

CRT Framework

HTTP framework or Request/Response framework for PHP

- **Requirement**

- PHP >= 7.4
- PDO PHP Extension
- Composer
- Apache, nginx or PHP's built-in web server

- **Installation**

Pertama-tama install terlebih dahulu composer pada mesin. Setelah composer terinstal, lalu install/update dependencies yang diperlukan oleh CRT Framework dengan menjalankan script dibawah ini pada console/command prompt mesin di dalam directory project CRT Framework:

> composer install

Untuk menjalankan CRT Framework pada browser, akses url project jika project berada pada root direktori mesin atau akses url lokal direktori project jika project berada di local, seperti contoh dibawah ini:

<http://domain.com/> => Jika project berada di root web servis mesin

<http://localhost/crt-framework/web/front.php> => Jika project berada di localhost

Untuk konfigurasi selanjutnya, yaitu konfigurasi virtual host mesin. Tahapan pembuatan virtual host mesin yaitu, buat virtual host baru pada konfigurasi vhost web server (baik pada web server apache maupun nginx), arahkan root directory pada settingan vhost ke file "web/front.php" (contoh: "C:/xampp/htdocs/crt-framework/web/front.php") dan atur name server nya. Jika vhost telah dibuat maka restart web server nya.

Setelah itu tambahkan name server dari vhost yang sudah dibuat ke file hosts mesin, untuk sistem operasi windows, file hosts berada di "C:\Windows\System32\drivers\etc\hosts".

- **Directory Structure**

```
example.com
|---- config
|---- src
|      |---- app
|      |      |---- Pegawai
|      |      |---- Controller
```

```

|         |         |         |         |---- PegawaiController.php
|         |         |         |---- Model
|         |         |         |---- Pegawai.php
|         |---- core
|         |         |---- Framework.php
|         |         |---- GlobalFunc.php
|         |---- pages
|         |         |---- welcome.php
|         |---- routes.php
|---- vendor
|         |---- autoload.php
|---- web
|         |---- assets
|         |         |---- css
|         |         |---- js
|         |---- front.php
|---- composer.json
|---- composer.lock

```

- **Directory structure explanation**

- **config**
Direktori ini berisi semua file konfigurasi aplikasi
- **src**
Direktori ini berisi kumpulan kode utama dari aplikasi mu, direktori ini juga memiliki beberapa subdirektori diantaranya: **app, assets, core, pages, routes.php**.
 - **app**
Berisi kode utama aplikasi seperti controller dan model
 - **core**
Direktori ini berisi kode inti dari framework
 - **pages**
Berisi kumpulan template/view aplikasi
 - **routes.php**
File routes.php ini berisi kumpulan route yang tersedia pada aplikasi
- **vendor**
Direktori ini berisi dependencies composer yang dibutuhkan aplikasi
- **web**

Direktori web berisi file controller utama aplikasi yang dimana menjadi gerbang utama aplikasi dan juga folder asset

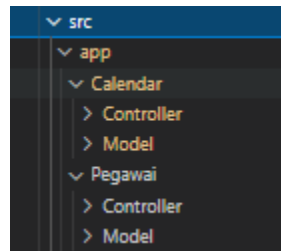
- **assets**

Berisi kumpulan asset aplikasi seperti file css, js, dan image

- **Rules**

1. Setiap model dan controller di pisahkan berdasarkan modul di dalam folder “src/app/”.

Contoh :



2. Penamaan model, controller, dan folder modul nya menggunakan Camel Case. Contoh : NamaModul
3. Untuk semua proses pengolahan data dilakukan di controller, model hanya berisi inisialisasi fungsi. Semua data yang dikirim ke template harus bersifat “final”, yang berarti tidak boleh ada pengolahan data di template

- **How to use**

Di bagian ini akan dijelaskan cara pemakaian dasar CRT Framework dari awal mula instalasi.

Dibawah ini merupakan contoh step - step pembuatan fitur dasar aplikasi pada framework ini:

1. **Menampilkan halaman dari template**

Pertama-tama buat terlebih dahulu route di file routes.php yang berada di folder “src/” seperti contoh dibawah ini:

```
$routes->add('welcome', new Routing\Route('/', [
    '_controller' => function(Request $request) {
        global $app;

        return $app->render_template('welcome', $request);
    },
]));
```

Dari kode diatas diketahui bahwa route yang kita buat dengan nama “welcome” dan url “/” menampilkan halaman dengan nama file template “welcome.php” yang berada di

folder “src/pages/”, untuk nama template yang akan dipanggil, tidak menggunakan ekstensi file .php, tetapi hanya menggunakan nama file nya.

2. Menampilkan halaman melalui controller

Pertama-tama buat terlebih dahulu route di file routes.php yang berada di folder “src/” seperti contoh dibawah ini:

```
$routes->add('hello', new Routing\Route('/hellos/get', [
    '_controller' => 'App\Calendar\Controller\TestingController::index'
]));
```

Dari kode diatas diketahui bahwa route yang dibuat dengan nama “hello” dan url “/hellos/get” menggunakan controller “TestingController” dengan method “index”. Untuk pemanggilan controller pada route harus disertakan dengan namespace class controller itu sendiri.

