El título', 'Esto es el cuerpo del artículo.', NOW());
INSERT INTO articles (title,body,created)
    VALUES ('Un título de nuevo', 'Y el cuerpo sigue.', NOW());
INSERT INTO articles (title,body,created)
    VALUES ('El título ataca de nuevo', '¡Esto es realmente emocionante! No.
↪', NOW());
```

La elección de los nombres para el nombre de la tabla y de algunas columnas no se ha hecho al azar. Si sigues las convenciones para nombres en la Base de Datos, y las demás convenciones en tus clases (ver más sobre convenciones aquí: *CakePHP Conventions*), aprovecharás la potencia del framework y ahorrarás mucho trabajo de configuración. CakePHP es suficientemente flexible como para acomodarse hasta en el peor esquema de base de datos, pero utilizando las convenciones ahorrarás tiempo.

Echa un vistazo a *las convencionnes* para más información, pero basta decir que nombrando nuestra tabla 'articles' automáticamente lo vincula a nuestro modelo Articles y que campos llamados *modified* y *created* serán gestionados automáticamente por CakePHP.

Al llamar 'articles' a nuestra tabla de artículos, estamos diciendo a CakePHP que vincule esta tabla por defecto al Modelo 'Articles', e incluiya los campos 'modified' y 'created' con ese nombre, los cuáles serán automáticamente administrados por CakePHP.

Configurando la Base de Datos

Rápido y sencillo, vamos a decirle a CakePHP dónde está la Base de Datos y cómo conectarnos a ella. Seguramente esta sea la primera y última vez que configuras nada.

Una copia del fichero de configuración de CakePHP puede ser hallado en **config/app.default.php**. Copia este fichero en su mismo directorio, pero nómbralo **app.php**.

El fichero de configuración debería de ser bastante sencillo: simplemente reemplaza los valores en la matriz '' Datasources.default'' con los que encajen con tu configuración. Una configuración completa de ejemplo podría parecerse a esto:

```
return [
    // Más configuración arriba
    'Datasources' => [
        'default' => [
            'className' => 'Cake\Database\Connection',
```

```
            'driver' => 'Cake\Database\Driver\Mysql',
            'persistent' => false,
            'host' => 'localhost',
            'username' => 'cake_blog',
            'password' => 'AngelF00dC4k3~',
            'database' => 'cake_blog',
            'encoding' => 'utf8',
            'timezone' => 'UTC'
        ],
    ],
    // Más configuración abajo
];
```

En cuanto guardes tu nuevo fichero **app.php** deberías de ser capaz de acceder mediante tu navegador web y ver la página de bienvenida de CakePHP. También debería decirte que se ha encontrado el fichero de configuración así como que ha podido conectarse a la base de datos.

Nota: Recuerda que debes tener PDO y pdo_mysql habilitados en tu php.ini.

Configuración Opcional

Aún hay unas pocas cosas que puedes configurar. La mayoría de desarrolladores acaban estos ítems de la lista de la compra, pero no se necesitan para este tutorial. Uno de ellos es definir un string de seguridad (security salt) para realizar los 'hash' de seguridad.

El string de seguridad se utiliza para generar 'hashes'. Cambia el valor por defecto editando el fichero **config/app.php**. No importa mucho el valor que contenga, cuanto más largo más difícil de averiguar:

```
'Security' => [
    'salt' => 'Algo largo y conteniendo un montón de distintos valores.',
],
```

Sobre mod_rewrite

Si eres nuevo usuario de apache, puedes encontrar alguna dificultad con mod_rewrite, así que lo trataremos aquí.

Si al cargar la página de bienvenida de CakePHP ves cosas raras (no se cargan las imágenes ni los estilos y se ve todo en blanco y negro), esto significa que probablemente mod_rewrite no está funcionando en tu sistema. Por favor, consulta la sección para tu servidor entre las siguientes acerca de re-escritura de URLs para poder poner en marcha la aplicación:

1. Comprueba que existen los ficheros .htaccess en el directorio en el que está instalada tu aplicación web. A veces al descomprimir el archivo o al copiarlo desde otra ubicación, estos ficheros no se copian correctamente. Si no están ahí, obtén otra copia de CakePHP desde el servidor oficial de descargas.
2. Asegúrate de tener activado el módulo mod_rewrite en la configuración de apache. Deberías tener algo así:

```
    LoadModule rewrite_module       libexec/httpd/mod_rewrite.so

(para apache 1.3)::

    AddModule       mod_rewrite.c

en tu fichero httpd.conf
```

Si no puedes (o no quieres) configurar mod_rewrite o algún otro módulo compatible, necesitarás activar las url amigables en CakePHP. En el fichero **config/app.php**, quita el comentario a la línea:

```
'App' => [
    // ...
    // 'baseUrl' => env('SCRIPT_NAME'),
]
```

Borra también los ficheros .htaccess que ya no serán necesarios:

```
/.htaccess
/webroot/.htaccess
```

Esto hará que tus url sean así: www.example.com/index.php/nombredelcontrolador/nombredelaaccion/parametro en vez de www.example.com/nombredelcontrolador/nombredelaaccion/parametro.

Si estás instalando CakePHP en otro servidor diferente a Apache, encontrarás instrucciones para que funcione la reescritura de URLs en la sección url-rewriting

Ahora continúa hacia *Tutorial de desarrollo del Blog - Añadiendo una capa* para empezar a construir tu primera aplicación en CakePHP.

Tutorial de desarrollo del Blog - Añadiendo una capa

Nota: The documentation is currently partially supported in es language for this page.

Por favor, siéntase libre de enviarnos un pull request en Github[13] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Crear un modelo Artículo (`Article`)

Los modelos son una parte fundamental en CakePHP. Cuando creamos un modelo, podemos interactuar con la base de datos para crear, editar, ver y borrar con facilidad cada ítem de ese modelo.

[13] https://github.com/cakephp/docs

Los modelos están separados entre los objetos `Tabla` (`Table`) y `Entidad` (`Entity`). Los objetos `Tabla` proporcionan acceso a la coleción de entidades almacenada en una tabla específica y va en **src/Model/Table**. El fichero que crearemos se guardará en **src/Model/Table/ArticlesTable.php**. El fichero completo debería tener este aspecto:

```
namespace App\Model\Table;

use Cake\ORM\Table;

class ArticlesTable extends Table
{
    public function initialize(array $config)
    {
        $this->addBehavior('Timestamp');
    }
}
```

Los convenios usados para los nombres son importantes. Llamando a nuestro objeto Tabla `ArticlesTable`, CakePHP deducirá automáticamente que esta Tabla será utilizada en el controlador ArticlesController, y que se vinculará a una tabla en nuestra base de datos llamada `articles`.

Nota: CakePHP creará dinámicamente un objeto para el modelo si no encuentra el fichero correspondiente en **src/Model/Table**. Esto significa que si te equivocas al nombrar el fichero (por ejemplo lo llamas articles-table.php —en minúscula— o ArticleTable.php —en singular) CakePHP no va a reconocer la configuración que escribas en ese fichero y utilizará valores por defecto.

Para más información sobre modelos, como callbacks y validaciones echa un vistazo al capítulo del Manual *Acceso a la base de datos & ORM*.

Crear el Controlador de Artículos (`Articles Controller`)

Vamos a crear ahora un controlador para nuestros artículos. En el controlador es donde escribiremos el código para interactuar con nuestros artículos. Es donde se utilizan los modelos para llevar a cabo el trabajo que queramos hacer con nuestros artículos. Vamos a crear un nuevo fichero llamado **ArticlesController.php** dentro del directorio **src/Controller**. A continuación puedes ver el aspecto básico que debería tener este controlador:

```
namespace App\Controller;

class ArticlesController extends AppController
{
}
```

Vamos a añadir una acción a nuestro nuevo controlador. Las acciones representan una función concreta o interfaz en nuestra aplicación. Por ejemplo, cuando los usuarios recuperan la url www.example.com/articles/index (que es lo mismo que www.example.com/articles/) esperan ver un listado de artículos. El código para tal acción sería este:

```
namespace App\Controller;

class ArticlesController extends AppController
{

    public function index()
    {
        $articles = $this->Articles->find('all');
        $this->set(compact('articles'));
    }
}
```

Por el hecho de haber definido el método `index()` en nuestro ArticlesController, los usuarios ahora pueden acceder a su lógica solicitando www.example.com/articles/index. Del mismo modo, si definimos un método llamado `foobar()` los usuarios tendrán acceso a él desde www.example.com/articles/foobar.

Advertencia: Puede que tengas la tentación de llamar tus controladores y acciones de cierto modo para obtener una URL en concreto. Resiste la tentación. Sigue las convenciones de CakePHP (mayúsculas, nombre en plural, etc.) y crea acciones comprensibles, que se dejen leer. Luego podrás asignar URLs a tu código utilizando "rutas", que veremos más adelante.

La única instrucción en la acción utiliza `set()` para pasar datos desde el controlador hacia la vista (que crearemos a continuación). La línea en cuestión asigna una variable en la vista llamada 'articles' igual al valor retornado por el método `find('all')` del objeto de tabla Artículos (`ArticlesTable`).

Para aprender más sobre los controladores, puedes visitar el capítulo *Controllers*.

Crear Vistas de Artículos (`Article Views`)

Ahora que tenemos nuestros datos fluyendo por el modelo, y que la lógica de nuestra aplicación está definida en nuestro controlador, vamos a crear una vista para la acción índex creada en el paso anterior.

Las vistas en CakePHP únicamente son fragmentos de presentación que encajan dentro de la plantilla (`layout`) de nuestra aplicación. Para la mayoría de aplicaciones son HTML mezclados con PHP, pero bien podrían acabar siendo XML, CSV o incluso datos binarios.

Una plantilla es una presentación de código que envuelve una vista. Se pueden definir múltiples plantillas y puedes cambiar entre ellas pero, por ahora, utilizaremos la plantilla por defecto (`default`).

¿Recuerdas cómo en la sección anterior hemos asignado la variable 'articles' a la vista utilizando el método `set()`? Esto asignaría el objeto de consulta (`query object`) a la vista para ser invocado por una iteración `foreach`.

Las vistas en CakePHP se almacenan en la ruta `/src/Template` y en un directorio con el mismo nombre que el controlador al que pertenecen (tendremos que crear una carpeta llamada 'Articles' en este caso). Para dar formato a los datos de este artículo en una bonita tabla, el código de nuestra vista debería ser algo así:

```
<!-- File: /src/Template/Articles/index.ctp -->
```

```
<h1>Artículos</h1>
<table>
    <tr>
        <th>Id</th>
        <th>Title</th>
        <th>Created</th>
    </tr>

    <!-- Aquí es donde iteramos nuestro objeto de consulta $articles,␣
↪mostrando en pantalla la información del artículo -->

    <?php foreach ($articles as $article): ?>
    <tr>
        <td><?= $article->id ?></td>
        <td>
            <?= $this->Html->link($article->title,
            ['controller' => 'Articles', 'action' => 'view', $article->id]) ?>
        </td>
        <td><?= $article->created->format(DATE_RFC850) ?></td>
    </tr>
    <?php endforeach; ?>
</table>
```

Esto debería ser sencillo de comprender.

Como habrás notado, hay una llamada a un objeto `$this->Html`. Este objeto es una instancia de la clase *`Cake\View\Helper\HtmlHelper`* de CakePHP. CakePHP proporciona un conjunto de ayudantes de vistas (`helpers`) para ayudarte a completar acciones habituales, como por ejemplo crear un enlace o un formulario. Puedes aprender más sobre esto en *Helpers*, pero lo que es importante destacar aquí es que el método `link()` generará un enlace HTML con el título como primer parámetro y la URL como segundo parámetro.

Cuando crees URLs en CakePHP te recomendamos emplear el formato de array. Se explica con detenimiento en la sección de Rutas (`Routes`). Si utilizas las rutas en formato array podrás aprovecharte de las potentes funcionalidades de generación de rutas inversa de CakePHP en el futuro. Además puedes especificar rutas relativas a la base de tu aplicación de la forma `/controlador/accion/param1/param2` o incluso utilizar *Using Named Routes*.

Llegados a este punto, deberías ser capaz de acceder con tu navegador a http://www.example.com/articles/index. Deberías ver tu vista, correctamente formatada con el título y la tabla listando los artículos.

Si te ha dado por hacer clic en uno de los enlaces que hemos creado en esta vista (que enlazan el título de un artículo hacia la URL `/articles/view/un\_id`), seguramente habrás sido informado por CakePHP de que la acción no ha sido definida todavía. Si no has sido infromado, o bien algo ha ido mal o bien ya la habías definido, en cuyo caso eres muy astuto. En caso contrario, la crearemos ahora en nuestro controlador de artículos:

```
namespace App\Controller;

class ArticlesController extends AppController
{
```

```
    public function index()
    {
         $this->set('articles', $this->Articles->find('all'));
    }

    public function view($id = null)
    {
        $article = $this->Articles->get($id);
        $this->set(compact('article'));
    }
}
```

Si observas la función view(), ahora el método set() debería serte familiar. Verás que estamos usando `get()` en vez de `find('all')` ya que sólo queremos un artículo concreto.

Verás que nuestra función view toma un parámetro: el ID del artículo que queremos ver. Este parámetro se gestiona automáticamente al llamar a la URL `/articles/view/3`, el valor '3' se pasa a la función view como primer parámetro `$id`.

También hacemos un poco de verificación de errores para asegurarnos de que el usuario realmente accede a dicho registro. Si el usuario solicita `/articles/view` lanzaremos una excepción `NotFoundException` y dejaremos al ErrorHandler tomar el control. Utilizando el método `get()` en la tabla Articles también hacemos una verificación similar para asegurarnos de que el usuario ha accedido a un registro que existe. En caso de que el artículo solicitado no esté presente en la base de datos, el método `get()` lanzará una excepción `NotFoundException`.

Ahora vamos a definir la vista para esta nueva función 'view' ubicándola en **src/Template/Articles/view.ctp**.

