**CakePHP**

**CakePHP at a glance**

**Conventions**

* Để cho cấu trúc nhất quán, không cần những cấu hình ko cần thiết

**Model Layer**

* Nhận data và chuyển đổi nó
* Bao gồm: lưu trữ, xác nhận, xử lý, …
* Dung class name đúng convention sẽ không cần phải định nghĩa, cakePHP sẽ tự hiểu model ứng với bảng nào?

**View Layer**

**Controller Layer**

* Xử lý request từ user, trả về response
* Người quản lý, đảm bảo những tài nguyên cần thiết cho những thằng khác (model, view) hoàn thành công việc

**Request Cycle**

* User gửi yêu cầu đến page của server
* Gồm 10 bước, nhưng chưa hiểu

**Controller Conventions**

* CamelCase, kết thúc bằng Controller
* Route users/view: view() của UsersController

**URL – Controller**

* /some-word/haha-hihi: ứng với SomeWordController::hahaHihi()

**File – Class Name:**

* Giống nhau

**Model – Database**

* Model: CamelCase
* Model: ArticleCategories -> table: article\_categories
* Tên cột: giống tên bảng
* Nếu mún dung những từ cho những qui tắc, dung Cake\Utility\Reflector
* Foreign key: thêm \_id vào sau tên bảng, như bảng users -> user\_id, article\_categories -> article\_category\_id
* Join table: dung dấu “\_” và xếp theo abc
* Khi lưu 1 record, cakePHP sẽ lưu thêm 1 mã UUID duy nhất (36 ký tự) cho record đó, dùng như khóa chính, bảng vẫn có khóa chính nhé. Cake\Utility\Text::uuid()

**View**

* 1 action viewAll() của ArticlesController sẽ tự động tìm view có dạng src/Template/Articles/view\_all.php

**Convention Example**

* Table: articles
* Table class: src/Model/Table/ArticlesTable.php
* Entity class: src/Model/Entity/Article.php
* Controller class: src/Controller/ArticlesController.php
* View template: src/Template/Articles/index.ctp
* Khi request về thanhbui.com/articles, nó sẽ tự động gọi action index() của ArticlesController, Model sẽ tự động tạo và liên hệ với db

**Folder Structure**

* Bin: chứa console thực thi
* Config: chứa file config, chi tiết về kết nối db
* Plugins: chứa plugins
* Logs: chứa log của mình
* Src: nơi đặt ứng dụng của mình
* Tests: nơi chứa test case
* Tmp: chứa thông tin tạm, thường là mô tả model, thông tin session
* Webroot: nơi bắt đầu ứng dụng, chứa các file có thể truy xuất từ người dùng

**Quick Start Guide**

* Model Layer:
* Những phương thức làm việc với collection of entities đc đặt vào class Table
* Những đặc tính liên quan đến 1 record đc đặt trong class Entity
* VD: password hashing liên quan đến từng record nên thuộc về entity object
* Tạo hàm setter/mutator: vào src\Model\Entity, vào model của bảng, thêm hàm protected bắt đầu \_set + Columnname. Đây là convention của cakePHP
* Với route: somePath/\*: \* là tham số đưa vào, trong controller, truy xuất nó bằng $this->request->params[‘pass’]: qui ước
* Finder methods: trong model, trả về object chứa data lấy từ db
* Html, Text helper: $this->Html, $this->Text
* H(): encode html output
* Dùng set() trong controller để gửi biến vào view
* Authenticaton: đc xử lý bởi Components, Components là những code trong controller để dùng cho 1 loại việc j đó. Thêm AuthComponent vào AppController để dùng cho các method
* Dùng Auth->allow để cho phép truy cập vào page nào đó mà ko cần auth
* Authorization adapter: xác định quyền truy cập, xem của từng user đối với từng page khi đã login vào

**Installation**

**Configuration**

* Cake có 1 file cấu hình mặc định, có thể thêm file và tải nó vào config/bootstrap.php

