REPUBLIC OF THE PHILIPPINES

DEPARTMENT OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

RUFINO JOHN E. AGUILAR

https://rufdev.github.io/

Module 1 : Intro to CodeIgniter

- Model View Controller (MVC)
- Local Environment Setup (Composer)
- Folder Structure
- Controllers
- Routes
- CodeIgniter Libraries and Helpers

Model View Controller (MVC)

- CodeIgniter is a popular PHP framework that follows the MVC (Model-View-Controller) architectural pattern. The MVC pattern is a software design pattern that separates an application into three interconnected components to promote better code organization, maintainability, and reusability

MVC

MODEL

Data and business logic

Interacting with database

Processing data

Enforcing business rules

VIEW

Presentation and user interface

Display data

Capture user input

Separate from application logic

CONTROLLER

Intermediary between model and view

Receive user request

Process request

Communicate with the model to retrieve data and send to the view

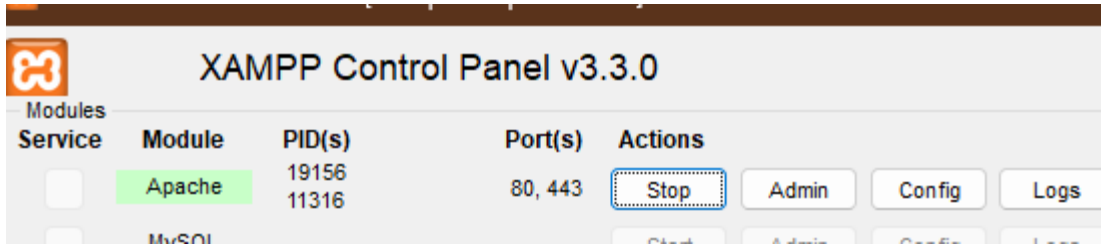
Handling HTTP request and routing

MVC Sample flow

- A user makes an HTTP request to a specific URL handled by a controller.
- The controller receives the request, processes it, and interacts with the Model to retrieve the necessary data from the database.
- The retrieved data is then passed to a View for rendering.
- The View combines the data with an HTML template and generates the final web page.
- The web page is sent back as an HTTP response to the user's browser.

Local Environment Setup (Composer)

- Install XAMPP
  - https://www.apachefriends.org/
- Install Composer
  - https://getcomposer.org/



A Dependency Manager for PHP

```
C:\Users\RufinoJohn>php -v
PHP 8.2.0 (cli) (built: Dec  6 2022 15:31:23) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.0, Copyright (c) Zend Technologies
```

REPUBLIC OF THE PHILIPPINES
DEPARTMENT OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

Local Environment Setup (Composer)

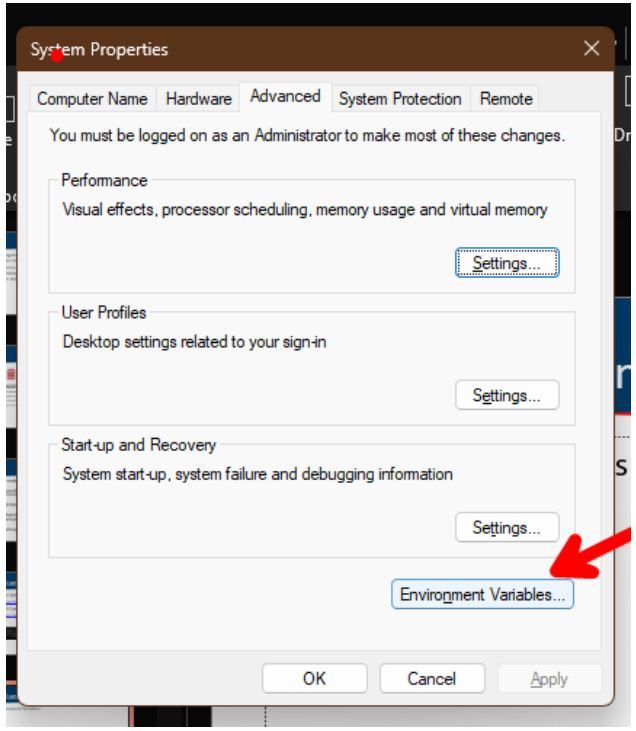
- Php.ini
  - extension=intl
- Httpd.conf
  - Rewrite _module modules/mod_rewrite.so

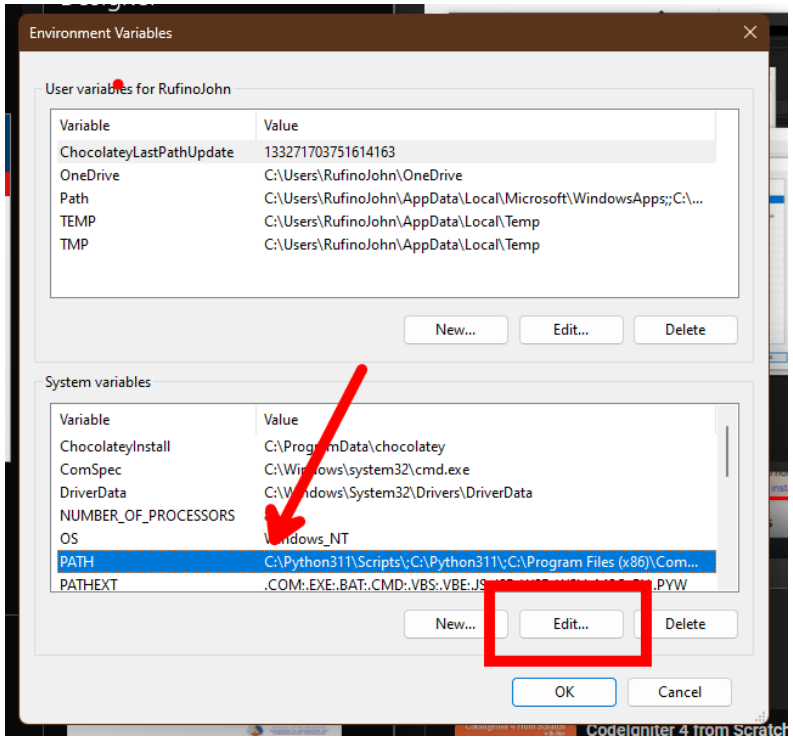
```
;extension=gmp
extension=intl
;extension=imap
;extension=ldap
```