```
<!-- File: /src/Template/Articles/view.ctp -->
<h1><?= h($article->title) ?></h1>
<p><?= h($article->body) ?></p>
<p><small>Created: <?= $article->created->format(DATE_RFC850) ?></small></p>
```

Verifica que esto funciona probando los enlaces en `/articles/index` o puedes solicitándolo manualmente accediendo a `/articles/view/1`.

Añadiendo Artículos

Leer de la base de datos y mostrar nuestros artículos es un gran comienzo, pero permitamos también añadir nuevos artículos.

Lo primero, añadir una nueva acción `add()` en nuestro controlador ArticlesController:

```
namespace App\Controller;

class ArticlesController extends AppController
{
    public $components = ['Flash'];

    public function index()
```

```
    {
        $this->set('articles', $this->Articles->find('all'));
    }

    public function view($id)
    {
        $article = $this->Articles->get($id);
        $this->set(compact('article'));
    }

    public function add()
    {
        $article = $this->Articles->newEntity();
        if ($this->request->is('post')) {
            $article = $this->Articles->patchEntity($article, $this->request->
↪data);
            if ($this->Articles->save($article)) {
                $this->Flash->success(__('Your article has been saved.'));
                return $this->redirect(['action' => 'index']);
            }
            $this->Flash->error(__('Unable to add your article.'));
        }
        $this->set('article', $article);
    }
}
```

Nota: Necesitas incluir el FlashComponent en cualquier controlador donde vayas a usarlo. Si lo ves necesario, inclúyelo en tu AppController.

Lo que la función add() hace es: si el formulario enviado no está vacío, intenta salvar un nuevo artículo utilizando el modelo Articles. Si no se guarda bien, muestra la vista correspondiente, así podremos mostrar los errores de validación u otras alertas.

Cada petición de CakePHP incluye un objeto `Request` que es accesible utilizando `$this->request`. El objeto de petición contiene información útil acerca de la petición que se recibe y puede ser utilizado para controlar el flujo de nuestra aplicación. En este caso, utilizamos el método `Cake\Network\Request::is()` para verificar que la petición es una petición HTTP POST.

Cuando un usuario utiliza un formulario y efectúa un POST a la aplicación, esta información está disponible en `$this->request->data`. Puedes usar la función `pr()` o `debug()` para mostrar el contenido de esa variable y ver la pinta que tiene.

Utilizamos el método mágico `__call` del `FlashComponent` para guardar un mensaje en una variable de sesión que será mostrado en la página después de la redirección. En la plantilla tenemos `<?= $this->Flash->render() ?>` que muestra el mensaje y elimina la correspondiente variable de sesión. El método `Cake\Controller\Controller::redirect` del controlador redirige hacia otra URL. El parámetro `['action' => 'index']` se traduce a la URL /articles (p.e. la acción index del controlador de artículos). Puedes echar un ojo al método `Cake\Routing\Router::url()` en la API[14]

[14] https://api.cakephp.org

para ver los formatos en que puedes especificar una URL para varias funciones de CakePHP.

Al llamar al método `save()`, comprobará si hay errores de validación primero y si encuentra alguno, no continuará con el proceso de guardado. Veremos a continuación cómo trabajar con estos errores de validación.

Validando los Datos

CakePHP te ayuda a evitar la monotonía al construir tus formularios y su validación. Todos odiamos teclear largos formularios y gastar más tiempo en reglas de validación de cada campo. CakePHP lo hace más rápido y sencillo.

Para aprovechar estas funciones es conveniente que utilices el FormHelper en tus vistas. La clase *Cake\View\Helper\FormHelper* está disponible en tus vistas por defecto a través de `$this->Form`.

He aquí nuestra vista `add`:

```
<!-- File: src/Template/Articles/add.ctp -->

<h1>Añadir Artículo</h1>
<?php
    echo $this->Form->create($article);
    echo $this->Form->input('title');
    echo $this->Form->input('body', ['rows' => '3']);
    echo $this->Form->button(__('Guardar artículo'));
    echo $this->Form->end();
?>
```

Hemos usado FormHelper para generar la etiqueta 'form'. La ejecución de `$this->Form->create()` genera el siguiente código:

```
<form method="post" action="/articles/add">
```

Si `create()` no tiene parámetros al ser llamado, asume que estás creando un formulario que envía vía POST a la acción `add()` (o `edit()` cuando `id` es incluido en los datos de formulario) del controlador actual.

El método `$this->Form->input()` se utiliza para crear elementos de formulario del mismo nombre. El primer parámetro le indica a CakePHP a qué campo corresponde y el segundo parámetro te permite especificar un abanico muy ámplio de opciones - en este caso, el número de filas del textarea que se generará. Hay un poco de introspección y "automagia" aquí: `input()` generará distintos elementos de formulario en función del campo del modelo especificado.

La llamada a `$this->Form->end()` cierra el formulario. También generará campos ocultos si la CSRF/prevención de manipulación de formularios ha sido habilitada.

Volvamos atrás un minuto y actualicemos nuestra vista **src/Template/Articles/index.ctp** para añadir un enlace de "Añadir Artículo". Justo antes del tag <table> añade la siguiente línea:

```
<?= $this->Html->link(
    'Añadir artículo',
```

```
    ['controller' => 'Articles', 'action' => 'add']
) ?>
```

Te estarás preguntando: ¿Cómo le digo a CakePHP la forma en la que debe validar estos datos? Muy sencillo, las reglas de validación se escriben en el modelo. Volvamos al modelo `Articles` y hagamos algunos ajustes:

```
namespace App\Model\Table;

use Cake\ORM\Table;
use Cake\Validation\Validator;

class ArticlesTable extends Table
{
    public function initialize(array $config)
    {
        $this->addBehavior('Timestamp');
    }

    public function validationDefault(Validator $validator)
    {
        $validator
            ->notEmpty('title')
            ->notEmpty('body');

        return $validator;
    }
}
```

El método `validationDefault()` le dice a CakePHP cómo validar tus datos cuando se invoca el método `save()`. Aquí hemos especificado que ambos campos, el cuerpo y el título, no pueden quedar vacíos. El motor de validaciones de CakePHP es potente y con numerosas reglas ya predefinidas (tarjetas de crédito, direcciones de e-mail, etc.) así como flexibilidad para añadir tus propias reglas de validación. Para más información en tal configuración, echa un vistazo a la documentación *Validation*.

Ahora que ya tienes las reglas de validación definidas, usa tu aplicación para crear un nuevo artículo con un título vacío y verás cómo funcionan. Como hemos usado el método `Cake\View\Helper\FormHelper::input()`, los mensajes de error se construyen automáticamente en la vista sin código adicional.

Editando Artículos

Editando artículos: allá vamos. Ya eres un profesional de CakePHP, así que habrás cogido la pauta. Crear una acción, luego la vista. He aquí cómo debería ser la acción `edit()` del controlador `ArticlesController`:

```
public function edit($id = null)
{
    $article = $this->Articles->get($id);
    if ($this->request->is(['post', 'put'])) {
```

```
        $this->Articles->patchEntity($article, $this->request->data);
        if ($this->Articles->save($article)) {
            $this->Flash->success(__('Tu artículo ha sido actualizado.'));
            return $this->redirect(['action' => 'index']);
        }
        $this->Flash->error(__('Tu artículo no se ha podido actualizar.'));
    }

    $this->set('article', $article);
}
```

Lo primero que hace este método es asegurarse de que el usuario ha intentado acceder a un registro existente. Si no han pasado el parámetro `$id` o el artículo no existe lanzaremos una excepción `NotFoundException` para que el `ErrorHandler` se ocupe de ello.

Luego verifica si la petición es POST o PUT. Si lo es, entonces utilizamos los datos recibidos para actualizar nuestra entidad artículo (`article`) utilizando el método 'patchEntity'. Finalmente utilizamos el objeto tabla para guardar la entidad de nuevo o mostrar errores de validación al usuario en caso de haberlos.

La vista sería algo así:

```
<!-- File: src/Template/Articles/edit.ctp -->

<h1>Edit Article</h1>
<?php
    echo $this->Form->create($article);
    echo $this->Form->input('title');
    echo $this->Form->input('body', ['rows' => '3']);
    echo $this->Form->button(__('Guardar artículo'));
    echo $this->Form->end();
?>
```

Mostramos el formulario de edición (con los valores actuales de ese artículo), junto a los errores de validación que hubiese.

CakePHP utilizará el resultado de `$article->isNew()` para determinar si un `save()` debería insertar un nuevo registro o actualizar uno existente.

Puedes actualizar tu vista índice (`index`) con enlaces para editar artículos específicos:

```
<!-- File: src/Template/Articles/index.ctp  (edit links added) -->

<h1>Artículos</h1>
<p><?= $this->Html->link("Añadir artículo", ['action' => 'add']) ?></p>
<table>
    <tr>
        <th>Id</th>
        <th>Title</th>
        <th>Created</th>
        <th>Action</th>
    </tr>

<!-- Aquí es donde iteramos nuestro objeto de consulta $articles, mostrando␣
↪en pantalla la información del artículo -->
```

```
<?php foreach ($articles as $article): ?>
    <tr>
        <td><?= $article->id ?></td>
        <td>
            <?= $this->Html->link($article->title, ['action' => 'view',
↪$article->id]) ?>
        </td>
        <td>
            <?= $article->created->format(DATE_RFC850) ?>
        </td>
        <td>
            <?= $this->Html->link('Editar', ['action' => 'edit', $article->
↪id]) ?>
        </td>
    </tr>
<?php endforeach; ?>

</table>
```

Borrando Artículos

Vamos a permitir a los usuarios que borren artículos. Empieza con una acción `delete()` en el controlador `ArticlesController`:

```
public function delete($id)
{
    $this->request->allowMethod(['post', 'delete']);

    $article = $this->Articles->get($id);
    if ($this->Articles->delete($article)) {
        $this->Flash->success(__('El artículo con id: {0} ha sido eliminado.
↪', h($id)));
        return $this->redirect(['action' => 'index']);
    }
}
```

La lógica elimina el artículo especificado por $id y utiliza `$this->Flash->success()` para mostrar al usuario un mensaje de confirmación tras haber sido redirigidos a `/articles`. Si el usuario intenta eliminar utilizando una petición GET, el 'allowMethod' devolvería una Excepción. Las excepciones que no se traten serán capturadas por el manejador de excepciones de CakePHP (`exception handler`) y una bonita página de error es mostrada. Hay muchas *Excepciones* que pueden ser utilizadas para indicar los varios errores HTTP que tu aplicación pueda generar.

Como estamos ejecutando algunos métodos y luego redirigiendo a otra acción de nuestro controlador, no es necesaria ninguna vista (nunca se usa). Lo que si querrás es actualizar la vista index.ctp para incluír el ya habitual enlace:

```
<!-- File: src/Template/Articles/index.ctp -->
```

```
<h1>Artículos</h1>
<p><?= $this->Html->link("Añadir artículo", ['action' => 'add']) ?></p>
<table>
    <tr>
        <th>Id</th>
        <th>Title</th>
        <th>Created</th>
        <th>Action</th>
    </tr>

<!-- Aquí es donde iteramos nuestro objeto de consulta $articles, mostrando␣
↪en pantalla la información del artículo -->

<?php foreach ($articles as $article): ?>
    <tr>
        <td><?= $article->id ?></td>
        <td>
            <?= $this->Html->link($article->title, ['action' => 'view',
↪$article->id]) ?>
        </td>
        <td>
            <?= $article->created->format(DATE_RFC850) ?>
        </td>
        <td>
            <?= $this->Form->postLink(
                'Eliminar',
                ['action' => 'delete', $article->id],
                ['confirm' => '¿Estás seguro?'])
            ?>
            <?= $this->Html->link('Editar', ['action' => 'edit', $article->
↪id]) ?>
        </td>
    </tr>
<?php endforeach; ?>

</table>
```

Utilizando `View\Helper\FormHelper::postLink()` crearemos un enlace que utilizará JavaScript para hacer una petición POST que eliminará nuestro artículo. Permitiendo que contenido sea eliminado vía peticiones GET es peligroso, ya que arañas web (`crawlers`) podrían eliminar accidentalmente tu contenido.

Nota: Esta vista utiliza el FormHelper para pedir confirmación vía diálogo de confirmación de JavaScript al usuario antes de borrar un artículo.

Rutas (`Routes`)

En muchas ocasiones, las rutas por defecto de CakePHP funcionan bien tal y como están. Los desarroladores que quieren rutas diferentes para mejorar la usabilidad apreciarán la forma en la que CakePHP relaciona las

URLs con las acciones de los controladores. Vamos a hacer cambios ligeros para este tutorial.

Para más información sobre las rutas así como técnicas avanzadas revisa *Connecting Routes*.

Por defecto CakePHP responde a las llamadas a la raíz de tu sitio (por ejemplo http://www.example.com) usando el controlador PagesController, mostrando una vista llamada "home". En lugar de eso, lo reemplazaremos con nuestro controlador `ArticlesController` creando una nueva ruta.

Las reglas de enrutamiento están en **config/routes.php**. Querrás eliminar o comentar la línea que define la raíz por defecto. Dicha ruta se parece a esto:

```
Router::connect('/', ['controller' => 'Pages', 'action' => 'display', 'home
↪']);
```

Esta línea conecta la url '/' con la página por defecto de inicio de CakePHP. Queremos conectarla a nuestro propio controlador, así que reemplaza dicha línea por esta otra:

```
Router::connect('/', ['controller' => 'Articles', 'action' => 'index']);
```

Esto debería, cuando un usuario solicita '/', devolver la acción index() del controlador ArticlesController.

Nota: CakePHP también calcula las rutas a la inversa. Si en tu código pasas el array `['controller' => 'Articles','action' => 'index']` a una función que espera una url, el resultado será '/'. Es buena idea usar siempre arrays para configurar las URL, lo que asegura que los links irán siempre al mismo lugar.

Conclusión

Creando aplicaciones de este modo te traerá paz, honor, amor, dinero a carretas e incluso tus fantasías más salvajes. Simple, no te parece? Ten en cuenta que este tutorial es muy básico, CakePHP tiene *muchas* otras cosas que ofrecer y es flexible aunque no hemos cubierto aquí estos puntos para que te sea más simple al principio. Usa el resto de este manual como una guía para construir mejores aplicaciones.

Ahora que ya has creado una aplicación CakePHP básica, estás listo para la vida real. Empieza tu nuevo proyecto y lee el resto del Cookbook así como la API[15].

Si necesitas ayuda, hay muchos modos de encontrar la ayuda que buscas - por favor, míralo en la página *Where to Get Help*. ¡Bienvenido a CakePHP!

Lectura sugerida para continuar desde aquí

Hay varias tareas comunes que la gente que está aprendiendo CakePHP quiere aprender después:

1. *More About Views*: Personaliza la plantilla *layout* de tu aplicación
2. *More About Views* Incluír vistas y reutilizar trozos de código
3. *Code Generation with Bake*: Generación básica de CRUDs

[15] https://api.cakephp.org

4. *Tutorial de aplicación Blog - Autenticación y Autorización*: Tutorial de autenticación y permisos

Tutorial de desarrollo de Blog - Parte 3

Crear categorias en Arbol

Vamos a continuar con nuestro blog e imaginar que queremos categorizar nuestros articulos. Queremos que las categorias estén ordenadas, y para esto, vamos a usar *Tree behavior* para ayudarnos a organizar las categorías.

Pero primero necesitamos modificar nuestras tablas.

Plugin de migración

Vamos a usar el migrations plugin[16] para crear una tabla en nuestra base de datos. Si tienes una tabla de articulos en tu base de datos, borrala.

Abre tu archivo `composer.json`. Generalmente el plugin de migración ya esta incluido en `require`. Si no es el caso, agrégalo:

```
"require": {
    "cakephp/migrations": "~1.0"
}
```

Luego corre el comando `composer update`. El plugin de migración se alojara en tu carpeta de `plugins`. Agrega también `Plugin::load('Migrations');` en el archivo bootstrap.php de tú aplicación.

Una vez que el plugin sea cargado, corre el siguiente comando para crear el archivo de migración:

```
bin/cake migrations create Initial
```

Un archivo de migración será creado en la carpeta `/config/Migrations`. Puedes abrir tu archivo y agregar las siguientes lineas:

```
<?php

use Phinx\Migration\AbstractMigration;

class Initial extends AbstractMigration
{
    public function change()
    {
        $articles = $this->table('articles');
        $articles->addColumn('title', 'string', ['limit' => 50])
            ->addColumn('body', 'text', ['null' => true, 'default' => null])
            ->addColumn('category_id', 'integer', ['null' => true, 'default'␣
↪=> null])
```

[16] https://github.com/cakephp/migrations

```
            ->addColumn('created', 'datetime')
            ->addColumn('modified', 'datetime', ['null' => true, 'default' =>␣
↪null])
            ->save();

        $categories = $this->table('categories');
        $categories->addColumn('parent_id', 'integer', ['null' => true,
↪'default' => null])
            ->addColumn('lft', 'integer', ['null' => true, 'default' => null])
            ->addColumn('rght', 'integer', ['null' => true, 'default' =>␣
↪null])
            ->addColumn('name', 'string', ['limit' => 255])
            ->addColumn('description', 'string', ['limit' => 255, 'null' =>␣
↪true, 'default' => null])
            ->addColumn('created', 'datetime')
            ->addColumn('modified', 'datetime', ['null' => true, 'default' =>␣
↪null])
            ->save();
    }
}
```

Ahora corre el siguiente comando para crear tús tablas:

```
bin/cake migrations migrate
```

Modificando las tablas

Con nuestras tablas creadas, ahora podemos enfocarnos en categorizar los artículos.

Suponemos que ya tienes los archivos (Tables, Controllers y Templates de Articles) de la parte 2 de esta serie de tutoriales, por lo que solamente vamos a agregar referencia a las categorías.

Necesitamos asociar las tablas de Articles y Categories. Abre el archivo **src/Model/Table/ArticlesTable.php** y agrega las siguientes lineas:

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

namespace App\Model\Table;

use Cake\ORM\Table;

class ArticlesTable extends Table
{
    public function initialize(array $config)
    {
        $this->addBehavior('Timestamp');
        // Just add the belongsTo relation with CategoriesTable
        $this->belongsTo('Categories', [
            'foreignKey' => 'category_id',
        ]);
    }
}
```

Generando el código base para las Categorías

Crea todos los archivos corriendo los siguientes comandos:

```
bin/cake bake model Categories
bin/cake bake controller Categories
bin/cake bake template Categories
```

La herramienta bake ha creado todos los archivos en un instante. Puedes darles una rápida leida si necesitas re-familiarizarte con la forma en la que CakePHP funciona.

Nota: Si estás en Windows recordá usar en lugar de / .

Agregar el TreeBehavior a CategoriesTable

TreeBehavior ayuda a manejar estructuras de árbol jerarquica en una tabla. Utiliza MPTT logic[17] para manejar los datos. Las estructuras en árbol MPTT están optimizadas para lecturas, lo cual las hace ideal para aplicaciones con gran carga de lectura como los blogs.

Si abres el archivo **src/Model/Table/CategoriesTable.php** veras que el TreeBehavior fue agregado a CategoriesTable en el método `initialize()`. Bake agrega este behavior a cualquier tabla que contenga las columnas `lft` y `rght`:

```
$this->addBehavior('Tree');
```

Con el TreeBehavior agregado ahora podras acceder a algunas funcionalidades como reordenar las categorias. Veremos eso en un momento.

Pero por ahora tendrás que removar los siguientes inputs en tus archivos add y edit de Categories:

```
echo $this->Form->input('lft');
echo $this->Form->input('rght');
```

Esos campos son manejados automáticamento por el TreeBehavior cuando una categoría es guardada.

Con tú navegador, agrega alguna nueva categoría usando la acción `/yoursite/categories/add`.

Reordenando categorías con TreeBehavior

En el index de categorias, puedes listar y re-ordenar categorias.

Vamos a modificar el método index en tu `CategoriesController.php`, agregando `move_up()` y `move_down()` para poder reordenar las categorías en ese árbol:

[17] http://www.sitepoint.com/hierarchical-data-database-2/

```
class CategoriesController extends AppController
{
    public function index()
    {
        $categories = $this->Categories->find('threaded')
            ->order(['lft' => 'ASC']);
        $this->set(compact('categories'));
    }

    public function move_up($id = null)
    {
        $this->request->allowMethod(['post', 'put']);
        $category = $this->Categories->get($id);
        if ($this->Categories->moveUp($category)) {
            $this->Flash->success('The category has been moved Up.');
        } else {
            $this->Flash->error('The category could not be moved up. Please,␣
↪try again.');
        }
        return $this->redirect($this->referer(['action' => 'index']));
    }

    public function move_down($id = null)
    {
        $this->request->allowMethod(['post', 'put']);
        $category = $this->Categories->get($id);
        if ($this->Categories->moveDown($category)) {
            $this->Flash->success('The category has been moved down.');
        } else {
            $this->Flash->error('The category could not be moved down.␣
↪Please, try again.');
        }
        return $this->redirect($this->referer(['action' => 'index']));
    }
}
```

En **src/Template/Categories/index.ctp** reemplazá el contenido existente por el siguiente:

```
<div class="actions columns large-2 medium-3">
    <h3><?= __('Actions') ?></h3>
    <ul class="side-nav">
        <li><?= $this->Html->link(__('New Category'), ['action' => 'add']) ?>
↪</li>
    </ul>
</div>
<div class="categories index large-10 medium-9 columns">
    <table cellpadding="0" cellspacing="0">
    <thead>
        <tr>
            <th>id</th>
            <th>Parent Id</th>
            <th>Title</th>
            <th>Lft</th>
```

```
            <th>Rght</th>
            <th>Name</th>
            <th>Description</th>
            <th>Created</th>
            <th class="actions"><?= __('Actions') ?></th>
        </tr>
    </thead>
    <tbody>
    <?php foreach ($categories as $category): ?>
        <tr>
            <td><?= $this->Number->format($category->id) ?></td>
            <td><?= $this->Number->format($category->parent_id) ?></td>
            <td><?= $this->Number->format($category->lft) ?></td>
            <td><?= $this->Number->format($category->rght) ?></td>
            <td><?= h($category->name) ?></td>
            <td><?= h($category->description) ?></td>
            <td><?= h($category->created) ?></td>
            <td class="actions">
                <?= $this->Html->link(__('View'), ['action' => 'view',
↪$category->id]) ?>
                <?= $this->Html->link(__('Edit'), ['action' => 'edit',
↪$category->id]) ?>
                <?= $this->Form->postLink(__('Delete'), ['action' => 'delete
↪', $category->id], ['confirm' => __('Are you sure you want to delete # {0}?
↪', $category->id)]) ?>
                <?= $this->Form->postLink(__('Move down'), ['action' => 'move_
↪down', $category->id], ['confirm' => __('Are you sure you want to move down
↪# {0}?', $category->id)]) ?>
                <?= $this->Form->postLink(__('Move up'), ['action' => 'move_up
↪', $category->id], ['confirm' => __('Are you sure you want to move up # {0}?
↪', $category->id)]) ?>
            </td>
        </tr>
    <?php endforeach; ?>
    </tbody>
    </table>
</div>
```

Modificando el ArticlesController

En tú `ArticlesController`, vamos a obtener el listado de categorías. Esto nos permitirá elegir una categoría para un Article al momento de crearlo o editarlo:

```
// src/Controller/ArticlesController.php

namespace App\Controller;

use Cake\Network\Exception\NotFoundException;

class ArticlesController extends AppController
{
```

```
    // ...

    public function add()
    {
        $article = $this->Articles->newEntity();
        if ($this->request->is('post')) {
            $article = $this->Articles->patchEntity($article, $this->request->
↪data);
            if ($this->Articles->save($article)) {
                $this->Flash->success(__('Your article has been saved.'));
                return $this->redirect(['action' => 'index']);
            }
            $this->Flash->error(__('Unable to add your article.'));
        }
        $this->set('article', $article);

        // Just added the categories list to be able to choose
        // one category for an article
        $categories = $this->Articles->Categories->find('treeList');
        $this->set(compact('categories'));
    }
}
```

Modificando el template de Articles

El template add de Article debería verse similar a esto:

```
.. code-block:: php
```

<!– File: src/Template/Articles/add.ctp –>

<h1>Add Article</h1> <?php echo $this->Form->create($article); // just added the categories input echo $this->Form->input('categories'); echo $this->Form->input('title'); echo $this->Form->input('body', ['rows' => '3']); echo $this->Form->button(__('Save Article')); echo $this->Form->end();

Ingresando a */yoursite/articles/add* deberías ver una lista de categorías para elegir.

Tutorial de aplicación Blog - Autenticación y Autorización

Siguiendo con nuestro ejemplo de aplicacion *Tutorial de desarrollo del Blog*, imaginá que necesitamos proteger ciertas URLs, dependiendo del usuario logeado. También tenemos otro requisito, permitir que nuestro blog tenga varios autores, cada uno habilitado para crear sus posts, editar y borrarlos a voluntad, evitando que otros autores puedan cambiarlos.

Creando el codigo para usuarios

Primero, vamos a crear una tabla en nuestra base de datos para guardar los datos de usuarios:

```
CREATE TABLE users (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50),
    password VARCHAR(255),
    role VARCHAR(20),
    created DATETIME DEFAULT NULL,
    modified DATETIME DEFAULT NULL
);
```

Siguimos las convenciones de CakePHP para nombrar tablas pero también estamos aprovechando otra convencion: al usar los campos username y password en nuestra tabla CakePHP configurará automáticamente la mayoria de las cosas al momento de implementar el login.