**Routing**

* 2 cách để khai báo route:
  + Static method
  + Scoped route builder: dùng cho nhiều route, xử lý tốt hơn
* /\*: truyền nhiều tham số, cakePHP sẽ tự động tách các tham số bằng dấu ‘/’. Trong controller dùng func\_get\_args
* Kiểm tra regex của url: nếu có $routes->fallbacks thì ko có tác dụng
* Đặt tên cho đường route. Sau này có tạo link thì dùng cái này cho nhanh

Router::connect(

‘abc’,

[controller, action, …],

[‘\_name’ => ‘hahaha’]

);

* /\*\*: chỉ lấy 1 tham số, cho dù chứa dấu ‘/’, ko tách các tham số bằng dấu ‘/’
* Route Element:

$routes->connect(

‘/:controller/:id’,

[‘action’ => ‘view’],

[‘id’ => ‘[0-9]+’]

);

* + Với code trên, khi ta request với url: apples/15, nó sẽ đến ApplesController, view method, truyền vào tham số 15
  + :controller: là 1 route element mặc định của cakePHP
* Giấu controller name, trong trường hợp chỉ có 1 controller. VD: /SomeController/view => /view

$routes->connect(

‘/:action’,

[‘controller’ => ‘SomeController’]

);

* Insensitive URL

$routes->connect(

‘/:userShortcut’,

[‘controller’ => ‘Ahih’, ‘action’ => ‘do ngu’],

[‘userShortcut’ => ‘?i:pricipal’]

);

* 1 số route element có sẵn của cakePHP: controller, action, plugin, …
* \_namePrefix: prefix cho name của route
* Prefix routing: prefix chính là thư mục con, sub-namespace của thư mục – namespace Controller.

VD: với đường route admin/users/edit/5, nếu dùng Route::prefix cho ‘admin’ nó sẽ tìm trong thư mục Admin (trong thư mục Controller), tìm UsersController, tìm view method.

Còn nếu ko dùng prefix, nó sẽ tìm UsersController trong thư mục Controller như bt.

Tương tự cho view.

Ko muốn dùng prefix: [‘prefix’ => false]

* Plugin Routing
* SEO-Friendly routing
* Routing File Extensions

**Passed Arguments**

* Func\_get\_args()
* $this->request->params
* $this->passedArgs
* Khi dùng routing array [‘controller’ => ‘Haha’, ‘action’ => ‘hehe’, 5, ‘huhu’], những value nào có key là số sẽ là các arguments

**Generating URLs – Reverse Routing**

* Thay đổi cấu trúc URL mà ko cần phải chỉnh sửa code nhiều

**Redirect routing**

* HTTP status 30x
* $routes->redirect();

**Custom Route Classes**

* Route Class phải đc tìm thấy trong namespace Routing\\Route
* Route Class nên mở rộng Cake\Routing\Route
* Route nên thực hiện ít nhất match() hoặc parse()
* [‘routeClass’ => …]
* Default route class
  + Router::defaultRouteClass()
* Fallbacks method: viết tắt của Router::defaultRouteClass()

