LearnCakePHP Documentation

Release

eagle

Jun 18, 2017

Contents:

CHAPTER 1

Introduction

Installation

- Install XAMPP, change port to 81
- Web server is in c->xampp->htdocs which is the place where the program will run through localhost:81
- Download Composer.phar by using the following console command line in the htdocs directory

```
php -r "readfile('https://getcomposer.org/installer');" | php
php composer.phar create-project --prefer-dist cakephp/app myblog
```

- If Composer is installed globally in the machine by using composer.exe then use the following comand line in the htdocs directory

```
composer create-project --prefer-dist cakephp/app myblog
```

Note:

- This command will create a folder in htdocs called myblog with the required files to run a CakePHP project
- To run the app go to localhost:81/myblog

Naming Conventions

- **Files and class names**
  - Filenames are underscored (my_cake_class.php)
  - Class names are CamelCased (MyCaseClass)
  - Controller class names and files end with “controller”

- **Models & Databases**
  - Model class name are singular and CamelCased (User)
  - Table names corresponding to models are plural and underscored(users)
  - You can use inflector to check singular and plural words(category and categories)
  - Foreign keys are recognized by default as the singular name of the related table followed by _id (user_id)
  - Tables where models interact will require a singular primary key to uniquely identify each row
- **Controllers**
  - Controller class names are plural and CamelCased and ends with "controller" (UsersController)
  - Index() is the default method in a controller
  - yourapp.com/welcome will call the welcome controller and the index() method
- **Views**
  - Template files should be named after the controller functions that they display
  - The doThis() function of the UsersController class looks for a template in src/Template/Users/do_this.ctp
- **Example**
  - Database table: "articles"
  - Table class: ArticlesTable, found at src/Model/Table/ArticlesTable.php
  - Entity class: Article, found at src/Model/Entity/Article.php
  - Controller class: ArticlesController, found at src/Controller/ArticlesController.php
  - View template, found at src/Template/Articles/index.ctp

Routing

- Routing maps URLs to controller actions
- Routing helps URLs look much better and perform better in search engines
- Apache's mod_rewrite is not required for routing but makes things look neater
- Default route pattern: http://example.com/controller/action/param1/param2..

Helpers

- Helpers are component-like classes for the presentation layer of the application
- Presentation logic is shared between views or layouts
- To use a helper, add it to the controller's helpers array
- var $helpers = array('Form','Html','Javascript','Time');

Helpers List			
Ajax	Number	XML	Cache
Paginator	Form	RSS	HTML
Session	JS	Text	Javascript
Time			

Components

- Components are packages of logic shared between Controllers
- CakePHP comes with a core set of components that can be loaded into the controller at any time
- Components usually have some sort of configuration in the $components array of the controller's beforeFilter() method

Core Components		
Security	Authentication	Sessions
Request Handler	Access Control list	Emails
Cookies		

Plugins

- CakePHP allows you to combine controllers, models and views together to package as a "plugin" that others can use in their Cake applications
- Plugins are tied to an application only by configuration. Otherwise, plugins operate in their own space.
- Stored in /app/plugins

Scaffolding

- Scaffolding allows developers to generate a very basic application that can perform CRUD(create, read, update and delete) operations
- Scaffolding just needs a model and controller
- Scaffolding is great for development or just to get something up and running but using it too much will cause a loss of flexibility

CHAPTER 2

Connect to Database

- Go to localhost:81/phpmyadmin
- Create a new user with password and all privileges
- Create a Database called myblog
- Create one table called posts with 7 columns
- Insert some data into posts table
- In the project folder go to config->app.php and in approximately line 230 start changing user name with user from phpmyadmin with the corresponded password and change the name of database to myblog

CHAPTER 3

Routing

- Routing allows us to map URLs to specific controller actions
- Routes are defined in the config/routes.php
- There are 2 interfaces to define a rout:

1- Scoped Builder (Recommended)

```php
<?php
Router::scope('/users', function($routes)
{
    $routes->connect('/',['controller'=>'Users', 'action'=>'index']);
});
```

2- Static Method

```php
<?php
Router::connect('/users',['controller'=>'Users', 'action'=>'index']);
```

Simple Route

```php
<?php
$routes->connect('/users',['controller'=>'Users','action'=>'index']);
//this will execute the index() method inside of the Users controller
```

```php
<?php
$routes->connect('/users',['controller'=>'Users']);
//this will do the same thing because index() is the default action
```