El siguiente paso es crear Users table, responsable de buscar, guardar y validar los datos de usuario:

```
// src/Model/Table/UsersTable.php
namespace App\Model\Table;

use Cake\ORM\Table;
use Cake\Validation\Validator;

class UsersTable extends Table
{

    public function validationDefault(Validator $validator)
    {
        return $validator
            ->notEmpty('username', 'A username is required')
            ->notEmpty('password', 'A password is required')
            ->notEmpty('role', 'A role is required')
            ->add('role', 'inList', [
                'rule' => ['inList', ['admin', 'author']],
                'message' => 'Please enter a valid role'
            ]);
    }

}
```

También vamos a crear UsersController; el siguiente contenido fue generado usando baked UsersController con el generador de código incluído con CakePHP:

```
// src/Controller/UsersController.php

namespace App\Controller;

use App\Controller\AppController;
use Cake\Event\Event;
use Cake\Network\Exception\NotFoundException;
```

```
class UsersController extends AppController
{

    public function beforeFilter(Event $event)
    {
        parent::beforeFilter($event);
        $this->Auth->allow('add');
    }

     public function index()
     {
        $this->set('users', $this->Users->find('all'));
    }

    public function view($id)
    {
        if (!$id) {
            throw new NotFoundException(__('Invalid user'));
        }

        $user = $this->Users->get($id);
        $this->set(compact('user'));
    }

    public function add()
    {
        $user = $this->Users->newEntity();
        if ($this->request->is('post')) {
            $user = $this->Users->patchEntity($user, $this->request->data);
            if ($this->Users->save($user)) {
                $this->Flash->success(__('The user has been saved.'));
                return $this->redirect(['action' => 'add']);
            }
            $this->Flash->error(__('Unable to add the user.'));
        }
        $this->set('user', $user);
    }

}
```

De la misma forma que creamos las vistas para los posts del blog o usando la herramienta de generación de código, creamos las vistas. Para los objetivos de este tutorial, mostraremos solamente add.ctp:

```
<!-- src/Template/Users/add.ctp -->

<div class="users form">
<?= $this->Form->create($user) ?>
    <fieldset>
        <legend><?= __('Add User') ?></legend>
        <?= $this->Form->input('username') ?>
        <?= $this->Form->input('password') ?>
        <?= $this->Form->input('role', [
            'options' => ['admin' => 'Admin', 'author' => 'Author']
```

```
        ]) ?>
    </fieldset>
<?= $this->Form->button(__('Submit')); ?>
<?= $this->Form->end() ?>
</div>
```

Autenticación (login y logout)

Ya estamos listos para agregar nuestra autenticación. En CakePHP esto es manejado por `Cake\Controller\Component\AuthComponent`, responsable de requerir login para ciertas acciones, de manejar el sign-in y el sign-out y también de autorizar usuarios logeados a ciertas acciones que estan autorizados a utilizar.

Para agregar este componente a tú aplicación abre el archivo **src/Controller/AppController.php** y agrega las siguientes lineas:

```
// src/Controller/AppController.php

namespace App\Controller;

use Cake\Controller\Controller;
use Cake\Event\Event;

class AppController extends Controller
{
    //...

    public function initialize()
    {
        $this->loadComponent('Flash');
        $this->loadComponent('Auth', [
            'loginRedirect' => [
                'controller' => 'Articles',
                'action' => 'index'
            ],
            'logoutRedirect' => [
                'controller' => 'Pages',
                'action' => 'display',
                'home'
            ]
        ]);
    }

    public function beforeFilter(Event $event)
    {
        $this->Auth->allow(['index', 'view', 'display']);
    }
    //...
}
```

No hay mucho que configurar, al haber utilizado convenciones para la tabla de usuarios. Simplemente asig-

namos las URLs que serán cargadas despues del login y del logout, en nuestro caso `/articles/` y `/` respectivamente.

Lo que hicimos en `beforeFilter()` fue decirle al AuthComponent que no requiera login para las acciones index y view en cada controlador. Queremos que nuestros visitantes puedan leer y listar las entradas sin registrarse.

Ahora necesitamos poder registrar nuevos usuarios, guardar el nombre de usuario y contraseña, y hashear su contraseña para que no sea guardada como texto plano. Vamos a decirle al AuthComponent que deje usuarios sin autenticar acceder a la funcion add del controlador users e implementemos las acciones de login y logout:

```
// src/Controller/UsersController.php

public function beforeFilter(Event $event)
{
    parent::beforeFilter($event);
    // Allow users to register and logout.
    // You should not add the "login" action to allow list. Doing so would
    // cause problems with normal functioning of AuthComponent.
    $this->Auth->allow(['add', 'logout']);
}

public function login()
{
    if ($this->request->is('post')) {
        $user = $this->Auth->identify();
        if ($user) {
            $this->Auth->setUser($user);
            return $this->redirect($this->Auth->redirectUrl());
        }
        $this->Flash->error(__('Invalid username or password, try again'));
    }
}

public function logout()
{
    return $this->redirect($this->Auth->logout());
}
```

El hasheo del password aún no está hecho, necesitamos una clase Entity para nuestra clase User para así manejar esta lógica específica. Crea el archivo **src/Model/Entity/User.php** y agrega las siguientes lineas:

```
// src/Model/Entity/User.php
namespace App\Model\Entity;

use Cake\Auth\DefaultPasswordHasher;
use Cake\ORM\Entity;

class User extends Entity
{

    // Make all fields mass assignable for now.
```

```
    protected $_accessible = ['*' => true];

    // ...

    protected function _setPassword($password)
    {
        return (new DefaultPasswordHasher)->hash($password);
    }

    // ...
}
```

Ahora cada vez que la propiedad password sea asignada a un usuario, será hasheada usando la clase `DefaultPasswordHasher`. Solamente nos falta un archivo para la vista de la acción login. Abre tu archivo **src/Template/Users/login.ctp** y agrega las siguientes lineas:

```
<!-- File: src/Template/Users/login.ctp -->

<div class="users form">
<?= $this->Flash->render('auth') ?>
<?= $this->Form->create() ?>
    <fieldset>
        <legend><?= __('Please enter your username and password') ?></legend>
        <?= $this->Form->input('username') ?>
        <?= $this->Form->input('password') ?>
    </fieldset>
<?= $this->Form->button(__('Login')); ?>
<?= $this->Form->end() ?>
</div>
```

Ya podés registrar un nuevo usuario accediendo a `/users/add` e iniciar sesión con las nuevas credenciales ingresando a `/users/login`. También al intentar acceder a alguna otra URL que no fue explicitamente autorizada, por ejemplo `/articles/add`, la aplicación te redireccionará automaticamente al la pagina de login.

Y eso es todo! Se ve demasiado simple para ser verdad. Volvamos un poco para explicar que pasa. La función `beforeFilter()` le dice al AuthComponent que no requiera login para la acción `add()` asi como para `index()` y `view()`, autorizadas en el `beforeFilter()` del AppController.

La función `login()` llama a `$this->Auth->identify()` del AuthComponent, y funciona sin ninguna otra configuración ya que seguimos la convención. Es decir, tener un modelo llamado User con los campos username y password, y usar un formulario que hace post a un controlador con los datos del usuario. Esta función devuelve si el login fue exitoso o no, y en caso de que tenga exito redirige a la URL puesta en AppController, dentro de la configuracion del AuthComponent.

El logout funciona simplemente al acceder a `/users/logout` y redirecciona al usuario a la URL configurada.

Autorización (quién está autorizado a acceder qué)

Como mencionamos antes, estamos convirtiendo este blog en una herramienta de autoría multiusuario, y para hacer esto necesitamos modificar la tabla de posts para agregar referencia al modelo User:

```
ALTER TABLE articles ADD COLUMN user_id INT(11);
```

También, un pequeño cambio en ArticlesController es necesario para guardar el usuario logeado como referencia en los artículos creados:

```
// src/Controller/ArticlesController.php

public function add()
{
    $article = $this->Articles->newEntity();
    if ($this->request->is('post')) {
        $article = $this->Articles->patchEntity($article, $this->request->
↪data);
        // Added this line
        $article->user_id = $this->Auth->user('id');
        // You could also do the following
        //$newData = ['user_id' => $this->Auth->user('id')];
        //$article = $this->Articles->patchEntity($article, $newData);
        if ($this->Articles->save($article)) {
            $this->Flash->success(__('Your article has been saved.'));
            return $this->redirect(['action' => 'index']);
        }
        $this->Flash->error(__('Unable to add your article.'));
    }
    $this->set('article', $article);
}
```

La función `user()` del AuthComponent devuelve datos del usuario actualmente logeado. Usamos este método para agregar datos a la información que será guardada.

Vamos a prevenir que autores puedan editar o eliminar los artículos de otros autores. La regla básica para nuestra aplicación es que los usuarios admin pueden acceder todas las URL, mientras que los usuarios normales (autores) solamente pueden acceder las acciones permitidas. Abre nuevamente AppController y agregá las siguientes opciones en la configuración del Auth:

```
// src/Controller/AppController.php

public function initialize()
{
    $this->loadComponent('Flash');
    $this->loadComponent('Auth', [
        'authorize' => ['Controller'], // Added this line
        'loginRedirect' => [
            'controller' => 'Articles',
            'action' => 'index'
        ],
        'logoutRedirect' => [
            'controller' => 'Pages',
```

```
            'action' => 'display',
            'home'
        ]
    ]);
}

public function isAuthorized($user)
{
    // Admin can access every action
    if (isset($user['role']) && $user['role'] === 'admin') {
        return true;
    }

    // Default deny
    return false;
}
```

Hemos creado un mecanismo de autorización muy simple. En este caso, los usuarios con el rol `admin` podrán acceder a cualquier URL del sitio cuando esten logeados, pero el resto de los usuarios no podrán hacer más que los usuarios no logeados.

Esto no es exactamente lo que queriamos, por lo que tendremos que agregar mas reglas a nuestro método `isAuthorized()`. Pero en lugar de hacerlo en AppController, vamos a delegar a cada controlador. Las reglas que vamos a agregar a ArticlesController deberian permitirle a los autores crear artículos, pero prevenir que editen artículos que no le pertenezcan. Abre el archivo ArticlesController.php y agregá las siguientes lineas:

```
// src/Controller/ArticlesController.php

public function isAuthorized($user)
{
    // All registered users can add articles
    if ($this->request->action === 'add') {
        return true;
    }

    // The owner of an article can edit and delete it
    if (in_array($this->request->action, ['edit', 'delete'])) {
        $articleId = (int)$this->request->params['pass'][0];
        if ($this->Articles->isOwnedBy($articleId, $user['id'])) {
            return true;
        }
    }

    return parent::isAuthorized($user);
}
```

Estamos sobreescribiendo el método `isAuthorized()` de AppController y comprobando si la clase padre autoriza al usuario. Si no lo hace entonces solamente autorizarlo a acceder a la acción add y condicionalmente acceder a edit y delete. Una última cosa por implementar, decidir si el usuario está autorizador a editar el post o no, estamos llamando la función `isOwnedBy()` del modelo Articles. Es en general una buena practica mover la mayor parte de la logica posible hacia los modelos:

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

public function isOwnedBy($articleId, $userId)
{
    return $this->exists(['id' => $articleId, 'user_id' => $userId]);
}
```

Esto concluye nuestro simple tutorial de autenticación y autorización. Para proteger el UsersController se puede seguir la misma técnica utilizada para ArticlesController. También es posible implementar una solución mas general en AppController, de acuerdo a tus reglas.

En caso de necesitar más control, sugerimos leer la guia completa sobre Auth en *Authentication*, donde encontrarás mas información para configurar el componente y crear clases de autorizacion a tú medida.

Lectura sugerida

1. *Code Generation with Bake* Generar código CRUD básico
2. *Authentication*: Registro y login de usuarios

CAPÍTULO 5

Contribuye

Existen varias maneras en las que puedes contribuir con CakePHP. Las siguientes secciones cubren las diferentes maneras en las que puedes ayudarnos:

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[18] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Documentación

Contribuir con la documentación es sencillo. Los archivos están hospedados en https://github.com/cakephp/docs. Siéntete libre de hacer un fork del repositorio, añadir tus cambios, mejoras y traducciones y comenzar a ayudar a través de un nuevo pull request. Puedes también editar los archivos directamente en GitHub, sin la necesidad de descargar y correr el proyecto de manera local mediante el botón de "Improve this Doc" que aparece en todas los documentos del Cookbook; esto te llevará al editor online de GitHub de esa página para que comiences a colaborar.

La documentación de CakePHP cuenta con integración continua[19], así que puedes er el status de varias ejecuciones[20] en el servidor de Jenkins cuando quieras.

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

[18] https://github.com/cakephp/docs
[19] https://es.wikipedia.org/wiki/Integraci%C3%B3n_continua
[20] https://ci.cakephp.org

Por favor, siéntase libre de enviarnos un pull request en Github[21] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Tickets

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[22] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Code

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[23] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Coding Standards

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[24] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[21] https://github.com/cakephp/docs
[22] https://github.com/cakephp/docs
[23] https://github.com/cakephp/docs
[24] https://github.com/cakephp/docs

Backwards Compatibility Guide

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[25] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[25] https://github.com/cakephp/docs

CAPÍTULO 6

Instalación

CakePHP se instala rápida y fácilmente. Los requisitos mínimos son un servidor web y una copia de CakePHP, y ya! Aunque este manual se enfoca principalmente en configurar Apache (ya que es el más utilizado), puedes configurar CakePHP para que corra con una variedad de servidores web como nginx, LightHTHPD o Microsoft IIS.

Requisitos

- Servidor HTTP. Por ejemplo: Apache. mod_rewrite es recomendado, pero no requerido.
- PHP 5.5.9 o mayor.
- extensión mbstring.
- extensión intl.

Técnicamente una base de datos no es necesaria, pero imaginamos que la mayoría de aplicaciones utiliza alguna. CakePHP soporta una gran variedad de sistemas de bases de datos:

- MySQL (5.1.10 o mayor).
- PostgreSQL.
- Microsoft SQL Server (2008 o mayor).
- SQLite 3.

Nota: Todos los drivers nativos necesitan PDO. Debes asegurarte de tener las extensiones de PDO correctas.

Licencia

CakePHP está licenciado bajo la Licencia MIT[26]. Esto significa que eres libre para modificar, distribuir y republicar el código fuente con la condición de que las notas de copyright queden intactas. También eres libre para incorporar CakePHP en cualquier aplicación comercial o de código cerrado.

Instalando CakePHP

CakePHP utiliza Composer[27], una herramienta de manejo de dependencias para PHP 5.3+, como el método de instalación oficialmente soportado.

Primero, necesitas descargar e instalar Composer, si no lo has hecho ya. Si tienes instalado cURL, es tan fácil como correr esto en un terminal:

```
curl -s https://getcomposer.org/installer | php
```

O, puedes descargar `composer.phar` desde el sitio web de Composer[28].

Para sistemas Windows, puedes descargar el Instalador de Composer para Windows aquí[29]. Para más instrucciones acerca de esto, puedes leer el README del instalador de Windows aquí[30].

Ya que has descargado e instalado Composer puedes generar una aplicación CakePHP ejecutando:

```
php composer.phar create-project --prefer-dist cakephp/app [app_name]
```

O si tienes Composer definido globalmente:

```
composer create-project --prefer-dist cakephp/app [app_name]
```

Una vez que Composer termine de descargar el esqueleto y la librería core de CakePHP, deberías tener una aplicación funcional de CakePHP instalada vía Composer. Asegúrate de que los ficheros composer.json y composer.lock se mantengan junto con el resto de tu código fuente.

Ahora puedes visitar el destino donde instalaste la aplicación y ver los diferentes avisos tipo semáforo de los ajustes.

Mantente al día con los últimos cambios de CakePHP

Si quieres mantenerte al corriente de los últimos cambios en CakePHP puedes añadir las siguientes líneas al `composer.json` de tu aplicación:

```
"require": {
    "cakephp/cakephp": "dev-master"
}
```

[26] http://www.opensource.org/licenses/mit-license.php
[27] http://getcomposer.org
[28] https://getcomposer.org/download/
[29] https://github.com/composer/windows-setup/releases/
[30] https://github.com/composer/windows-setup

Donde `<branch>` es el nombre del branch que quieres seguir. Cada vez que ejecutes `php composer.phar update` recibirás las últimas actualizaciones del branch seleccionado.

Permisos

CakePHP utiliza el directorio **tmp** para varias operaciones. Descripciones de Modelos, el caché de las vistas y la información de la sesión son algunos ejemplos de lo anterior. El directorio **logs** es utilizado para para escribir ficheros de log por el motor de `FileLog` por defecto.

Asegúrate de que los directorios **logs**, **tmp** y todos sus subdirectorios tengan permisos de escritura por el usuario del Servidor Web. La instalación de CakePHP a través de Composer se encarga de este proceso haciendo que dichos directorios tengan los permisos abiertos globalmente con el fin de que puedas tener ajustado todo de manera más rápida. Obviamente es recomendable que revises, y modifiques si es necesario, los permisos tras la instalación vía Composer para mayor seguridad.

Un problema común es que **logs**, **tmp** y sus subdirectorios deben poder ser modificados tanto por el usuario del Servidor Web como por el usuario de la línea de comandos. En un sistema UNIX, si los usuarios mencionados difieren, puedes ejecutar los siguientes comandos desde el directorio de tu aplicación para asegurarte de que todo esté configurado correctamente:

```
HTTPDUSER=`ps aux | grep -E '[a]pache|[h]ttpd|[_]www|[w]ww-data|[n]ginx' |␣
↪grep -v root | head -1 | cut -d\  -f1`
setfacl -R -m u:${HTTPDUSER}:rwx tmp
setfacl -R -d -m u:${HTTPDUSER}:rwx tmp
setfacl -R -m u:${HTTPDUSER}:rwx logs
setfacl -R -d -m u:${HTTPDUSER}:rwx logs
```

Configuración

Configurar una aplicación de CakePHP puede ser tan simple como colocarla en el directorio raíz de tu Servidor Web, o tan complejo y flexible como lo desees. Esta sección cubrirá los dos tipos principales de instalación de CakePHP: Desarrollo y Producción.

- Desarrollo: fácil de arrancar, las URLs de la aplicación incluyen el nombre del directorio de la aplicación de CakePHP y es menos segura.
- Producción: Requiere tener la habilidad de configurar el directorio raíz del Servidor Web, cuenta con URLs limpias y es bastante segura.

Desarrollo

Este es el método más rápido para configurar CakePHP. En este ejemplo utilizaremos la consola de CakePHP para ejecutar el servidor web nativo de PHP para hacer que tu aplicación esté disponible en **http://host:port**. Para ello ejecuta desde el directorio de la aplicación:

```
bin/cake server
```

Por defecto, sin ningún argumento, esto colocará tu aplicación en **http://localhost:8765/**.

Si tienes algún conflicto con **localhost** o el puerto **8765**, puedes indicarle a la consola de CakePHP que corra el servidor de manera más específica utilizando los siguientes argumentos:

```
bin/cake server -H 192.168.13.37 -p 5673
```

Esto colocará tu aplicación en **http://192.168.13.37:5673/**.

¡Eso es todo! Tu aplicación de CakePHP está corriendo perfectamente sin tener que haber configurado el servidor web manualmente.

Advertencia: El servidor de desarrollo *nunca* debe ser utilizado en un ambiente de producción. Se supone que esto es un servidor básico de desarrollo y nada más.

Si prefieres usar un servidor web "real", Debes poder mover todos tus archivos de la instalación de CakePHP (incluyendo los archivos ocultos) dentro la carpeta raíz de tu servidor web. Debes entonces ser capaz de apuntar tu navegador al directorio donde moviste los archivos y ver tu aplicación en acción.

Producción

Una instalación de producción es una manera más flexible de montar una aplicación de CakePHP. Utilizando este método, podrás tener un dominio entero actuando como una sola aplicación de CakePHP. Este ejemplo te ayudará a instalar CakePHP donde quieras en tu sistema de ficheros y tenerlo disponible en `http://www.example.com`. Toma en cuenta que esta instalación requiere que tengas los derechos de cambiar el directorio raíz (`DocumentRoot`) del servidor web Apache.

Después de instalar tu aplicación utilizando cualquiera de los métodos mencionados en el directorio elegido - asumiremos que has escogido /cake_install - tu estructura de ficheros debe ser la siguiente:

```
/cake_install/
    bin/
    config/
    logs/
    plugins/
    src/
    tests/
    tmp/
    vendor/
    webroot/ (este directorio es ajutado como el DocumentRoot)
    .gitignore
    .htaccess
    .travis.yml
    composer.json
    index.php
```

```
    phpunit.xml.dist
    README.md
```

Si utilizas Apache debes configurar la directiva `DocumentRoot` del dominio a:

```
DocumentRoot /cake_install/webroot
```

Si tu configuración del Servidor Web es correcta debes tener tu aplicación disponible ahora en http://www.example.com.

A rodar!

Muy bien, ahora veamos a CakePHP en acción. Dependiendo de los ajustes que hayas utilizado, deberías dirigirte en tu navegador a http://example.com/ o http://localhost:8765/. En este punto, encontrarás la página principal de CakePHP y un mensaje que te dice el estado actual de tu conexión a la base de datos.

¡Felicidades! Estás listo para *Crear tu primera aplicación de CakePHP*.

URL Rewriting

Apache

Mientras que CakePHP está diseñado para trabajar con mod_rewrite recién sacado del horno, usualmente hemos notado que algunos usuarios tienen dificultades para lograr que todo funcione bien en sus sistemas.

Aquí hay algunas cosas que puedes tratar de conseguir para que funcione correctamente. La primera mirada debe ir a httpd.conf. (Asegura de que estás editando el httpd.conf del sistema en lugar del httpd.conf de un usuario o sitio específico)

Hay archivos que pueden variar entre diferentes distribuciones y versiones de Apache. Debes también mirar en http://wiki.apache.org/httpd/DistrosDefaultLayout para obtener información.

1. Asegura de que un archivo .htaccess de sobreescritura esté permitido y que *AllowOverride* esté ajustado en *All* para el correcto *DocumentRoot*. Debes ver algo similar a:

```
# Cada directorio al que Apache puede acceder puede ser configurado
# con sus respectivos permitidos/denegados servicios y características
# en ese directorios (y subdirectorios).
#
# Primero, configuramos el por defecto para ser muy restrictivo con sus
# ajustes de características.
<Directory />
    Options FollowSymLinks
    AllowOverride All
#    Order deny,allow
#    Deny from all
</Directory>
```

2. Asegura que tu estás cargando mod_rewrite correctamente. Debes ver algo similar a esto:

```
LoadModule rewrite_module libexec/apache2/mod_rewrite.so
```

   En muchos sistemas esto estará comentado por defecto, así que solo debes remover el símbolo # al comienzo de la línea.

   Después de hacer los cambios, reinicia Apache para asegurarte que los ajustes estén activados.

   Verifica que tus archivos .htaccess está actualmente en directorio correcto. Algunos sistemas operativo tratan los archivos que empiezan con '.' como oculto y por lo tanto no podrás copiarlos.

3. Asegúrate que tu copia de CakePHP provenga desde la sección descargas del sitio o de nuestro repositorio de Git, y han sido desempacados correctamente, revisando los archivos .htaccess.

   El directorio app de CakePHP (Será copiado en la raíz de tu aplicación por bake):

```
<IfModule mod_rewrite.c>
   RewriteEngine on
   RewriteRule    ^$    webroot/    [L]
   RewriteRule    (.*) webroot/$1    [L]
</IfModule>
```

   El directorio webroot de CakePHP (Será copiado a la raíz de tu aplicación web por bake):

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]
</IfModule>
```

   Si tu sitio aún tiene problemas con mod_rewrite, querrás probar modificar los ajustes para el Servidor Virtual. En Ubuntu, edita el archivo **/etc/apache2/sites-available/default** (la ubicación depende de la distribución). En este archivo, debe estar `AllowOverride None` cambiado a''AllowOverride All'', así tendrás:

```
<Directory />
    Options FollowSymLinks
    AllowOverride All
</Directory>
<Directory /var/www>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order Allow,Deny
    Allow from all
</Directory>
```

   En macOS, otra solución es usar la herramienta virtualhostx[31] para crear servidores virtuales y apuntarlos a tu carpeta.

   Para muchos servicios de alojamiento (GoDaddy, 1and1), tu servidor web estará actualmente sirviendo desde un directorio de usuario que actualmente usa mod_rewrite. Si tu estás instalando CakePHP

[31] http://clickontyler.com/virtualhostx/

en la carpeta de usuario (http://example.com/~username/cakephp/), o alguna otra estructura de URL que ya utilice mod_rewrite, necesitarás agregar una declaración a los archivos .htaccess que CakePHP usa (.htaccess, webroot/.htaccess).

Esto puede ser agregado a la misma sección con la directiva RewriteEngine, entonces por ejemplo, tu .htaccess en el webroot debería verse algo así:

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /path/to/app
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]
</IfModule>
```

Los detalles de estos cambios dependerán de tu configuración, y puede incluir algunas líneas adicionales que no están relacionadas con CakePHP. Por favor dirígete a la documentación en línea de Apache para más información.

4. (Opcional) Para mejorar la configuración de producción, debes prevenir archivos adicionales inválidos que sean tomados por CakePHP. Modificando tu .htaccess del webroot a algo cómo esto:

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /path/to/app/
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_URI} !^/(webroot/)?(img|css|js)/(.*)$
    RewriteRule ^ index.php [L]
</IfModule>
```

Lo anterior simplemente previene que archivos adicionales incorrectos sean enviados a index.php en su lugar muestre la página 404 de tu servidor web.