**Creating persistent URL parameters**

* Filter function: thực thi trc khi request URL đc so sánh với các route

Router::addUrlFilter

**Handling Named Parameters in URLs**

* DispatcherFactory

**Blog Tutorial**

* $routes->fallbacks: nếu như request 1 url ko có trong route, nó sẽ tự động tìm trong controller
* $this->set(compact(‘a’)) == $this->set(‘a’, $a)
* $this->Html->link(str, [‘action’ => ‘sth’]): controller trong trường hợp này là controller dẫn đến view chứa lệnh này
* $this->Model->get(id): lấy ra 1 record, nếu ko tồn tại thì throw exception
* Success(), error() của Flash: tạo 1 biến session chứ message, ở view dùng $this->Flash->render
* $this->Model->newEntity()
* $this->Model->patchEntity()
* $this->redirect([‘action’ => ‘findlove’]): chuyển tới url có action là findlove nằm trong cùng controller
* $this->Form->input(name, [‘row’ => 3]): tạo textarea
* $this->Html->link(‘Edit’, [‘action’ => ‘edit’, 3]) 🡪 someCon/edit/3
* $this->Form->create($article):
  + Nếu tham số đưa vào là ko có, ví dụ $article trc đó = $this->Model->newEntity() thì form sẽ tự tạo ra đường link là “Controller/add”
  + Nếu tham số đưa vào là có, ví dụ $article trc đó là record, thì nó tự tạo ra đường link là “Controller/edit/id”, id tự động lấy từ $article và tự động điền các giá trị của $article vào các input
* $this->request->allowMethod([‘post’, ‘delete’]): nếu request ko thuộc method cho phép thì sẽ quăng exception
* $this->Form->postLink: tạo ra link loại post (thưc ra là có cái form ẩn) có nhúng javascript trong đó
* Bake thêm behavior Tree cho các bảng nào các cột lft và rght
* $this->Model1->Model2->find(‘treeList’): khi đưa biến này vào view, nó sẽ tự động tìm input có name giống, và chuyển cái input đó thành select box với các option có từ biến này, biến này là câu query
* $this->Form->input(name, [‘options’ => […]]): ra select box
* $this->Auth->redirectUrl(): trả về url sau khi login?
* $this->Auth->logout(): trả về url sau khi logout?
* loginRedirect: khi vào trang login trực tiếp, khi login xong nó sẽ đi tới url của loginRedirect, còn nếu vô 1 trang # mà nó bắt mình nhảy qua trang login, thì sau khi login xong nó sẽ trở lại trang mình vô lúc đầu. Mặc định là ‘/’
* $this->Auth->identify(), $this->Auth->setUser
* $this->Auth->user(‘id’)
* $this->request->action
* Trong loadComponent Auth:
  + ‘authorize’ => ‘Controller’: khi có option này, khi vào controller, nó sẽ vào method isAuthorized trc, nếu hàm này trả về true thì là có quyền đi tiếp, nếu false thì ko. Hàm này sẽ nhận 1 tham số là $user, là record của user đăng nhập hiện tại
* Khi dùng component Auth, nó sẽ mặc định hash password của mình khi mình login

**Request and Response Objects**

**Request**

* Request là request object mặc định. Cứ mỗi request đc gửi đến thì obj Request đc tạo ra và đc truyền vào các lớp của ứng dụng: Controller, Cells, View, Helper, Components (tham chiếu qua Controller)
* **Request parameter**:
  + Truy xuất tham số route (route element):

$this->request->params[‘controller’]

$this->request->param(‘controller’)

* + Passed arguments

$this->request->pass

$this->request[‘pass’]

$this->request->params[‘pass’]

* 1 số parameter hữu ích: plugin, controller, action, prefix
* **Query string parameters**
  + URL: demo/example/?page=123

$this->request->query(‘page’)

Trả về null nếu tồn tại key này

* **Request body data**: dùng cho data dạng post như submit form, trả về null nếu kho tồn tại
  + $this->request->data(‘title’)

$this->request->data[‘title’][0][‘haha’]