Passing Parameters

```php
<?php
Router::connect(
'/users/:id',
['controller'=>'Users', 'action'=>'view'],
['id'=>'\d+','pass'=>['id']] //'\d+' is a regular expression that will validate that␣
↪the id is a digit
);
```

Wildcard

```php
<?php
$routes->connect(
    '/listings/*',
    ['controller'=>'Listings','action'=>'display']
);
```

Note:

- Use the * to specify a Wildcard
- The above Wildcard will take /listings/[anything] to the display() method of the Listings controller
- You can also use ** to capture the remainder of a URL as a single passed argument

Route Elements

- Specifying route elements allows you to define places in the URL where parameters for controller actions should lie

```php
<?php
$routes->connect(
'/:controller/:id',
['action'=>'view'],
['id'=>'[0-9]+']  //regular expression [0-9]
);
```

- This will allow you to view models from any controller with a URL like /controllername/:id

Special Route elements

Controller	Action	Plugin	Prefix
_ext	_base	_scheme	_host
_port	_full	_ssl	_method
name			

Named Elements

- You can use named routes to make calling them more convenient as well as boost performance a bit

```
<?php
$routes->connect('/login',
    ['controller'=>'Users','action'=>'login'],
    ['_name'=>'login]
);
//Generate a URL
$url = Router::url(['_name'=>'login]);
//with query strings
$url = Router::url(['_name'=>'login','username'=>'jimmy']);
```

Route Prefixes

- Specifying route elements allows you to define places in the URL where parameters for controller actions should lie

```
<?php
Router::prefix('admin',function($routes){
    $routes->connect('/create',['controller'=>'Posts','action'=>'create']);
});
```

- Now we can go to /admin/create to create a new Posts

Plugin Routes

- You can create special routes for plugins

```
<?php
Router::plugin('MyPlugin',function($routes){
    $routes->connect('/:controller');
});
```

- Routes connected above will automatically have the prefix of '/my_plugin'

Dashed Route

- Dashed Route can be used to format URLs to use dashes

```
<?php
Router::plugin('ToDo',['path'=>'to-do'],function($routes){
$routes->fallbacks('DashedRoute');
});
```

File Extensions

```
<?php
Router::extensions(['html','json']);
//this will allow you to use the .html or .json extensions in your routes
//you can also set extensions per scope
Router::scope('/api',function($routes){
    $routes->extensions(['json','xml']);
});
```

Restful Routes

You can easily generate RESTful routes with the Router

```
<?php
Router::scope('/',function($routes){
    $routes->extensions([json']);
    $routes->resources('Posts');
});
```

Nested resources

- Once resources are in a scope, you can connect sub-resource routes as well

```
<?php
Router::scope('/api',function($routes){
    $routes->resources('Posts',function($routes){
        $routes->resources('Comments');
    });
});
```

Note: /api/posts/:id/Comments /api/posts/:id/Comments/:id

CHAPTER 4

Controllers

- Controllers hold the actions that your routes are mapped to
- **Controllers are responsible for:**
  - The interpretation of any request data
  - Making sure the right models are called
  - Making sure the right response or view is rendered
- Tour models should be fat and your Controllers thin
- Doing this will help your code be more reusable and also makes it easier to test

Naming Conventions

- Files should end in “Controller” (UsersController.php)
- Classes should also end in Controller (UsersController)
- Named after the primary model (User)

AppController Class

- All controllers you create will extend the AppController Class
- Location is src/Controller/AppController.php
- You can use the “initializer()” method in your controller class to use a constructor
- AppController extends Cake/Controller/Controller

The Request Object

- The CakePHP router use connecting routes to find and create the controller instance
- The request data is stored in the request object
- Access using the $this->request property

Controller Setup

```
<?php
namespace App\Controller;
use App\Controller\AppController;
class BooksController extends AppController{
    public function index(){
        /Action code
    }
    public function view($id){
        //Action code
    }
    public function search($query){
        //Action code
    }
}
```

Passing View data

- Controller::set() is used to pass data from the controller to the view

```
$this->set('firstName','John');
$this->set('lastName','Doe');
```

- Because of CakePHP strict conventions, you co not need to render the view manually
- In View:

```
<?php
My name is <?=h($firstName)?> <?=h($lastName)?>
//h is a helper function
```

View Options

- You can set specific view options using viewBuilder()

```
<?php
$this->viewBuilder()
    ->helper(['MyHelper'])
    ->theme('SomeTheme')
    ->className
```