Adicionalmente puedes crear una página 404 que concuerde, o usar la página 404 incluida en CakePHP agregando una directiva `ErrorDocument`:

```
ErrorDocument 404 /404-not-found
```

nginx

nginx no hace uso de un archivo .htaccess como Apache, por esto es necesario crear la reescritura de URL en la configuraciones de *site-available*. Esto usualmente se encuentra en `/etc/nginx/sites-available/your_virtual_host_conf_file`. Dependiendo de la configuración, tu necesitarás modificar esto, pero por lo menos, necesitas PHP corriendo como una instancia FastCGI:

```
server {
    listen   80;
    server_name www.example.com;
    rewrite ^(.*) http://example.com$1 permanent;
}
```

```
server {
    listen   80;
    server_name example.com;

    # root directive should be global
    root   /var/www/example.com/public/webroot/;
    index  index.php;

    access_log /var/www/example.com/log/access.log;
    error_log /var/www/example.com/log/error.log;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        try_files $uri =404;
        include /etc/nginx/fastcgi_params;
        fastcgi_pass    127.0.0.1:9000;
        fastcgi_index   index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }
}
```

En algunos servidores (Como Ubuntu 14.04) la configuración anterior no funcionará recién instalado, y de todas formas la documentación de nginx recomienda una forma diferente de abordar esto (http://nginx.org/en/docs/http/converting_rewrite_rules.html). Puedes intentar lo siguiente (Notarás que esto es un bloque de servidor {}, en vez de dos, pese a que si quieres que example.com resuelva a tu aplicación CakePHP en adición a www.example.com consulta el enlace de nginx anterior):

```
server {
    listen   80;
    server_name www.example.com;
    rewrite 301 http://www.example.com$request_uri permanent;

    # root directive should be global
    root   /var/www/example.com/public/webroot/;
    index  index.php;

    access_log /var/www/example.com/log/access.log;
    error_log /var/www/example.com/log/error.log;

    location / {
        try_files $uri /index.php?$args;
    }

    location ~ \.php$ {
        try_files $uri =404;
        include /etc/nginx/fastcgi_params;
        fastcgi_pass    127.0.0.1:9000;
        fastcgi_index   index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
```

```
    }
}
```

IIS7 (Windows)

IIS7 no soporta de manera nativa los archivos .htaccess. Mientras hayan *add-ons* que puedan agregar soporte a estos archivos, puedes también importar las reglas htaccess en IIS para usar las redirecciones nativas de CakePHP. Para hacer esto, sigue los siguientes pasos:

1. Usa el Intalador de plataforma Web de Microsoft[32] para instalar el Modulo de Redirreción 2.0[33] de URLs o descarga directamente (32-bit[34] / 64-bit[35]).
2. Crear un nuevo archivo llamado web.config en tu directorio de raíz de CakePHP.
3. Usando Notepad o cualquier editor de XML, copia el siguiente código en tu nuevo archivo web.config:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <rewrite>
            <rules>
                <rule name="Exclude direct access to webroot/*"
                  stopProcessing="true">
                    <match url="^webroot/(.*)$" ignoreCase="false" />
                    <action type="None" />
                </rule>
                <rule name="Rewrite routed access to assets(img, css, files,␣
↪js, favicon)"
                  stopProcessing="true">
                    <match url="^(img|css|files|js|favicon.ico)(.*)$" />
                    <action type="Rewrite" url="webroot/{R:1}{R:2}"
                      appendQueryString="false" />
                </rule>
                <rule name="Rewrite requested file/folder to index.php"
                  stopProcessing="true">
                    <match url="^(.*)$" ignoreCase="false" />
                    <action type="Rewrite" url="index.php"
                      appendQueryString="true" />
                </rule>
            </rules>
        </rewrite>
    </system.webServer>
</configuration>
```

Una vez el archivo web.config es creado con las reglas de redirección amigables de IIS, los enlaces, CSS, JavaScript y redirecciones de CakePHP deberían funcionar correctamente.

[32] http://www.microsoft.com/web/downloads/platform.aspx
[33] http://www.iis.net/downloads/microsoft/url-rewrite
[34] http://www.microsoft.com/en-us/download/details.aspx?id=5747
[35] http://www.microsoft.com/en-us/download/details.aspx?id=7435

No puedo usar Redireccionamientos de URL

Si no quieres o no puedes obtener mod_rewirte (o algun otro modulo compatible) en el servidor a correr, necesitarás usar el decorador de URL incorporado en CakePHP. En **config/app.php**, descomentar la línea para que se vea así:

```
'App' => [
    // ...
    // 'baseUrl' => env('SCRIPT_NAME'),
]
```

También remover estos archivos .htaccess:

```
/.htaccess
webroot/.htaccess
```

Esto hará tus URL verse así www.example.com/index.php/controllername/actionname/param antes que www.example.com/controllername/actionname/param.

CAPÍTULO 7

Configuration

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[36] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[36] https://github.com/cakephp/docs

CAPÍTULO 8

Routing

class `Cake\Routing\`**`Router`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[37] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Connecting Routes

Using Named Routes

Dispatcher Filters

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[38] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

---

[37] https://github.com/cakephp/docs
[38] https://github.com/cakephp/docs

CAPÍTULO 9

Request & Response Objects

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[39] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[39] https://github.com/cakephp/docs

CAPÍTULO 10

Controllers

class `Cake\Controller\`**`Controller`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[40] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

More on Controllers

The Pages Controller

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[41] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Components

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

[40] https://github.com/cakephp/docs
[41] https://github.com/cakephp/docs

Por favor, siéntase libre de enviarnos un pull request en Github[42] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Authentication

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[43] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

CookieComponent

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[44] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Cross Site Request Forgery

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[45] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[42] https://github.com/cakephp/docs
[43] https://github.com/cakephp/docs
[44] https://github.com/cakephp/docs
[45] https://github.com/cakephp/docs

FlashComponent

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[46] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Security

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[47] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Pagination

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[48] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Request Handling

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[49] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

[46] https://github.com/cakephp/docs
[47] https://github.com/cakephp/docs
[48] https://github.com/cakephp/docs
[49] https://github.com/cakephp/docs

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

CAPÍTULO 11

Views

class `Cake\View\`**`View`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[50] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

More About Views

View Cells

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[51] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Themes

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

[50] https://github.com/cakephp/docs
[51] https://github.com/cakephp/docs

Por favor, siéntase libre de enviarnos un pull request en Github[52] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

JSON and XML views

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[53] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Helpers

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[54] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

FlashHelper

class `Cake\View\Helper\`**`FlashHelper`**(*View $view*, *array $config* = [])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[55] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[52] https://github.com/cakephp/docs
[53] https://github.com/cakephp/docs
[54] https://github.com/cakephp/docs
[55] https://github.com/cakephp/docs

FormHelper

class Cake\View\Helper**FormHelper** (*View $view*, *array $config* =[ ])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[56] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

HtmlHelper

class Cake\View\Helper**HtmlHelper** (*View $view*, *array $config* =[ ])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[57] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

NumberHelper

class Cake\View\Helper**NumberHelper** (*View $view*, *array $config* =[ ])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[58] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

PaginatorHelper

class Cake\View\Helper**PaginatorHelper** (*View $view*, *array $config* =[ ])

[56] https://github.com/cakephp/docs
[57] https://github.com/cakephp/docs
[58] https://github.com/cakephp/docs

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[59] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

RSS

class `Cake\View\Helper\`**`RssHelper`**(*View $view*, *array $config* =[ ])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[60] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

SessionHelper

class `Cake\View\Helper\`**`SessionHelper`**(*View $view*, *array $config* =[ ])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[61] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

TextHelper

class `Cake\View\Helper\`**`TextHelper`**(*View $view*, *array $config* =[ ])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[62] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

[59] https://github.com/cakephp/docs
[60] https://github.com/cakephp/docs
[61] https://github.com/cakephp/docs
[62] https://github.com/cakephp/docs

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

TimeHelper

class Cake\View\Helper**TimeHelper**(*View $view*, *array $config* =[])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[63] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

UrlHelper

class Cake\View\Helper**UrlHelper**(*View $view*, *array $config* =[])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[64] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[63] https://github.com/cakephp/docs
[64] https://github.com/cakephp/docs

CAPÍTULO 12

Acceso a la base de datos & ORM

En CakePHP el acceso a la base de datos se hace por medio de dos objetos primarios. El primero son **repositories** -repositorios- o **table objects** -objetos de tabla-. Estos objetos proveen acceso a colecciones de datos. Nos permiten guardar nuevos registros, modificar y borrar existentes, definir relaciones y realizar operaciones en masa. El segundo tipo de objeto son **entities** -entidades-. Las Entidades representan registros individuales y permiten definir funcionalidad y comportamiento a nivel de registro/fila.

Estas dos clases son responsables de manejar todo lo que sucede con datos, validez, interacción y evolución en tu área de trabajo.

El ORM incluído en CakePHP se especializa en base de datos relacionales, pero puede ser extendido para soportar alternativas.

El ORM de CakePHP toma ideas y conceptos de los modelos ActiveRecord y Datamapper. Aspira a crear una implementación híbrida que combine aspectos de los dos modelos para crear un ORM rápido y fácil de usar.

Antes de comentar explorando el ORM, asegurate de configurar tu conexion *configure your database connections*.

Nota: Si estás familiarizado con las versiones anteriores de CakePHP, deberías leer *New ORM Upgrade Guide* para entender las diferencias entre CakePHP 3.0 y las versiones anteriores.

Ejemplo rápido

Para comenzar no es necesario escribir código. Si has seguido las convenciones de nombres para las tablas puedes comenzar a utilizar el ORM. Por ejemplo si quisieramos leer datos de nuestra tabla `articles`:

```
use Cake\ORM\TableRegistry;
$articles = TableRegistry::get('Articles');
$query = $articles->find();
foreach ($query as $row) {
```

```
    echo $row->title;
}
```

Como se ve, no es necesario agregar código extra ni ninguna otra configuración, gracias al uso de las convenciones de CakePHP. Si quisieramos modificar nuestra clase ArticlesTable para agregar asociaciones o definir métodos adicionales deberiamos agregar las siguientes lineas en **src/Model/Table/ArticlesTable.php**

```
namespace App\Model\Table;

use Cake\ORM\Table;

class ArticlesTable extends Table
{

}
```

Las clases Table usan una version en CamelCase del nombre de la tabla, con el sufijo `Table`. Una vez que tú clase fue creada, puedes obtener una referencia a esta usando `ORM\TableRegistry` como antes:

```
use Cake\ORM\TableRegistry;
// Now $articles is an instance of our ArticlesTable class.
$articles = TableRegistry::get('Articles');
```

Ahora que tenemos una clase Table concreta, probablemente querramos usar una clase Entity concreta. Las clases Entity permiten definir métodos de acceso y mutación, lógica para registros individuales y mucho mas. Comenzaremos agregando las siguientes lineas en **src/Model/Entity/Article.php**:

```
namespace App\Model\Entity;

use Cake\ORM\Entity;

class Article extends Entity
{

}
```

Las Entity usan la version CamelCase en singular del nombre de la tabla como su nombre. Ahora que hemos creado una clase Entity, cuando carguemos entidades de nuestra base de datos, vamos a obtener instancias de nuestra clase Article:

```
use Cake\ORM\TableRegistry;

// Now an instance of ArticlesTable.
$articles = TableRegistry::get('Articles');
$query = $articles->find();