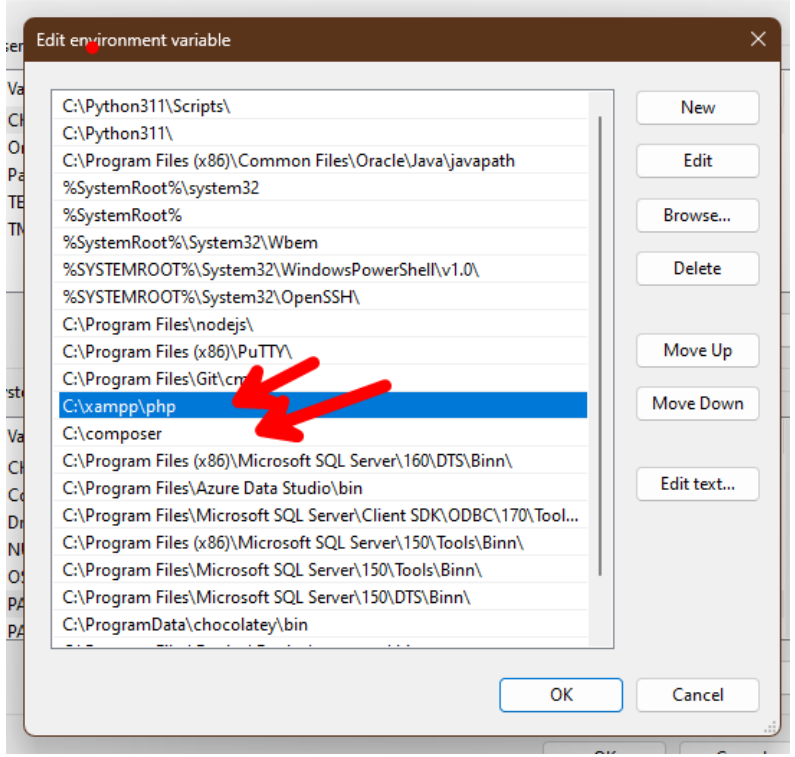
```
#LoadModule reqtimeout_module modules/mod_reqtimeou
LoadModule rewrite_module modules/mod_rewrite.so
#LoadModule sed_module modules/mod_sed.so
#LoadModule session_module modules/mod_session.so
```

Local Environment Setup (Composer)

- Environment Variables







Local Environment Setup (Composer)

- Installation

Installation

In the folder above your project root:

```
composer create-project codeigniter4/appstarter project-root
```

```
RufinoJohn@PICTD-RUFY MINGW64 /c/xampp/htdocs
$ composer create-project codeigniter4/appstarter ciapp
```

Local Environment Setup (Composer)

- Download and navigate to project directory

```
MINGW64:/c/xampp/htdocs
  - Installing sebastian/environment (5.1.5): Extracting archive
  - Installing sebastian/diff (4.0.5): Extracting archive
  - Installing sebastian/comparator (4.0.8): Extracting archive
  - Installing sebastian/code-unit (1.0.8): Extracting archive
  - Installing sebastian/cli-parser (1.0.1): Extracting archive
  - Installing phpunit/php-timer (5.0.3): Extracting archive
  - Installing phpunit/php-text-template (2.0.4): Extracting archive
  - Installing phpunit/php-invoker (3.1.1): Extracting archive
  - Installing phpunit/php-file-iterator (3.0.6): Extracting archive
  - Installing theseer/tokenizer (1.2.1): Extracting archive
  - Installing nikic/php-parser (v4.17.1): Extracting archive
  - Installing sebastian/lines-of-code (1.0.3): Extracting archive
  - Installing sebastian/complexity (2.0.2): Extracting archive
  - Installing sebastian/code-unit-reverse-lookup (2.0.3): Extracting archive
  - Installing phpunit/php-code-coverage (9.2.29): Extracting archive
  - Installing phar-io/version (3.2.1): Extractin
  - Installing phar-io/manifest (2.0.3): Extracti
  - Installing myclabs/deep-copy (1.11.1): Extrac
  - Installing doctrine/instantiator (2.0.0): Ext
  - Installing phpunit/phpunit (9.6.13): Extracti
18 package suggestions were added by new dependen
Generating optimized autoload files
27 packages you are using are looking for funding
Use the `composer fund` command to find out more!
No security vulnerability advisories found

RufinoJohn@PICTD-RUFY MINGW64 /c/xampp/htdocs
$ |
```