- You can use the correct naming conventions and the view will be rendered automatically but to do it manually, use:

```
<?php
$this->render()
//or specify a view
$this->render('someView')
```

Handling Redirects

- The most common way to redirect would be to use Controller::redirect()

```
<?php
return $this->redirect(
    ['controller'=>'Users','action'=>'register']
    );
```

- Relative and Absolute Paths:

```
<?php
return $this->redirect('/orders/thanks');
return $this->redirect('http://www.example.com');
```

- Go back to referrer:

```
<?php
return $this->redirect($this->referrer());
```

Actions and Model loading

- To redirect to another action on the same controller:

```
<?php
$this->setAction('index');
```

- Load a model table that is not the controller default

```
<?php
$this->loadModel('Articles');
$recentArticles = $this->Articles->find('all', [
    'limit'=>5,
    'order'=>'Articles.created DESC'
]);
```

Model Pagination

- Use the $paginate attribute to easily page result from the model

```
<?php
class PostController extends AppController
{
```

```
    public $paginate = [
        'Post'=>[
            'conditions'=>['published'=>1]
        ]
    ];
}
```

Loading Components & Helpers

- Define components in the controller's initialize() function:

```
<?php
public function initialize(){
    parent::initialize();
    $this->loadComponent('Csrf');
    $this->loadComponent('Comments', Configure::read('Comments'));
}
```

- Load Helpers:

```
<?php
public $helpers = ['Form','Html'];
```

Life-Cycle Callbacks

- beforeFilter(Event $event) is executed before every action in the controller. Useful for session and permission operations.
- beforeRender(Event $event) is executed after the action logic
- AfterFilter(Event $event) is executed after controller action and after render is complete

CHAPTER 5

Controller & Route Example

- Go to url localhost:81/myblog/posts and it will through an error asking to create PostsController class in src/Controller/PostsController.php with the following code:

```
<?php
namespace App\Controller;
use App\Controller\AppController;
class PostsController extends AppController
{

}
```

- Create PostsController.php file in src/Controller folder and but the above code in it
- Now if you go to the url it will ask for an index function inside PostsController class

```
<?php
namespace App\Controller;
use App\Controller\AppController;
class PostsController extends AppController
{
    public function index()
    {

    }
}
```

- Now CakePHP will ask for a view for the PostsController::index() in src/Template/Post/index.ctp but we will change the Route
- Go to config/routes.php to add a route

```
<?php
Router::scope('/posts', function(RouteBuilder $routes){
    $routes->connect('/',
        ['controller' => 'Posts']
```

```
    );
});
```

- To test that the route is working add the following code to the PostsController index action then go to url localhost:81/myblog/posts

```
<?php
die('This is the posts/index');
```

Note: die function will stop the execution of the program at this point and without it the program will proceed to need a view for the controller

- Note that you can use a constructor code to run a code as soon as AppController start

```
<?php
public function initialize(){
    die('This is the initialize constructor for AppController');
}
```

Note: initialize() function will override index() function

- **Now we will pass a parameter after post url and use it in the controller:**
  - In routes.php add the following code:

```
<?php
Router::scope('/posts/:id', function($routes){
    $routes->connect('/',
    ['controller'=>'Posts', 'action'=>'view'],
    ['id'=>'\d+', 'pass' => ['id']] //the \d+ is a regular expression for␣
↪validation
    );
});
```

  - In PostsController.php add the following code:

```
<?php
public function view($id){
    die('This is post '.$id);
}
```

- Note that you have to comment the initialize function that we have created before or it will override this function
- Now if you go to url localhost:81/myblog/post/1 it will show This is post 1
- The following code are in the same scope (posts) and should be modified:

```
<?php
Router::scope('/posts', function(RouteBuilder $routes){
$routes->connect('/',
    ['controller' => 'Posts']
    );
});

Router::scope('/posts/:id', function($routes){
```

```
    $routes->connect('/',
    ['controller'=>'Posts', 'action'=>'view'],
    ['id'=>'\d+', 'pass' => ['id']]
    );
});
```

- Modified to:

```
<?php
Router::scope('/posts', function(RouteBuilder $routes){
    $routes->connect('/',
        ['controller' => 'Posts']
    );
    $routes->connect('/:id',
    ['controller'=>'Posts', 'action'=>'view'],
    ['id'=>'\d+', 'pass' => ['id']]
    );
});
```

