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:

  > 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

- **Directory Structure**

```
example.com
|---- config
|---- src
|       |---- app
|       |       |---- Pegawai
|       |       |       |---- Controller
|       |       |       |       |---- PegawaiController.php
|       |       |       |---- Model
|       |       |               |---- Pegawai.php
|       |---- assets
|       |       |---- css
|       |       |---- js
|       |---- core
|       |       |---- Framework.php
|       |       |---- GlobalFunc.php
```

```
|       |---- pages
|       |       |---- welcome.php
|       |---- routes.php
|---- vendor
|       |---- autoload.php
|---- web
|       |---- front.php
|---- composer.json
|---- composer.lock
```

- **Penjelasan struktur direktori**
    - **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
        - **assets**
          Berisi kumpulan asset aplikasi seperti file css, js, dan image
        - **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

- **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 extensi file .php, tetapi hanya menggunakan nama file nya.

  2. **Menampilkan halaman melalui controller**
     Pertama-tama buat terlebih dahlu 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 menthod “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 “src/assets/”. Untuk memanggil asset seperti file css, javascript, dan image, sudah disediakan fungsi untuk memanggil asset dari folder “src/assets/”,

```
$assets('css/style.css');
```

berikut contoh penggunaanya 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>
```