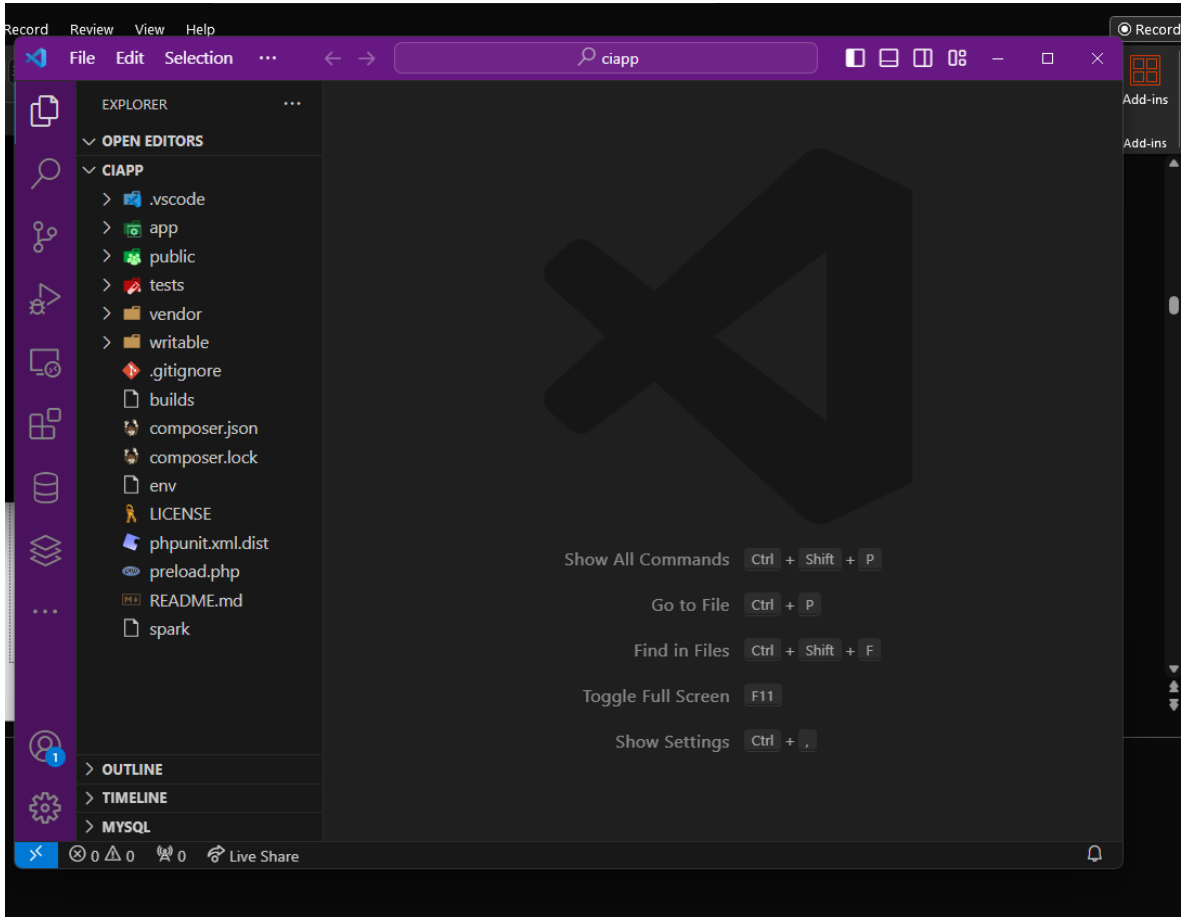
```
MINGW64:/c/xampp/htdocs/ciapp

RufinoJohn@PICTD-RUFY MINGW64 /c/xampp/htdocs
$ cd ciapp

RufinoJohn@PICTD-RUFY MINGW64 /c/xampp/htdocs/ciapp
$ |
```

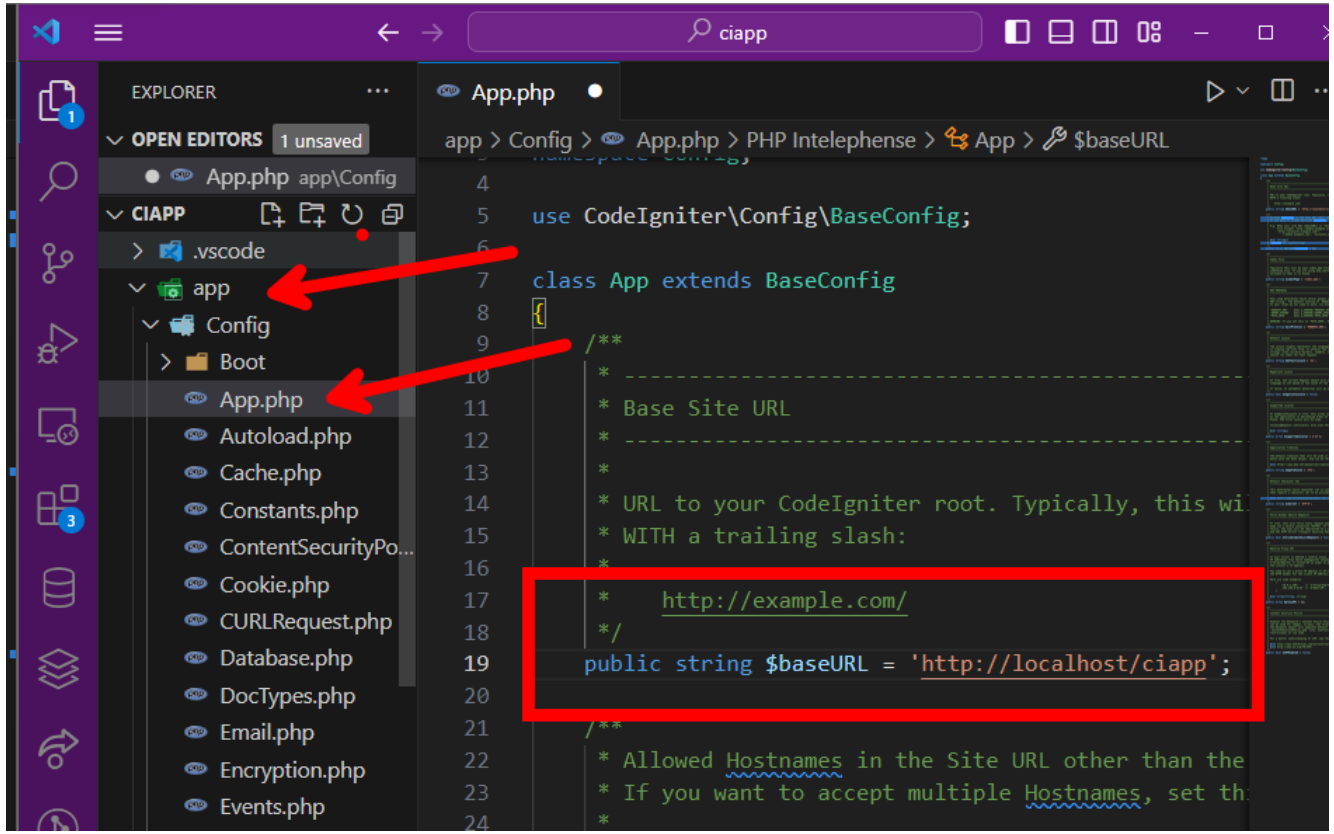
Local Environment Setup (Composer)