- **Add create and edit route to the post scope**
  - In routes.php add the following code

```
<?php

    $routes->connect('/create',
        ['controller'=>'Posts', 'action'=>'create']
    );
    $routes->connect('edit/:id',
        ['controller'=>'Posts', 'action'=>'edit'],
        ['id' => '\d+', 'pass' => ['id']]
    );
```

  - In PostsController.php add the following code

```
<?php
public function edit($id){
    die('editing post '.$id);
}
public function create(){
    die('create post');
}
```

- This will add url localhost:81/myblog/edit/<any number> and localhost:81/myblog/create

Query String

- **To get the a parameter from url like localhost:81/myblog/posts/hello?name=mike**
  - In PostsController.php add the following code:

```
<?php
public function hello(){
    die('hello '.$this->request->query['name']);
    //or
    //die('hello '.$this->request['url']['name']);
}
```

- And add the following code to routes.php inside posts scope

```
<?php
$routes->connect('/hello',
    ['controller'=>'Posts', 'action', => 'hello']
);
```

- Now if you go to url localhost:81/myblog/posts/hello?name=mike will get hello mike

- And for multiple parameter in a url like localhost:81/myblog/posts/hello?name=mike&age=34 use the following code for the hello() function

```
<?php
public function hello(){
    //print_r($this->request->query);
    //or for more info about parameters
    print_r($this->request->params);
    die();
}
```

- And to know if the request is a post or get use the following in the hello() function

```
<?php
if($this->request->is('post')){
    die('This is a POST');
} elseif($this->request->is('get')){
    die('This is a GET');
}
```

- And to get path information use the following in hello() function

```
<?php
//die($this->request->webroot); //will output: /myblog/
die($this->request->base); //will output: /myblog
die($this->request->hear); //will output: /myblog/posts/hello
die($this->request->header('User-Agent')); //will output: agent information
```

Router Prefixes

- Adding prefix for admin localhost:81/myblog/admin add the following code to routes.php

```
Router::prefix('admin', function($routes){
    $routes->connect('/', ['controller' => 'Dashboard']);
    $routes->connect('/create', ['controller' => 'Posts', 'action' => 'create']);
});
```

- Now if you go to localhost:81/myblog/admin it will ask for DashboardController.php which we will add to a new folder in src/Controller folder called Admin
- **Also create PostsController.php in the same folder**

- DashboardController.php

```
<?php
namespace App\Controller\Admin;  // note the Admin namespace point to the␣
↪Admin folder
```

```
use App\Controller\AppController;

class DashboardController extends AppController
{
    public function index(){
        die('Dashboard');
    }
}
```

- PostsController.php

```
<?php
namespace App\Controller\Admin;  // note the Admin namespace point to the␣
↪Admin folder
use App\Controller\AppController;

class PostsController extends AppController
{
    public function create(){
        die('Admin creating Post');
    }
}
```

- So now if you go to localhost:81/myblog/admin you will get Dashboard and if you go to localhost:81/myblog/admin/create you will get Admin Creating Post

---

Note: Note that if you change the route to scope instead of prefix you will get create post which is the result for PostsController.php that belong to the Controller folder not the one belongs to Admin folder and this is the diffident between scope and prefix where prefix will look for a controller within a sub-namespace which in this case is the Admin folder

---

CHAPTER 6

Views & Templates

What is a View?

- Views are responsible for generating and displaying the output of an application
- Everything that the user sees is output from a view/view template

View Forms

- Views can be displayed as different forms with different extensions
- some of the most common types are:

HTML	XML
JSON	RSS

Anatomy of the view layer

- Views - Templates that hold the presentation logic of an application
- Elements - Small, reusable bits of code rendered inside views
- Layouts - Template wrapper code. Most views are rendered inside of a layout
- Helpers - Classes that encapsulate view logic that can be used in many areas of the application like date formatter or currency formatter

AppView

- All application have a default AppView class which is located in src/View/AppView.php

- AppView can be used to load helpers that can be used in all of an application's views. Use "initialize()" for this

View Templates

- By default, templates are in PHP and use a special file extension called ".ctp"(CakePHP Template)
- These files contain representational logic and are stored in "src/Template"
- Templates should be named after the controller action that it corresponds to (eg. src/Template/Posts/view.ctp)

View Variables

- Variables can be set in a controller and passed on to a view using "set()"

```
<?php
$this->set('name','John');
```

- You can then use the "name" variable in the view

```
<?php
<p>My name is <?=name?></p>
```

- Variable data can easily be escaped using "h()"

```
<?php
<p>My name is <?=h(name)?></p> // this is for security
```