**Controller**

* Mỗi request đưa lên, cakePHP sẽ tìm và tạo ra object controller tương ứng
* Khi action của controller kết thúc, nó sẽ tự động dùng view mặc định, nếu ko mún thì mình có thể tự return response
* Controller::set có thể nhận parameter là associative array với các key là các biến đưa vào view
* Dùng viewBuilder để chỉnh sửa, bổ sung helper, theme, view class, …
* cakePHP sẽ tự động gọi $this->render() ở cuối mỗi controller, nếu ko mún điều này, set $this->autoRender = false
* render view khác: $this->render(‘/Ahaha/Ahihi’)
* render view trong plugin: (‘PluginName.PluginController/customView’)
* $this->redirect(url, Http status)
* $this->redirect($this->referrer()): trở lại trang trc
* $this->setAction(act): chuyển qua action khác cùng controller, url vẫn giữ nguyên nhé
* $this->loadModel(model): dùng model khác với mặc định của Controller
* Request Life-cycle Callbacks: cakePHP cung cấp 1 số callback/event để xử lý logic đối với 1 request
  + Event: Initialize, startup, beforeRidirect, beforeRender, shutdown
  + Callback: gắn với event?
    - beforeFilter: gọi trong initialize event, gọi trc mỗi action của controller, dùng check permission,…. Gọi khi missing action
    - beforeRender: gọi sau mỗi action
    - afterFilter: sự kiện shutdown, method chạy sau cùng
* **Components**: gói logic chia sẻ cho các controller. Ngoài các component cakePHP cung cấp, ta có thể tạo ra component
  + **Authentication**
* CakePHP cung cấp 1 số cách để xác thực: FormAuthenticate, BasicAuthenticate, DigestAuthenticate. Mặc định AuthComponent dùng FormAuthenticate
* $this->Auth->config: config cho Auth
  + $this->Auth->config(‘authenticate’, [

‘Basic’ => [‘userModel’ => ‘Members’],

‘Form => [‘userModel’ => ‘Members’],

]);

* + Thiết lập model cho FormAuthenticate và BasicAuthenticate là bảng members. Mặc định là bảng users.
  + Để ko lặp lại những cái giống nhau cho Form và Basic, dùng AuthComponent::ALL
  + Các thiết lập:
    - Fields: chỉ ra các field để xác thực, mặc định là username và password
    - userModel: bảng để xác thực, mặc định là bảng users
    - finder: method để copy 1 record, mặc định là all
    - passwordHasher: mặc định là Default
    - scope, contain, userField: cũ, dùng finder

**Error & Exception Handling**

* Mặc định CakePHP dùng class ErrorHandler hoặc ConsoleErrorHandler để bắt và log error
* Default error handler cũng xử lý những uncaught exception
* Các option của error handling:
  + errorLevel: chỉ ra level của error cần bắt
  + trace: ghi ra log file
  + exceptionRenderer: class hiển thị uncaught exceptions, đặt trong src/Error
  + log
  + skipLog: mảng các Exception ko cần log
  + extraFatalErrorMemory: số MB cần tang khi gặp fatal error để xử lý, ghi log
* fatal error handler sẽ đc gọi độc lập với cấu hình debug và errorLevel, nhưng kết quả sẽ khác nhau dựa trên cấu hình debug (1, 0)
* **Tạo ra error handler**
* exceptionRenderer: render exception theo ý của mình, có thể dựa vào loại lỗi, có thể cung cấp hiển thị, logic xử lý theo ý mún ứng với từng loại?
* Tạo ra controller để xử lý exception: mặc định là App\Controller\ErrorController (nếu có)
* Mở rộng BaseErrorHandler/ErrorHandler:
  + Tạo class error extends từ BaseErrorHandler/ErrorHandler
  + Viết lại \_displayException/\_displayError
  + dùng echo chứ không dùng return
  + Có thể dùng redirect: $this->controller->redirect([])\
  + Đăng ký lại trong bootstrap.php (new Class())->register()
* ExceptionRenderer:
  + Muốn thay đổi layout, ko cho hiển thị ra, bật login:
    - Tắt debug, bật log
    - Hoặc sửa lại view error
  + Mỗi loại exception sẽ hiển thị view khác nhau: tạo hàm public, trong đó echo ra hoặc redirect
  + Dùng setAction sẽ ko thay đổi url?
  + Sửa lại view cho tất cả: viết lại hàm render:

public function render()

{

parent::render();

return $this->\_outputMessage('error123');

// return $this->controller->render('error123');

}

* + Có thể bắt error trong đây lun: dùng public function error/typeError/…, nếu dùng class error (extends từ BaseErrorHandler) thì sẽ ko vô đây?
  + Để render 1 view mà ko cần redirect: dùng $this->controller->render(‘\Error\error123’); lệnh này sẽ trả lại view ở src/Template/Error/error123