- code .
  - Open project on VS Code



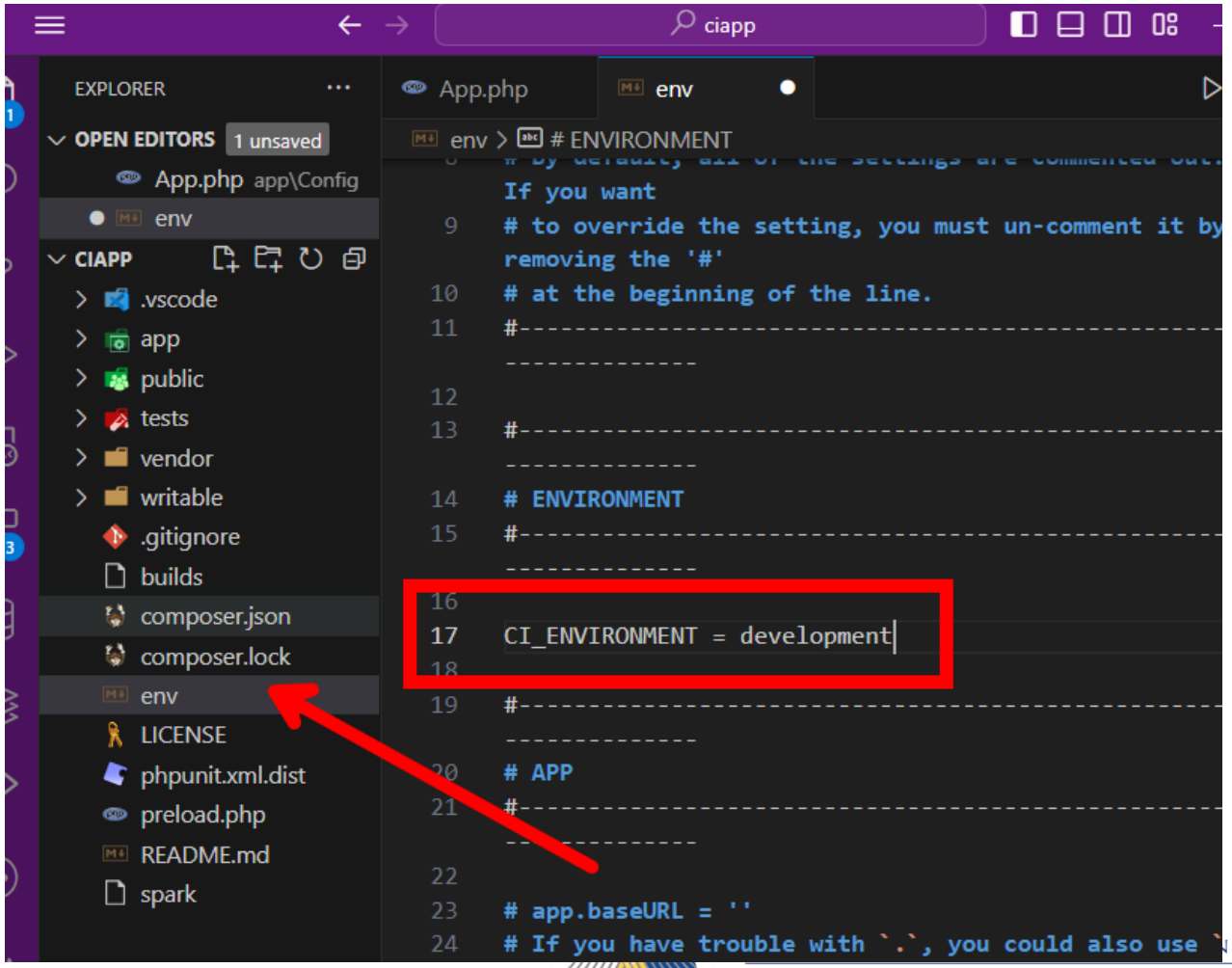
Local Environment Setup (Composer)

- Change baseURL variable



Local Environment Setup (Composer)

- CI_ENVIRONMENT
  - Development



Local Environment Setup (Composer)

- Apache2
  - Enable mod_rewrite

Enabling mod_rewrite

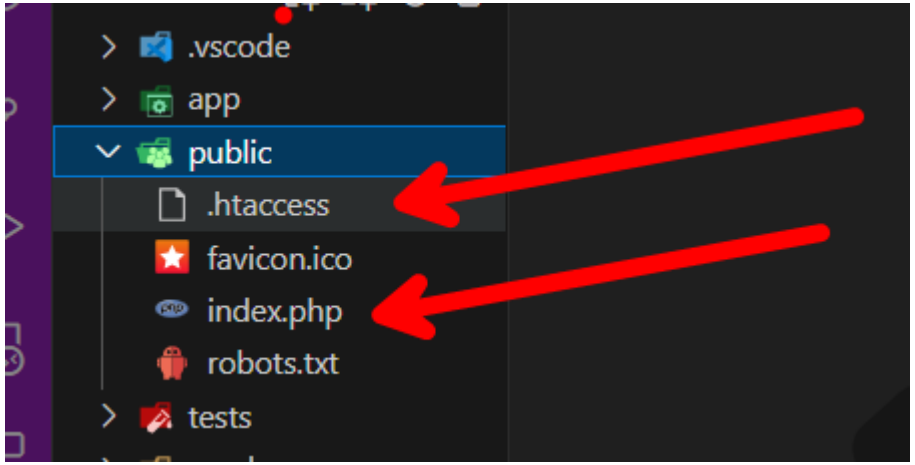
The "mod_rewrite" module enables URLs without "index.php" in them, and is assumed in our user guide.