Extending Views

- Views can be wrapped in another by using view extending. This allows us to follow DRY(Don't Repeat Yourself) principles

```
<?php
<h1><?=$this->fetch('title')?></h1>
<?=$this->fetch('content')?>

<div> class="actions">
    <h3>User Actions</h3>
    <?=$this->fetch('sidebar')?>
</div>
//title, content and sidebar are files in another folder
```

View Blocks

- Using view blocks, we can define blocks in our views that will be defined somewhere else
- This is ideal for things like sidebars, footers, etc

```
<?php
$this->start('sidebar');
echo $this->element('sidebar/recent_posts');
echo $this->element('sidebar/recent_comments');
$this->end();
```

Layouts

- Layouts are templates that wrap another template and contain code that can be displayed on all pages/view
- The default layout is at src/Templates/Layouts/default.ctp

```
<?php
<?=$this->fetch('content')?>
//displays the content from the current rendered view
```

- set the title with

```
<?php
<?=$this->assign('title','Welcome');?>
```

Elements

- Elements are small bits of view code that can be re-used from page to page
- This would be things like ad boxes, navigation, sliders, etc
- Can be considered a mini-view

```
<?php
echo $this->element('navbar');
```

- Can add a second param to pass values

Custom View Classes

- You can add custom classes for new types of data view
- Classes go in "src/View"
- Should be suffixed with View (eg. pdfView)
- When referencing a view class, do not use "View"

```
<?php
$builder->viewClass('pdf');
```

Themes

- Themes are plugins that focus on providing template files. They are used to switch the look of the page quickly and easily

- Set the theme name in the beforeRender() callback in the controller

```
<?php
class PostController extends AppController{
    public function beforeRender(Event $event){
        $this->viewBuilder()->theme('Modern');
    }
}
```

JSON & XML Views (Data Views)

- JsonView & XmlView allow us to create JSON & XML responses
- You must enable RequestHandlerComponent

```
<?php
$this->loadComponent('RequestHandler');
```

- You can generate these either by using the _serialize key of by just creating normal template files

CHAPTER 7

Views Example

- First change the homepage to be post instead of default by changing the route of default home page in routes.php

```
<?php
Router::scope('/', function ($routes){
    $routes->connect('/', ['controller'=> 'Posts', 'action' => 'index', 'home']);
}
```

- Create a new folder in src/Template directory called Posts and inside it a new file called index.ctp
- In PostsController.php make sure that index do not contain die function or the program will stop executing before reaching Posts view
- Add the following code to index

```
<div class = "row">
    <div class="columns large-3 medium-4">
        <h3>sidebar</h3>
    </div>
    <div class="columns large-9 medium-8">
        <h1>Main Content</h1>
    </div>
</div>
```

- **Now we will remove Documentation and API links and replace it with Home link by going to src/Template/Layout/default**
  - Remove the following code:

```
<li><a target="_blank" href="http://book.cakephp.org/3.0/">Documentation</a></
↪li>
<li><a target="_blank" href="http://api.cakephp.org/3.0/">API</a></li>
```

  - And modify top-bar-section class as follows:

```
<div class="top-bar-section">
    <ul class="right">
```

```
        <li><?php echo $this->Html->link('Home','/');?></li>
    </ul>
</div>
```

Note: You can specify the link with this code

```
<li><a target="Home" href="http://localhost:81/myblog/">Home</li>
```

– **Now we will get a value from the controller to the view**

* In PostsController.php add the following code to index function

```
<?php
$this->set('person', 'John');
```

* In index.ctp in Template/Posts folder add the following code in one of the columns

```
<?php
<h1><?= $person </h1>
```

* This is another example in PostsController.php add the following code to index function

```
<?php
$people = ['Mike', 'Paul', 'Jeff', 'Michelle'];
$this->set('people', $people);
```

* In index.php add the following code to one of the div

```
<?php
 <ul>
    <?php foreach($people as $person){ ?>
        <li><?= $person ?></li>
    <?php } ?>
</ul>
```

* Another example to get values from a multidimensional array, add the following code to PostsController.php

```
<?php
    $posts = [
        ['id' => 1, 'title' => 'First Post', 'body' => 'This is my␣
↪first post'],
        ['id' => 2, 'title' => 'second Post', 'body' => 'This is my␣
↪second post'],
        ['id' => 3, 'title' => 'third Post', 'body' => 'This is my␣
↪third post'],
    ];
    $this->set('posts', $posts);
```