foreach ($query as $row) {
    // Each row is now an instance of our Article class.
    echo $row->title;
}
```

CakePHP usa convenciones de nombres para asociar las clases Table y Entity. Si necesitas modificar qué entidad utilizada una tabla, puedes usar el método `entityClass()` para especificar el nombre de una clase.

Vea *Table Objects* y *Entities* para mas información sobre como utilizar objetos Table y Entity en su aplicación.

Más información

Database Basics

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[65] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Query Builder

class `Cake\ORM\`**`Query`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[66] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Table Objects

class `Cake\ORM\`**`Table`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[67] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

[65] https://github.com/cakephp/docs
[66] https://github.com/cakephp/docs
[67] https://github.com/cakephp/docs

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Entities

class Cake\ORM**Entity**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[68] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Retrieving Data & Results Sets

class Cake\ORM**Table**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[69] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Validating Data

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

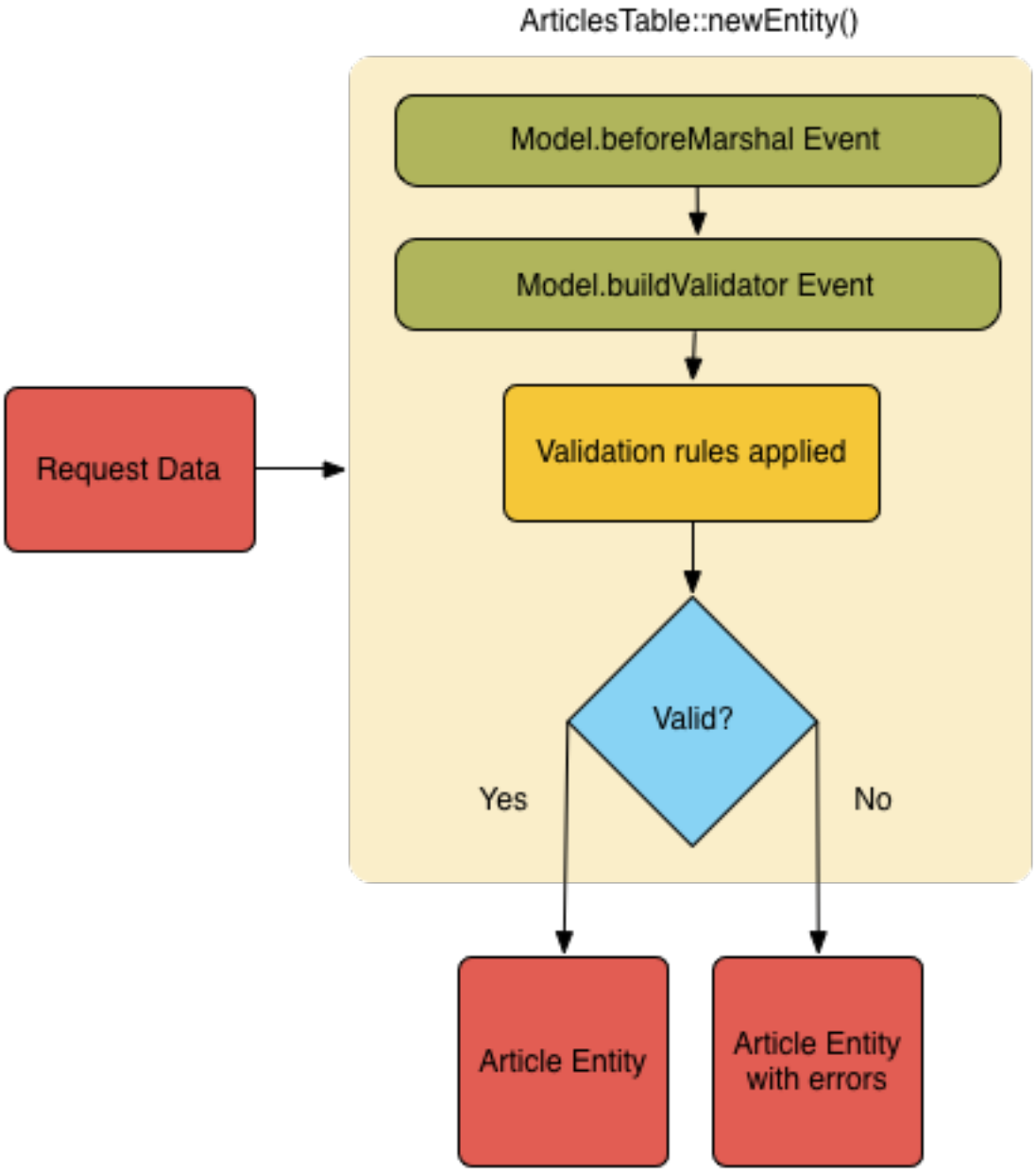
Por favor, siéntase libre de enviarnos un pull request en Github[70] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

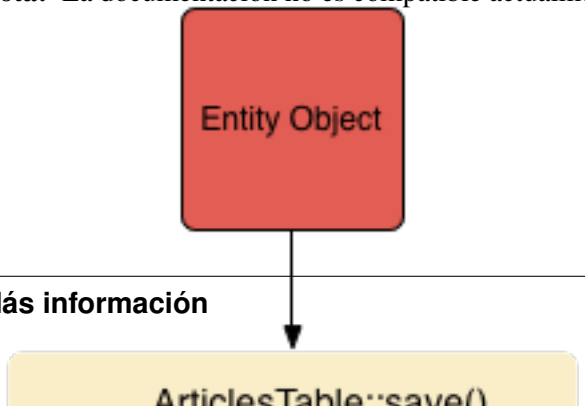
[68] https://github.com/cakephp/docs
[69] https://github.com/cakephp/docs
[70] https://github.com/cakephp/docs

Saving Data

class `Cake\ORM\`**`Table`**



Nota: La documentación no es compatible actualmente con el idioma español en esta página.



Por favor, siéntase libre de enviarnos un pull request

en Github[71] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Deleting Data

class Cake\ORM**Table**

Cake\ORM\Table::**delete**(*E*

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[72] o utilizar el botón **Improve this Doc** para

[71] https://github.com/cakephp/docs
[72] https://github.com/cakephp/docs

proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Associations - Linking Tables Together

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[73] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de

[73] https://github.com/cakephp/docs

selección superior para obtener información sobre el tema de esta página.

Behaviors

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[74] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[74] https://github.com/cakephp/docs

Core Behaviors

CounterCache Behavior

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[75] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Timestamp Behavior

class `Cake\Model\Behavior\`

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

[75] https://github.com/cakephp/docs

Por favor, siéntase libre de enviarnos un pull request en Github[76] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Translate

class `Cake\Model\Behavior\`

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[77] o utilizar el botón **Improve this**

[76] https://github.com/cakephp/docs
[77] https://github.com/cakephp/docs

Doc para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

TreeBehavior

class `Cake\Model\Behavior\`

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[78] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en

[78] https://github.com/cakephp/docs

el menú de selección superior para obtener información sobre el tema de esta página.

Schema System

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[79] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[79] https://github.com/cakephp/docs

ORM Cache Shell

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[80] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[80] https://github.com/cakephp/docs

CAPÍTULO 13

Bake Console

Code Generation with Bake

La consola de CAKE se ejecuta usando PHP CLI (command line interface). Si tiene problemas para ejecutar el script, asegurese de:

1. Tener instalado el PHP CLI y que estén los módulos correspondientes habilitados (ej: MySQL y intl).
2. Si el host de base de datos es 'localhost', intente realizar la conexión con el ip '127.0.0.1'. En algunos casos PHP CLI tiene problemas al referenciar por nombre de host (localhost).
3. Dependiendo de como su computadora este configurada, la ejecución del comando CAKE BAKE (cake bash script) puede requerir que permisos de ejecución al lanzar `bin/cake bake`.

Antes de comenzar la ejecución, asegurese de disponer al menos una conexion a una base de datos configurada. Ver sección *database configuration* para mas información.

A fin de comenzar con la ejecución del comando, debe abrir la consola de windows y ejecutar "Cake Bake"

1. Ir a Inicio (Start) > Ejecutar (Run)
2. Escribir "cmd" y presionar 'Enter'
3. Navegar hasta llegar a la carpeta de instalación del cake
4. Acceder a la carpeta 'bin'
5. Escribir 'Cake bake' lo cual deberá devolver un listado con todas las tareas/actividades disponibles.

El resultado debería ser algo similar a lo siguiente:

```
$ bin/cake bake

Welcome to CakePHP v3.1.6 Console
---------------------------------------------------------------
App : src
Path: /var/www/cakephp.dev/src/
PHP: 5.5.8
```

```
---------------------------------------------------------------
The following commands can be used to generate skeleton code for your␣
↪application.

Available bake commands:

- all
- behavior
- cell
- component
- controller
- fixture
- form
- helper
- mailer
- migration
- migration_snapshot
- model
- plugin
- shell
- shell-helper
- template
- test

By using 'cake bake [name]' you can invoke a specific bake task.
```

Puede obtener más información sobre lo que realiza cada una de las actividades y sus opciones usando el parametro '–help' option:

```
$ bin/cake bake controller --help

Welcome to CakePHP v3.1.6 Console
---------------------------------------------------------------
App : src
Path: /var/www/cakephp.dev/src/
---------------------------------------------------------------
Bake a controller skeleton.

Usage:
cake bake controller [subcommand] [options] [<name>]

Subcommands:

all  Bake all controllers with CRUD methods.

To see help on a subcommand use `cake bake controller [subcommand] --help`

Options:

--help, -h        Display this help.
--verbose, -v     Enable verbose output.
--quiet, -q       Enable quiet output.
--plugin, -p      Plugin to bake into.
```

```
--force, -f       Force overwriting existing files without prompting.
--connection, -c  The datasource connection to get data from.
                  (default: default)
--theme, -t       The theme to use when baking code.
--components      The comma separated list of components to use.
--helpers         The comma separated list of helpers to use.
--prefix          The namespace/routing prefix to use.
--no-test         Do not generate a test skeleton.
--no-actions      Do not generate basic CRUD action methods.

Arguments:

name  Name of the controller to bake. Can use Plugin.name to bake
    controllers into plugins. (optional)
```

Temas Bake / Templates

La opción `theme` es genérica para todos los comandos bake y permite cambiar los templates de bake utilizados para generar los archivos finales. Para crear sus propios templates, ver *bake theme creation documentation*.

Extending Bake

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[81] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[82] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

---

[81] https://github.com/cakephp/docs
[82] https://github.com/cakephp/docs

CAPÍTULO 14

Caching

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[83] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[83] https://github.com/cakephp/docs

CAPÍTULO 15

Shells, Tasks & Console Tools

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[84] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

More Topics

Shell Helpers

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[85] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Interactive Console (REPL)

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

[84] https://github.com/cakephp/docs
[85] https://github.com/cakephp/docs

Por favor, siéntase libre de enviarnos un pull request en Github[86] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Running Shells as Cron Jobs

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[87] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

I18N Shell

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[88] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Completion Shell

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[89] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[86] https://github.com/cakephp/docs
[87] https://github.com/cakephp/docs
[88] https://github.com/cakephp/docs
[89] https://github.com/cakephp/docs

Plugin Shell

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[90] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Routes Shell

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[91] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Upgrade Shell

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[92] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Server Shell

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[93] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

[90] https://github.com/cakephp/docs
[91] https://github.com/cakephp/docs
[92] https://github.com/cakephp/docs
[93] https://github.com/cakephp/docs

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Cache Shell

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[94] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[94] https://github.com/cakephp/docs

CAPÍTULO 16

Debugging

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[95] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[95] https://github.com/cakephp/docs

CAPÍTULO 17

ES - Deployment

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[96] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[96] https://github.com/cakephp/docs

CAPÍTULO 18

Email

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[97] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[97] https://github.com/cakephp/docs

CAPÍTULO 19

Error & Exception Handling

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[98] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[98] https://github.com/cakephp/docs

CAPÍTULO 20

Events System

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[99] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[99] https://github.com/cakephp/docs

CAPÍTULO 21

Internationalization & Localization

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[100] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[100] https://github.com/cakephp/docs

CAPÍTULO 22

Logging

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[101] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[101] https://github.com/cakephp/docs

CAPÍTULO 23

Modelless Forms

class `Cake\Form\`**`Form`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[102] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[102] https://github.com/cakephp/docs

CAPÍTULO 24

Plugins

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[103] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[103] https://github.com/cakephp/docs

CAPÍTULO 25

REST

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[104] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[104] https://github.com/cakephp/docs

CAPÍTULO 26

Security

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[105] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Security

class `Cake\Utility\`**`Security`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[106] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[105] https://github.com/cakephp/docs
[106] https://github.com/cakephp/docs

CAPÍTULO 27

Sessions

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[107] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[107] https://github.com/cakephp/docs

CAPÍTULO 28

Testing

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[108] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[108] https://github.com/cakephp/docs

CAPÍTULO 29

Validation

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[109] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[109] https://github.com/cakephp/docs

CAPÍTULO 30

App Class

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[110] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[110] https://github.com/cakephp/docs

CAPÍTULO 31

Collections

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[111] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[111] https://github.com/cakephp/docs

CAPÍTULO 32

Folder & File

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[112] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[112] https://github.com/cakephp/docs

CAPÍTULO 33

Hash

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[113] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[113] https://github.com/cakephp/docs

CAPÍTULO 34

Http Client

class `Cake\Network\Http\`**`Client`**(*mixed $config* =[])

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[114] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

---

[114] https://github.com/cakephp/docs

CAPÍTULO 35

Inflector

class `Cake\Utility\`**`Inflector`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[115] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[115] https://github.com/cakephp/docs

CAPÍTULO 36

Number

class `Cake\I18n\`**`Number`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[116] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[116] https://github.com/cakephp/docs

CAPÍTULO 37

Registry Objects

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[117] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[117] https://github.com/cakephp/docs

CAPÍTULO 38

Text

class `Cake\Utility\`**`Text`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[118] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

---

[118] https://github.com/cakephp/docs

CAPÍTULO 39

Time

class `Cake\Utility\`**`Time`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[119] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

---

[119] https://github.com/cakephp/docs

CAPÍTULO 40

Xml

class `Cake\Utility\`**`Xml`**

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[120] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[120] https://github.com/cakephp/docs

CAPÍTULO 41

Constants & Functions

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[121] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[121] https://github.com/cakephp/docs

CAPÍTULO 42

Debug Kit

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[122] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[122] https://github.com/cakephp/docs

CAPÍTULO 43

Migrations

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[123] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

[123] https://github.com/cakephp/docs

CAPÍTULO 44

Apéndices

En los apéndices encontrarás información relacionada a las nuevas características introducidas en cada versión, así como también las guías de migración entre versiones.

Guía de Migración a 3.x

3.x Migration Guide

Migration guides contain information regarding the new features introduced in each version and the migration path between versions.

3.3 Migration Guide

3.3 Migration Guide

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[124] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

3.2 Migration Guide

3.2 Migration Guide

[124] https://github.com/cakephp/docs

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[125] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

3.1 Migration Guide

Guía de Migración a 3.1

CakePHP 3.1 es completamente compatible con la versión 3.0. Esta página contiene los cambios y mejoras hechas en la versión 3.1.

Enrutamiento

- La clase por de ruta por defecto ha sido cambiada a `DashedRoute` en el repositorio `cakephp/app`. Tu código base actual no se verá afectado por esto, pero es recomendable usar esta clase desde ahora en adelante.
- Opciones de prefijos de nombre fueron añadidos a varios métodos del constructor de ruta. Ver la sección *Using Named Routes* para obtener más información.

Consola

- `Shell::dispatchShell()` ya no imprimirá el mensaje de bienvenida desde el intérprete de comandos emisor.
- La función ayudante `breakpoint()` ha sido añadida. Esta función provee un *snippet* de código que puede ser puesto en `eval()` para disparar una consola interactiva. Esta función es muy útil cuando se depuran los casos de prueba, o otros scripts desde la línea de comandos interactiva (CLI).
- Las opciones de consola `--verbose` y `--quiet` ahora controlan stdout/stderr como niveles de registros de salida.

Ayudantes agregados para la línea de comandos

- Las aplicaciones de consola ahora pueden crear clases ayudantes que encapsulan bloques de salida lógica. Ver la sección *Shell Helpers* para mayor información.

[125] https://github.com/cakephp/docs

RoutesShell

- RoutesShell ha sido añadido y ahora te permite un uso simple para depurar y testear desde CLI. Ver la sección *Routes Shell* para más información.

Controlador

- Las siguientes propiedades del Controlador están obsoletas:
  - layout
  - view - reemplazada con `template`
  - theme
  - autoLayout
  - viewPath - reemplazada con `templatePath`
  - viewClass - reemplazada con `className`
  - layoutPath

  En lugar de ajustar estas propiedades en tu controlador, debes ajustarlos en la vista usando el método con el nombre de la propiedad:

```
// En un controlador, en vez de
$this->layout = 'avanzado';

// Debes usar
$this->viewBuilder()->layout('avanzado');
```

Estos métodos deben ser llamados después de determinar que clase de vista será usada para un controlador/acción.

AuthComponent

- Una nueva opción de configuración `storage` ha sido añadida. Contiene el nombre de la clase de almacenamiento que `AuthComponent` utiliza para almacenar el registro de usuario, Por defecto se usa `SessionStorage`. Si se usa un autenticador sin estado debes configurar `AuthComponent` para que use `MemoryStorage` en su lugar.
- Una nueva opción de configuración `checkAuthIn` ha sido añadida. Contiene el nombre del evento que la autenticación debe comprobar una vez realizada. Por defecto `Controller.startup` es usado, pero tu puedes ajustar esto en `Controller.initialize` si deseas que la autenticación compruebe antes del método `beforeFilter()` del controlador a ejecutar.
- Las opciones `scope` y `contain` para las clases autenticadoras están obsoletas. En su lugar debes usar la opción `finder` para configurar un método localizador personalizado y modificar la consulta usada para buscar el usuario allí.

- La lógica para ajustar la variable de sesión `Auth.redirect`, que se usa para obtener la URL de redirección luego de iniciar sesión, ha cambiado. Ahora se establece solo cuando se intenta acceder a una URL protegida sin autenticación. Entonces `Auth::redirectUrl()` retornará la URL protegida después de iniciar sesión. Bajo circunstancias normales, cuando un usuario accede directamente a la página de inicio de sesión, `Auth::redirectUrl()` retornará el valor establecido en la configuración `loginRedirect`.

FlashComponent

- `FlashComponent` ahora apila los mensajes Flash cuando los ajustas con el método `set()` o `__call()`. Esto significa que la estructura en *Session* para guardar los mensajes Flash ha cambiado.

CsrfComponent

- El tiempo de expiración de la cookie CSRF ahora se podrá ajustar como un valor compatible con `strtotime()`.
- Los tokens inválidos CSRF ahora arrojarán una excepción `Cake\Network\Exception\InvalidCsrfTokenException` en vez de `Cake\Network\Exception\ForbiddenException`.

RequestHandlerComponent

- `RequestHandlerComponent` ahora intercambia el diseño y la plantilla basado en la extensión o la cabecera `Accept` en la llamada `beforeRender()` en lugar de `startup()`.
- `addInputType()` and `viewClassMap()` están obsoletos. En su lugar debes usar `config()` para modificar estas configuraciones en tiempo de ejecución.
- Cuando `inputTypeMap` o `viewClassMap` están definidas en el componente de configuraciones, *sobrescribirá* los valores por defecto. Este cambio permite remover las configuraciones por defecto.

Network

HttpClient

- El tipo mime por defecto usado para enviar peticiones ha cambiado. Previamente usaba siempre `multipart/form-data`. En la versión 3.1, `multipart/form-data` sólo es usado cuando hay archivos subidos presentes. Cuando no hay archivos subidos, `application/x-www-form-urlencoded` será usado en su lugar.

ORM

Ahora puedes *'Lazily Eager Load Associations'*. Esta característica te permite cargar asociaciones adicionales de manera condicional dentro del resultado ajustado, entidad o colección de entidades.

Los métodos `patchEntity()` y `newEntity()` ahora soportan la opción `onlyIds`. Esta opción te permite restringir que las asociaciones *hasMany/belongsToMany* sólo usen la lista `_ids`. Esta opción por defecto es `false`.

Query

- `Query::notMatching()` ha sido añadido.
- `Query::leftJoinWith()` ha sido añadido.
- `Query::innerJoinWith()` ha sido añadido.
- `Query::select()` ahora soporta los objetos `Table` y `Association` como parámetros. Estos tipos de parámetros seleccionarán todas las columnas de la tabla prevista o en la instancia asociada de la tabla de destino.
- `Query::distinct()` ahora acepta una cadena para diferenciar una sola columna.
- `Table::loadInto()` ha sido añadido.
- Las funciones nativas SQL `EXTRACT`, `DATE_ADD` y `DAYOFWEEK` han sido abstraídas a `extract()`, `dateAdd()` y `dayOfWeek()` respectivamente.

Vista

- Ahora puedes configurar `_serialized` a `true` para `JsonView` y `XmlView` y así serializar todas las variables en vez de especificar una por una.
- `View::$viewPath` está obsoleto. Debes usar `View::templatePath()` en su lugar.
- `View::$view` está obsoleto. Debes usar `View::template()` en su lugar.
- `View::TYPE_VIEW` está obsoleto. Debes usar `View::TYPE_TEMPLATE` en su lugar.

Helper

SessionHelper

- `SessionHelper` está obsoleta. Puedes usar `$this->request->session()` directamente.

FlashHelper

- `FlashHelper` ahora permite mostrar múltiples mensajes si fueron configuradas múltiples mensajes con `FlashComponent`. Cada mensaje será mostrado en su propio elemento. Los mensajes serán mostrados en el orden que fueron configurados.

FormHelper

- Nueva opción `templateVars` ha sido añadida. `templateVars` permite pasar variables adicionales a tu plantilla de control de formulario personalizado.

Email

- Las clases `Email` y `Transport` han sido movidas bajo el nombre de espacio `Cake\Mailer`. Sus antiguos espacios de nombre aún son utilizables como alias.
- El perfil `default` de email es ahora automáticamente ajustado cuando una instancia de `Email` cuando es creada. Este comportamiento es similar a como era en la versión 2.x.

Mailer

- La clase `Mailer` ha sido añadida. Esta clase ayuda a crear *emails* reusables en una aplicación.

I18n

Tiempo

- `Time::fromNow()` ha sido añadido. Este método hace fácil calcular la diferencia de tiempo desde 'ahora'.
- `Time::i18nFormat()` ahora soporta calendarios no-gregorianos cuando formatea fechas.

Validaciones

- `Validation::geoCoordinate()` ha sido añadido.
- `Validation::latitude()` ha sido añadido.
- `Validation::longitude()` ha sido añadido.
- `Validation::isInteger()` ha sido añadido.
- `Validation::ascii()` ha sido añadido.
- `Validation::utf8()` ha sido añadido.

3.0 Migration Guide

New ORM Upgrade Guide

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[126] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Información General

CakePHP Development Process

Nota: La documentación no es compatible actualmente con el idioma español en esta página.

Por favor, siéntase libre de enviarnos un pull request en Github[127] o utilizar el botón **Improve this Doc** para proponer directamente los cambios.

Usted puede hacer referencia a la versión en Inglés en el menú de selección superior para obtener información sobre el tema de esta página.

Glosario

arreglo de rutas Un arreglo de atributos que son pasados a `Router::url()`. Típicamente se ve algo así:

```
['controller' => 'Posts', 'action' => 'view', 5]
```

Atributos HTML Un arreglo con claves => valores que son colocados en los atributos HTML. Por ejemplo:

```
// Dado
['class' => 'mi-clase', 'target' => '_blank']

// Generará
class="mi-clase" target="_blank"
```

Si una opción puede usar su nombre como valor, entonces puede ser usado `true`:

[126] https://github.com/cakephp/docs
[127] https://github.com/cakephp/docs

```
// Dado
['checked' => true]

// Generará
checked="checked"
```

Sintaxis de plugin La sintáxis de plugin se refiere a el punto que separa los nombres de clases indicando que la clase es parte de un plugin:

```
// El plugin es "DebugKit", y el nombre de la clase es "Toolbar".
'DebugKit.Toolbar'

// El plugin es "AcmeCorp/Tools", y el nombre de clase es "Toolbar".
'AcmeCorp/Tools.Toolbar'
```

Notación de punto La notación de punto define una arreglo de rutas, separando los niveles anidados con `.` Por ejemplo:

```
Cache.default.engine
```

Apuntará a el siguiente valor:

```
[
    'Cache' => [
        'default' => [
            'engine' => 'File'
        ]
    ]
]
```

CSRF *Cross Site Request Forgery*. Previene los ataques de replay o playback, peticiones duplicadas y peticiones falsificadas desde otros dominios.

CDN *Content Delivery Network*. Le puedes pagar a un proveedor para que ayude a distribuir el contenido a centros de datos alrededor del mundo. Esto ayuda a poner elementos estáticos más cerca de tus usuarios geográficamente.

routes.php Un archivo en el directorio `config` que contiene las configuraciones de enrutamiento. Este archivo es incluido antes que cada petición sea procesada. Se deben conectar todas las rutas que necesita tu aplicación para que cada petición sea enrutada correctamente al controlador + acción.

DRY *Don't repeat yourself*. Es un principio de desarrollo de software orientado a reducir la repetición de la información de todo tipo. En CakePHP, DRY se utiliza para que pueda escribir las cosas una vez y reutilizarlos a través de su aplicación.

PaaS *Platform as a Service*. Las Plataforma como proveedores de servicios proporcionará recursos de hosting, bases de datos y almacenamiento en caché basado en la nube. Algunos proveedores populares incluyen Heroku, EngineYard y PagodaBox.

DSN *Data Source Name*. Una cadena de conexión formateada para que sea como una URI. CakePHP soporta conexiones DSN para Caché, Base de datos, Registro y de E-mail.

PHP Namespace Index

C

Cake\Collection, 123
Cake\Console, 91
Cake\Controller, 61
Cake\Database, 73
Cake\Database\Schema, 82
Cake\Form, 109
Cake\I18n, 135
Cake\Model\Behavior, 81
Cake\Network\Http, 131
Cake\ORM, 73
Cake\Routing, 57
Cake\Utility, 143
Cake\Validation, 119
Cake\View, 65
Cake\View\Helper, 69

Índice

A

arreglo de rutas, **157**
Atributos HTML, **157**

C

Cake\Collection (namespace), **123**
Cake\Console (namespace), **91**
Cake\Controller (namespace), **61**
Cake\Database (namespace), **73**
Cake\Database\Schema (namespace), **82**
Cake\Form (namespace), **109**
Cake\I18n (namespace), **135**
Cake\Model\Behavior (namespace), **79–81**
Cake\Network\Http (namespace), **131**
Cake\ORM (namespace), **73–76**
Cake\Routing (namespace), **57**
Cake\Utility (namespace), **115**, **133**, **139**, **141**, **143**
Cake\Validation (namespace), **119**
Cake\View (namespace), **65**
Cake\View\Helper (namespace), **66–69**
CDN, **158**
Client (clase en Cake\Network\Http), **131**
Controller (clase en Cake\Controller), **61**
CSRF, **158**

D

delete() (Cake\ORM\Table method), **76**
DRY, **158**
DSN, **158**

E

Entity (clase en Cake\ORM), **74**

F

FlashHelper (clase en Cake\View\Helper), **66**
Form (clase en Cake\Form), **109**
FormHelper (clase en Cake\View\Helper), **67**

H

HtmlHelper (clase en Cake\View\Helper), **67**

I

Inflector (clase en Cake\Utility), **133**

N

Notación de punto, **158**
Number (clase en Cake\I18n), **135**
NumberHelper (clase en Cake\View\Helper), **67**

P

PaaS, **158**
PaginatorHelper (clase en Cake\View\Helper), **67**

Q

Query (clase en Cake\ORM), **73**

R

Router (clase en Cake\Routing), **57**
routes.php, **158**
RssHelper (clase en Cake\View\Helper), **68**

S

Security (clase en Cake\Utility), **115**
SessionHelper (clase en Cake\View\Helper), **68**
Sintaxis de plugin, **158**

T

Table (clase en Cake\ORM), 74
Text (clase en Cake\Utility), 139
TextHelper (clase en Cake\View\Helper), 68
Time (clase en Cake\Utility), 141
TimeHelper (clase en Cake\View\Helper), 69
TimestampBehavior (clase en Cake\Model\Behavior), 79
TranslateBehavior (clase en Cake\Model\Behavior), 80
TreeBehavior (clase en Cake\Model\Behavior), 81

U

UrlHelper (clase en Cake\View\Helper), 69

V

View (clase en Cake\View), 65

X

Xml (clase en Cake\Utility), 143