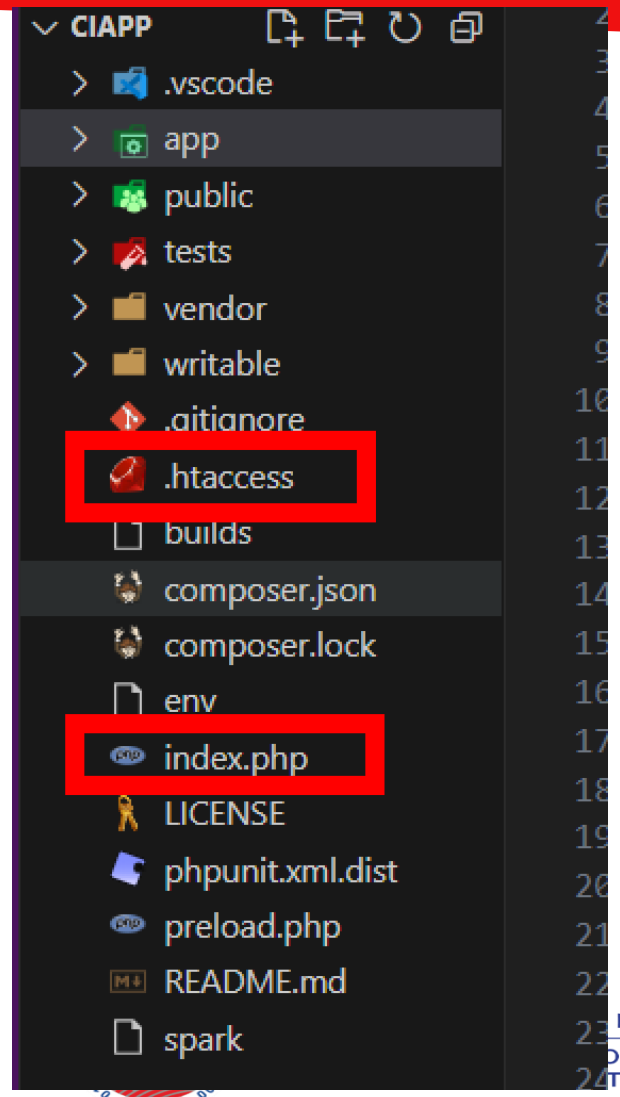
Make sure that the rewrite module is enabled (uncommented) in the main configuration file, e.g., **apache2/conf/httpd.conf**:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

Local Environment Setup (Composer)

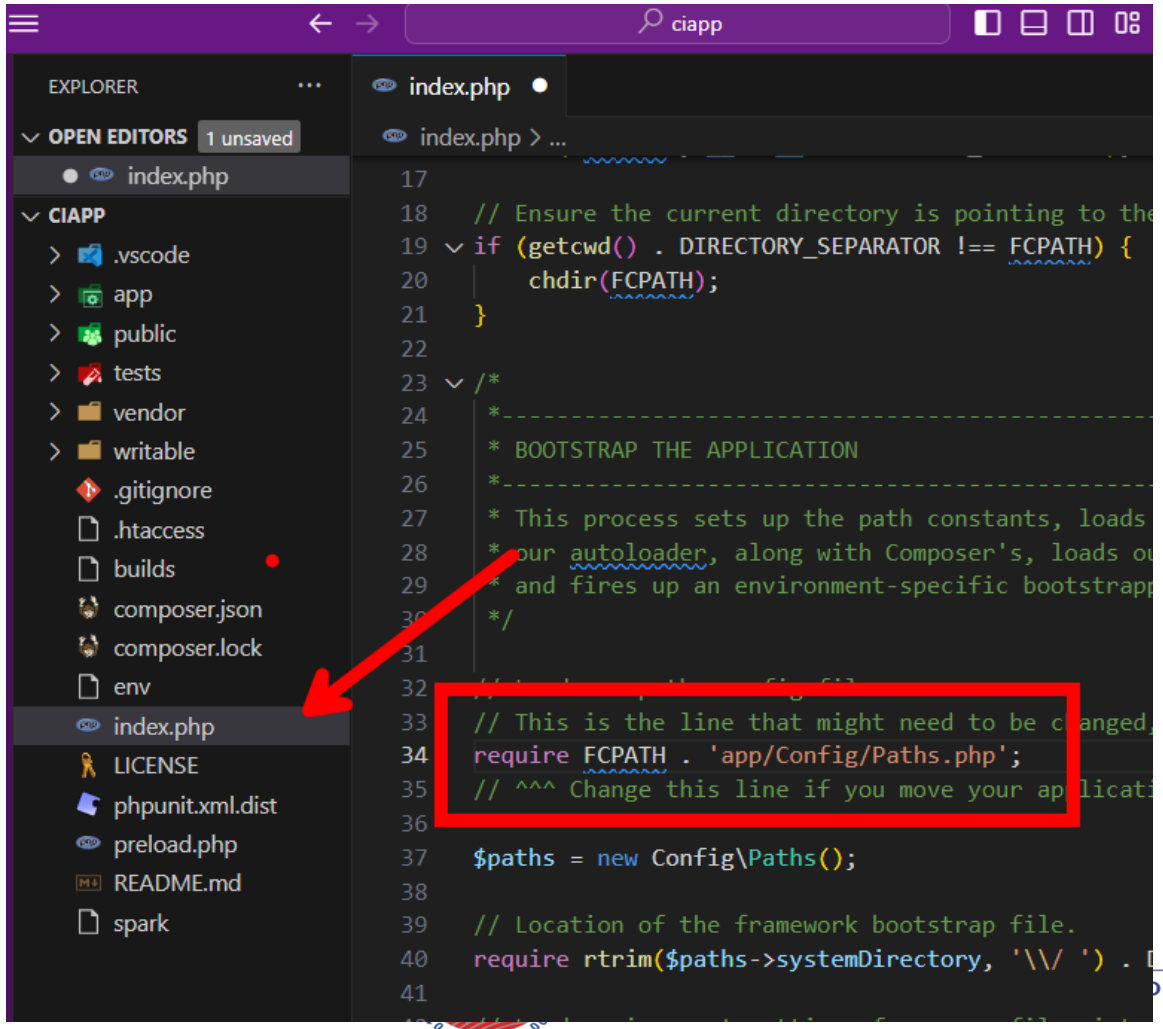
- Move .htaccess and index.php
  - To root directory