* In index.ctp add the following code

```
<?php foreach($posts as $post) { ?>
    <div>
        <h4><?= $post['title'] ?></h4>
        <p><?= $post['body'] ?></p>
    </div>
```

```
    <hr>
<?php } ?>
```

* An another example we will work with the view method in PostsController.php so first we have to create a new file in Template/Post folder called view.ctp
* Then add the following code to view method in PostsController.php

```
<?php
$post = [
    'id' => $id,
    'title' => 'First Post',
    'body' => 'This is my first post'
];
$this->set('post', $post);
```

Note: We have previously made a route for this functionality in routes.php posts scope

```
<?php
$routes->connect('/:id',
    ['controller'=>'Posts', 'action'=>'view'],
    ['id'=>'\d+', 'pass' => ['id']]
);
```

* And in view.ctp add the following code

```
<div class = "row">
    <div class="columns large-3 medium-4">
        <h3>sidebar</h3>
    </div>
    <div class="columns large-9 medium-8">
        <h1><?= $post['title'] ?></h1>
        <p><?= $post['body'] ?></p>
    </div>
</div>
```

* Now if you go to localhost:81/myblog/posts/1 you will get the results

Note: in index.ctp you can make a link for each of the id by using the following code

```
<h4><?= $this->Html->link($post['title'], '/posts/'.$post['id']) ?></
↪h4>
```

CHAPTER 8

Component and Helpers

Component

- Add a new file in controller folder Called DevsController.php with the following code

```
<?php
namespace App\Controller;
use App\Controller\AppController;

class DevsController extends AppController
{
    public function index()
    {

    }
}
```

- In Template folder add a new folder called Devs and inside create a new file called index.ctp

```
<?php
echo 'Devs';
//go to localhost:81/myblog/devs to see the result
```

- Create BlogComponent.php file in Controller/Component folder with the following code:

```
<?php
namespace App\Controller\Component;
use Cake\Controller\Component;

class BlogComponent extends Component
{
    public function sayHello()
    {
        return 'Hello';
```

```
    }
}
```

- **In PostController.php add the following code:**
  - Create initialize() function

```
<?php
public function initialize()
{
    $this->loadComponent('Blog');
}
```

  - In index() function add the following code

```
<?php
die($this->Blog->sayHello());
```

- Now if you go to url localhost:81/myblog/posts will return Hello

---

Note: Because we have created a route from default page to PostController we will get the same result by going to url localhost:81/myblog

---

- Now remove die() function from PostsController() and add the following code to BlogComponent.php

```
<?php
public function getPosts()
{
    $posts = [
                ['id' => 1, 'title' => 'First Post', 'body' => 'This is my first post
↪'],
                ['id' => 2, 'title' => 'second Post', 'body' => 'This is my second␣
↪post'],
                ['id' => 3, 'title' => 'third Post', 'body' => 'This is my third post
↪'],
            ];
    return $posts;
}
```

- Change PostController.php to be as follows:

```
<?php
private $posts;
public function initialize()
{
    $this->loadComponent('Blog');
    $this->posts = $this->Blog->getPosts();
}
```

- Add the following code to DevsController.php

```
<?php
private $posts;
public function initialize()
{
    $this->loadComponent('Blog');
    $this->posts = $this->Blog->getPosts();
```

```
}
public function index()
{
    $this->set('posts', $this->posts);
}
```

- Add a new component file in component directory called DevsComponent.php with the following code:

```
<?php
namespace App\Controller\Component;

use Cake\Controller\Component;

class DevsComponent extends Component
{
    public function generatePassword()
    {
        $password = '';
        $desired_length = rand(8,12);

        for ($length = 0; $length < $desired_length; $length++)
        {
            $password .= chr(rand(32,126));
        }
        return $password;
    }
}
```

- And Add the following to DevsController.php

```
<?php
  namespace App\Controller;
  use App\Controller\AppController;

  class DevsController extends AppController
  {
      public function initialize()
      {
          $this->loadComponent('Devs');
      }
      public function index()
      {
          $this->set('password', $this->Devs->generatePassword());
      }
  }
```

- Then add the following to Template/devs/index.ctp

```
<?php
<h4>Password: <?= $password ?></h4>
```