Direktori controller berada di “src/app>NamaModul/Controller>NamaController.php”. Lalu pada bagian “TestingController” kita panggil template yang akan ditampilkan pada method “index”

```
namespace App\Calendar\Controller;

use App\Calendar\Model\LeapYear;
use Core\GlobalFunc;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

class TestingController extends GlobalFunc
{
    public function index(Request $request)
    {
        return $this->render_template('welcome', $request);
    }
}
```

3. Menampilkan data dari controller ke template

Direktori controller berada di “src/app>NamaModul/Controller>NamaController.php”. Untuk menampilkan data dari controller ke template, kita hanya harus menambahkan data yang akan ditampilkan di fungsi render_template seperti dibawah ini:

```
namespace App\Calendar\Controller;

use App\Calendar\Model\LeapYear;
use Core\GlobalFunc;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
```

```

class TestingController extends GlobalFunc
{
    public function index(Request $request)
    {
        $nama_pegawai = "Deden";
        $perusahaan = "PT. SEJAHTERA TERUS";

        return $this->
render_template('welcome', $request, ['nama_pegawai' => $nama_pegawai, 'perusahaan' => $perusahaan]
    );

    }
}

```

Kita bisa menentukan sendiri nama variable dari data yang akan dikirim. Pada bagian template, kita langsung tampilkan data dari controller tadi

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    Nama : <?= $nama_pegawai ?>
    Perusahaan : <?= $perusahaan ?>
</body>
</html>

```

4. Menampilkan data dari url ke template

Kita asumsikan url yang dipanggil adalah “/page/1”, pertama kita buat terlebih dahulu route di file routes.php yang berada di folder “src/” seperti contoh dibawah ini:

```

$routes->add('detail', new Routing\Route('/page/{id}', [
    '_controller' => 'App\Calendar\Controller\TestingController::detail ',
]));

```

Pada “TestingController” kita ambil terlebih dahulu data yang dikirim di url tadi sebelum dikirim ke template

```

namespace App\Calendar\Controller;

use App\Calendar\Model\LeapYear;

```

```

use Core\GlobalFunc;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

class TestingController extends GlobalFunc
{
    public function detail(Request $request)
    {
        $id = $request->attributes->get('id');

        return $this->render_template('pegawai/detail', ['id' => $id]);
    }
}

```

5. Memanggil fungsi dari model melalui controller ke template

Kita buat terlebih dahulu model dengan nama "Testing.php" di folder "src/app>NamaModul/Model/". Lalu kita buat fungsi dengan nama "count_arr"

```

namespace App\Calendar\Model;

class Testing
{
    public function count_arr($arr)
    {
        $jumlah_arr = count($arr);

        return $jumlah_arr;
    }
}

```

Pada bagian controller kita panggil model "Testing.php" dengan cara menambahkan kode berikut di atas class controller

```

use App\Calendar\Model\Testing;

```

Setelah model di panggil, kita buat object baru dari class model yang dipanggil tadi dan kita kirim ke template

```

public function detail(Request $request)
{
    $arr = [1, 2, 3];
    $testing = new Testing;
    $count_arr = $testing->count_arr($arr);

    return $this->render_template('pegawai/detail', ['jumlah' => $count_arr]);
}

```

Setelah itu kita tampilkan datanya di template

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  Jumlah : <?= $jumlah ?>
</body>
</html>
```

6. Memanggil asset dari folder assets

Folder asset berada di “web/assets/”. Untuk memanggil asset seperti file css, javascript, dan image kita hanya perlu menambahkan path assets, (contoh: ‘/assets/css/style.css’). Berikut contoh penggunaannya pada template jika lokasi asset: “src/assets/css/style.css”, “src/assets/js/script.js”

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link href="/assets/css/style/css" rel="stylesheet">
</head>
<body>
  <h2>Welcome</h2>

  <script src="/assets/js/script.js"></script>
</body>
</html>
```

7. Buat dan ambil session

Untuk setiap proses pembuatan dan pengambilan session sudah tidak diperlukan lagi fungsi untuk memulai session (contoh: session_start()). Fungsi untuk memulai session sudah di tambahkan di GlobalFunc.php, yang perlu kita lakukan hanya membuat dan mengambil session.

Pertama-tama kita harus membuat constructor dan memanggil fungsi untuk memulai session pada controller yang digunakan saat menggunakan session, seperti contoh di bawah ini:

```
public function __construct()  
{  
    parent::beginSession();  
}
```

Untuk membuat session baru menggunakan fungsi

```
$this->session->set('test', 'bisa');
```

Dan untuk memanggil session menggunakan fungsi

```
$this->session->get('test');
```