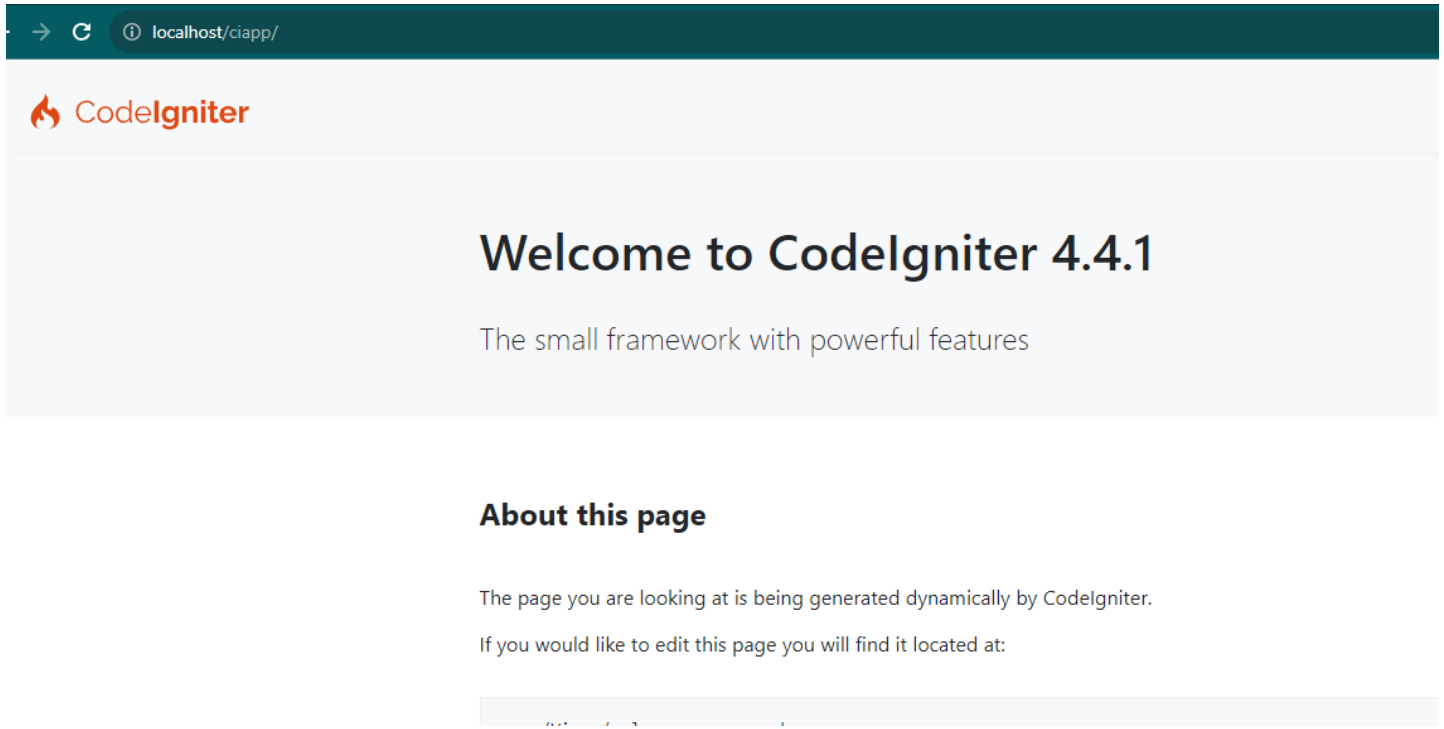
Local Environment Setup (Composer)

- Index.php
  - Change ../app to app

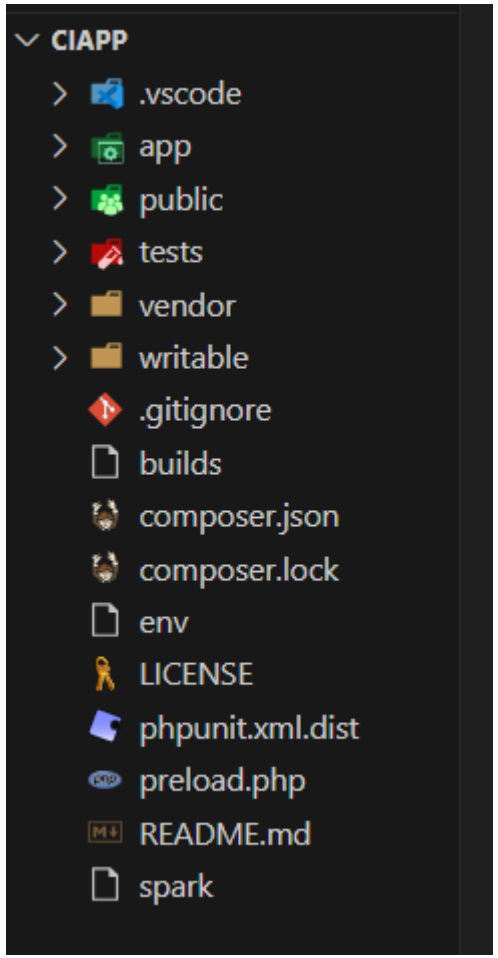


Local Environment Setup (Composer)