Helpers

- You can find Html helpers ($this->Html) examples in Template/Layout/default.ctp
- Also in default.ctp you can add custom style attribute by adding the following code in the <head> tag

```
<?PHP
<style>
    body{
        <?= $this->Html->style([
        'background' => '#000'
        ]); ?>
    }
</style>
// to generate a link use
$this->Html->link('link text', 'link location');
// to add attribute to Html helpers use the following
$this->Html->link('link text', 'link location', ['class' => 'active']);
```

- **To add an image using html helper use the following code:**
  - in PostController.php modify view function as follows:

```
<?php
$post = [
    'id' => $id,
    'title' => 'First Post',
    'body' => 'This is my first post',
    //added image link
    'image' => 'http://oi63.tinypic.com/21blk5g.jpg'
];
$this->set('post', $post);
```

  - in Posts/view.ctp adde the folowing to view the image

```
<?= $this->Html->image($post['image'], ['alt' => 'myImage']); ?>
```

- There are multiple helpers available like media

Nested list

- **To use nested list use the following code**
  - In PostsController.php add the following to the view function

```
<?php
$languages = [
    'Languages' => [
        'English' => [
            'American',
            'Canadian',
            'British'
        ],
        'Spanish',
        'German'
    ]
];
$this->set('languages', $languages);
```

  - In Posts/view.ctp add the following to view the list

```
<?php
<?= $this->Html->nestedList($languages); ?>
```

Html Table Helper

- Use the following code to create a table using Html table helper

```
<?php
<table>
    <?= $this->Html->tableHeaders(
        ['Id', 'Name', 'Email']
    ); ?>
    <?= $this->Html->tableCells([
        ['1', 'John', 'j@j.com'],
        ['2', 'Bob', 'b@b.com']
    ]); ?>
</table>
```

Html Form Helper

- In Template/Layout/default.ctp add the following ling in <div class="top-bar-section>

```
<?php
<li><?= $this->Html->link('Create Post', '/posts/create'); ?> </li>
```

- Create a new Template Posts folder called create.ctp with the following code:

```
<h3>Create Post</h3>
<?= $this->Form->create(); ?>
    <?= $this->Form->input('title', array(
                                'label' => 'Post Title',
                                'class' => 'class-name'
                        )); ?>
    <?= $this->Form->input('body', array(
                                'label' => 'Post Body',
                                'type' => 'textarea',
                                'escape' => false,
                                'class' => 'class-name'
                        )); ?>
    <?= $this->Form->input('category', array(
                                'label' => 'Categoty',
                                'type' => 'select',
                                'empty' => 'Select One',
                                'options' => ['Web Development', 'Design', 'Marketing
↪'],
                                'class' => 'class-name'
                        )); ?>
    <?= $this->Form->input('author', array(
                                'label' => 'Author',
                                'class' => 'class-name'
                        )); ?>
    <?= $this->Form->input('time', [
        'type' => 'time',
```

```
        'interval' => 15
    ]); ?>
    <?= $this->Form->inputs([
        'name' => ['label' => 'Name', 'class' => 'class-name'],
        'age' => ['label' => 'Age', 'class' => 'class-name']
    ]); ?>
    <hr>
    <?= $this->Form->submit('Submit', array('class' => 'class-name')); ?>
<?= $this->Form->end(); ?>
```

Text Helper

```
<?= $this->Text->truncate($post['body'],196,['ellipsis' => '...', 'exact' => false]);␣
↪?>
```

CHAPTER 9

Models & Baking

- In src/Model/Table folder create a new file called PostsTable.php with the following code:

```
<?php
namespace App\Model\Table;
use Cake\ORM\Table;

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

- In PostsController.php remove the code from initialize and remove private $posts function and modify index function as follows:

```
<?php
public function index(){
    $posts = $this->Posts->find('all');
    $this->set(compact('posts'));
}
```

- **Now to get posts of certain $id change view function in PostsController to have the following code**
  - In PostsController.php

```
<?php
public function view($id){
    $posts = $this->Posts->find('all');
    $post = $this->Posts->get($id);
    $this->set(compact('posts', 'post'));
}
```

  - In View.ctp

```
<div class = "row">
    <div class="columns large-3 medium-4">
        <h3>sidebar</h3>
    </div>
    <div class="columns large-9 medium-8">
        <h1><?= $post['title'] ?></h1>
        <p><?= $post['body'] ?></p>
    </div>
</div>
```

- To sort the results so the newest posts will be on top modify the index function as follows:

```
<?php
public function index(){
    $posts = $this->Posts->find('all', array('order' => array('created' => 'desc')));
    $this->set(compact('posts'));
}
```

- **To create a new post do the following:**
  - PostsController.php add the following code:

```
<?php
public function initialize(){
    parent::initialize();
        $this->loadComponent('Flash');
    }
    public function create(){
        $post = $this->Posts->newEntity();

    if($this->request->is('post')){
        $post = $this->Posts->patchEntity($post, $this->request->data);
        if($this->Posts->save($post)){
            $this->Flash->success(__('Post Created'));
            return $this->redirect(['action' => 'index']);
        }
        $this->Flash->error(__('Unable to save post'));
    }
    $this->set('post', $post);
}
```

- To add timestamp to each post add the following to Template/Posts/index.ctp after the title tag:

```
<?php
<small><strong><?= $post['created']->format(DATE_RFC850); ?></strong></small>
```

Validation

- To Add validation add the following code in the src/Model/PostsTable.php

```
<?php
namespace App\Model\Table;
use Cake\ORM\Table;
use Cake\Validation\Validator;

class PostsTable extends Table{
```

```
    public function initialize(array $config){
        $this->addBehavior('Timestamp');
    }
    public function validationDefault(Validator $validator){
        $validator
            ->notEmpty('title')
            ->RequirePresence('title')
            ->notEmpty('body')
            ->RequirePresence('body')
            ->notEmpty('author')
            ->RequirePresence('author')
            ->notEmpty('category')
            ->RequirePresence('category');
        return $validator;
    }
}
```

Update

- To update a post add the following code to PostsController.php

```
<?php
public function edit($id){
    //die('create post');
    //$posts = $this->Posts->find('all');
    //$this->set(compact('posts'));

    $post = $this->Posts->get($id);

    if($this->request->is(['post', 'put'])){
        $this->Posts->patchEntity($post, $this->request->data);
        if($this->Posts->save($post)){
            $this->Flash->success(__('Post Updated'));
            return $this->redirect(['action' => 'index']);
        }
        $this->Flash->error(__('Unable to update post'));
    }
    $this->set('post', $post);
}
```

- Create a new file under Layout/Posts called edit.ctp with the following code which is almost same as create.ctp

```
<?php
<h3>Edit Post</h3>
<?= $this->Form->create($Post); ?>
    <?= $this->Form->input('title', array(
                                'label' => 'Post Title',
                                'class' => 'class-name'
                        )); ?>
    <?= $this->Form->input('body', array(
                                'label' => 'Post Body',
                                'type' => 'textarea',
                                'escape' => false,
                                'class' => 'class-name'
                        )); ?>
```

```
    <?= $this->Form->input('category', array(
                                'label' => 'Categoty',
                                'type' => 'select',
                                'empty' => 'Select One',
                                'options' => ['Web Development', 'Design', 'Marketing
↪'],
                                'class' => 'class-name'
                        )); ?>
    <?= $this->Form->input('author', array(
                                'label' => 'Author',
                                'class' => 'class-name'
                        )); ?>
    <?= $this->Form->input('time', [
        'type' => 'time',
        'interval' => 15
    ]); ?>
    <?= $this->Form->inputs([
        'name' => ['label' => 'Name', 'class' => 'class-name'],
        'age' => ['label' => 'Age', 'class' => 'class-name']
    ]); ?>
    <hr>
    <?= $this->Form->submit('Submit', array('class' => 'class-name')); ?>
<?= $this->Form->end(); ?>
```

- Now edit Layout/Posts/view.ctp to have the following code:

```
<?php
<div class = "row">
<div class="columns large-3 medium-4">
    <h3>sidebar</h3>
</div>
<div class="columns large-9 medium-8">
   <h1><?= $post['title'] ?></h1>
   <p><?= $post['body'] ?></p>
</div>

<?= $this->Html->link('Edit Post', ['action' => 'edit', $post['id']], ['class' =>
↪'class-name']); ?>
```

Delete

- In PostsController.php add the following code:

```
<?php
public function delete($id){
    $this->request->allowMethod(['post', 'delete']);
    $post = $this->Posts->get($id);
    if($this->Posts->delete($post)){
        $this->Flash->success(__('Post Deleted'));
        return $this->redirect(['action' => 'index']);
    }
}
```

- Now in Layout/Posts/view.ctp add the following code:

```
<?= $this->Form->postLink('delete',
                          ['action' => 'delete', $post['id']],
                          ['confirm' => 'Are you sure?','class' => 'className']); ?>
```

CHAPTER 10

Indices and tables

- genindex
- modindex
- search