- CodeIgniter 4 is ready

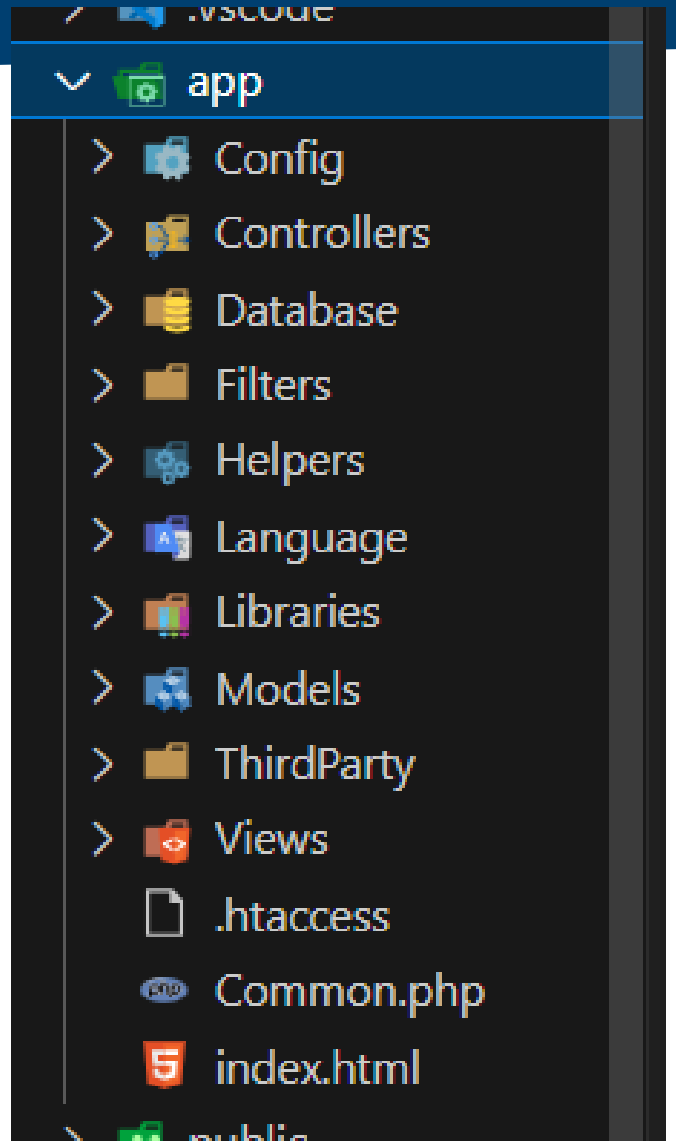


Folder Structure



- index.php loads all config files (initialize)
- .env file system variables (ex. Database)
- vendors folder are dependencies/libraries
- test (ex. Unit Testing)
- public – template/css/dependencies/assets
- writable
  - Log files
  - Debugging

Folder Structure



- app folder
  - Config – Configuration files
  - Controllers
  - Database – migrations and seeding
  - Filter – authentication/middleware/guards
  - Models – Classes/models
  - Views – UI/templates/partial views

Controllers

- is a fundamental part of the application that handles and processes incoming HTTP requests and acts as an intermediary between the Model (for data manipulation) and the View (for rendering the response)
- Receiving Requests
- Request Processing
- Routing
- Data Retrieval
- Response Preparation
- View loading
- User input handling
- Redirect and responses

Controllers

- extends BaseController

app > Controllers > Supportticket.php > PHP Intelephense > Supportticket

```php
namespace App\Controllers;

use App\Controllers\BaseController;
use CodeIgniter\HTTP\Response;

You, 2 months ago | 1 author (You)
class Supportticket extends BaseController
{
    public function index()
    {…
    }

    public function getall()
    {…
    }

    public function getbyid($id)
    {…
    }


    public function create()
    {…
    }

    public function update($id)
    {…
    }
      You, 2 months ago • added source code and git ignore
    public function delete($id)
    {…
    }
}
```

Controllers

- Constructor

Constructor

The CodeIgniter's Controller has a special constructor `initController()`. It will be called by the framework after PHP's constructor `__construct()` execution.

Controllers

- Included Properties

Request Object

The application's main Request Instance is always available as a class property, `$this->request`.

Response Object

The application's main Response Instance is always available as a class property, `$this->response`.

Logger Object

An instance of the Logger class is available as a class property, `$this->logger`.

Controllers

- Validating Data
  - Rules for General Use

$this->validate()

To simplify data checking, the controller also provides the convenience method `validate()`. The method accepts an array of rules in the first parameter, and in the optional second parameter, an array of custom error messages to display if the items are not valid. Internally, this uses the controller's `$this->request` instance to get the data to be validated.

```php
public function updateUser(int $userID)
{
    if (! $this->validate([
        'email' => "required|is_unique[users.email,id,{$userID}]",
        'name'  => 'required|alpha_numeric_spaces',
    ])) {
        // The validation failed.
        return view('users/update', [
            'errors' => $this->validator->getErrors(),
        ]);
    }

    // The validation was successful.

    // Get the validated data.
    $validData = $this->validator->getValidated();

    // ...
}
```

Routes

- routes are used to map URLs (Uniform Resource Locators) to specific controller methods. They define how incoming HTTP requests should be handled by the application. Routes help you create user-friendly URLs and determine which controller methods should respond to different URLs.
- Mapping URLs to Controllers/Methods
- Route Configuration
- Default Route
- Wildcards and Parameters
- HTTP Verbs
- Regular Expressions

Routes

```
40  //office api
41  $routes->group("api", ["filter" => "groupfilter:superadmin"], function ($routes) {
42      $routes->get("office", "Office::index");
43      $routes->post("office/list", "Office::getall");
44      $routes->get("office/(:num)", "Office::getbyid/$1");
45      $routes->post("office", "Office::create");
46      $routes->put("office/(:num)", "Office::update/$1");
47      $routes->delete("office/(:num)", "Office::delete/$1");
48
49  });
50
```

Routes

When you specify a route, you choose a method to corresponding to HTTP verbs (request method). If you expect a GET request, you use the `get()` method:

```
<?php

$routes->get('/', 'Home::index');
```

Routes

When you specify a route, you choose a method to corresponding to HTTP verbs (request method). If you expect a GET request, you use the `get()` method:

```
<?php

$routes->get('/', 'Home::index');
```

Routes

```php
<?php

// Calls $Users->list()
$routes->get('users', 'Users::list');

// Calls $Users->list(1, 23)
$routes->get('users/1/23', 'Users::list/1/23');
```

Routes

A URL containing the word **journals** in the first segment will be mapped to the `\App\Controllers\Blogs` class, and the default method, which is usually `index()`:

```php
<?php

$routes->get('journals', 'Blogs');
```

Routes

A URL containing the segments **blog/joe** will be mapped to the `\App\Controllers\Blogs` class and the `users()` method. The ID will be set to `34`:

```php
<?php

$routes->get('blog/joe', 'Blogs::users/34');
```

Routes

HTTP verb Routes

You can use any standard HTTP verb (GET, POST, PUT, DELETE, OPTIONS, etc):

```php
<?php

$routes->post('products', 'Product::feature');
$routes->put('products/1', 'Product::feature');
$routes->delete('products/1', 'Product::feature');
```

Routes

Controller's Namespace

When you specify a controller and method name as a string, if a controller is written without a leading `\`, the Default Namespace will be prepended:

```php
<?php

// Routes to \App\Controllers\Api\Users::update()
$routes->post('api/users', 'Api\Users::update');
```

Routes

View Routes

New in version 4.3.0.

If you just want to render a view out that has no logic associated with it, you can use the `view()` method. This is always treated as GET request. This method accepts the name of the view to load as the second parameter.

```php
<?php

// Displays the view in /app/Views/pages/about.php
$routes->view('about', 'pages/about');
```

Routes

Alias Filter

You specify an alias defined in **app/Config/Filters.php** for the filter value:

```
<?php

$routes->get('admin', ' AdminController::index', ['filter' => 'admin-auth']);
```

Routes

Grouping Routes

You can group your routes under a common name with the `group()` method. The group name becomes a segment that appears prior to the routes defined inside of the group. This allows you to reduce the typing needed to build out an extensive set of routes that all share the opening string, like when building an admin area:

```php
<?php

$routes->group('admin', static function ($routes) {
    $routes->get('users', 'Admin\Users::index');
    $routes->get('blog', 'Admin\Blog::index');
});
```

Routes

Nesting Groups

It is possible to nest groups within groups for finer organization if you need it:

```php
<?php

$routes->group('admin', static function ($routes) {
    $routes->group('users', static function ($routes) {
        $routes->get('list', 'Admin\Users::list');
    });
});
```

Routes

Confirming Routes

CodeIgniter has the following command to display all routes.

spark routes

Displays all routes and filters:

```
php spark routes
```

The output is like the following:

```
+---------+---------+---------------+-------------------------------+----------------+----------------+
| Method  | Route   | Name          | Handler                       | Before Filters | After Filters  |
+---------+---------+---------------+-------------------------------+----------------+----------------+
| GET     | /       | »             | \App\Controllers\Home::index  |                | toolbar        |
| GET     | feed    | »             | (Closure)                     |                | toolbar        |
+---------+---------+---------------+-------------------------------+----------------+----------------+
```

CodeIgniter Libraries and Helpers

- CodeIgniter (CI) is equipped with a set of libraries and helpers that provide various functionalities to simplify web development. These libraries and helpers can save you time and effort by offering pre-built functions for common tasks.

CodeIgniter Libraries and Helpers

- **Database Library**: This library simplifies database operations and provides a convenient and secure way to interact with databases.
- **Session Library**: Used for managing user sessions and handling session data.
- **Form Validation Library**: Helps validate and sanitize form data, making it easier to ensure data integrity.
- **Email Library**: Provides features for sending emails via various protocols, including SMTP and sendmail.
- **Image Manipulation Library**: Enables image processing, such as resizing, cropping, and rotating images.
- **Upload Library**: Makes it easy to handle file uploads and provides various configuration options for uploaded files.
- **Pagination Library**: Allows you to create paginated result sets for long lists of data.
- **Template Parser Library**: Provides a simple template system to separate logic from presentation in views.
- **Security Library**: Offers tools for improving application security, including data validation and handling XSS (Cross-Site Scripting) and CSRF (Cross-Site Request Forgery) protection.
- **Caching Library**: Helps improve application performance by caching parts of the output.

- **URL Helper**: Contains functions for creating links, working with URLs, and routing.
- **Form Helper**: Simplifies form creation and handling by generating form elements.
- **Text Helper**: Includes functions for working with text, such as formatting and word processing.
- **Date Helper**: Provides date and time manipulation functions.
- **File Helper**: Offers functions for file-related operations, like reading, writing, and deleting files.
- **HTML Helper**: Assists with generating HTML elements and tags.
- **Date Helper:** Offers functions for working with dates and times, including date formatting.
- **Cookie Helper**: Allows you to set, retrieve, and delete cookies.