Membuat Project Dengan Framework Phalcon

Pemrograman Web F 2018

Salah satu keunggulan dari Framework Phalcon adalah kita dapat mengatur sendiri struktur direktori dari project kita. Sehingga, kita bebas menentukan arsitektur yang sesuai dengan kebutuhan projek kita.

Dalam tutorial ini akan digunakan arsitektur Model – View – Controller (MVC) sederhana seperti gambar berikut ini.

```
my-project
+-- app
|   +-- cache
|   +-- config
|   +-- controllers
|   +-- models
|   +-- views
|   +-- Bootstrap.php
+-- public
|   +-- index.php
|   +-- .htaccess
+-- .htaccess
```

Referensi : https://github.com/phalcon/mvc/tree/master/simple

- Folder **app** berfungsi untuk menyimpan seluruh file fungsionalitas dari project kita seperti file konfigurasi, cache,controller, model, view dan class **Bootstraper.php**

- Folder **public** berfungsi sebagai *starting point* dari project kita ketika dijalankan nantinya, folder ini berisi file – file yang dapat diakses secara public oleh pengguna nantinya.

Silahkan buat project dengan direktori folder seperti diatas.

Melakukan Bootstraping

Yang harus pertama kali dilakukan ketika memulai project Phalcon baru adalah membuat sebuah file **bootstrap**. File ini berfungsi untuk melakukan konfigurasi pada project kita. Kita dapat melakukan inisialisasi komponen dan *behavior* aplikasi pada file ini.

Untuk melakukan bootstrapping ini, kita membutuhkan 2 buah file. Yaitu sebuah file **index.php** dan file class **Bootstrap.php**.

File **index.php** berfungsi sebagai *starting point* dari aplikasi kita, segala *request* yang masuk dari user akan masuk melalui file ini dan diarahkan ke *handler* yang sesuai dari project kita.

Buatlah sebuah file **index.php** dengan isi sebagai berikut di folder **public**.

```php
<?php

use Phalcon\Di\FactoryDefault;
use Phalcon\Mvc\Application;
use Phalcon\Db\Adapter\Pdo\Mysql as DBAdapter;

error_reporting(E_ALL);
ini_set('display_errors', 1);
header('Content-Type: text/html; charset=utf-8');
mb_internal_encoding("UTF-8");

// Define some absolute path constants to aid in locating resources
define('BASE_PATH', dirname(__DIR__));
define('APP_PATH', BASE_PATH . '\app');
// ...

require_once APP_PATH. '/Bootstrap.php';

$app = new Bootstrap();

$app->init();
```

Dalam file **index.php** ini kita melakukan beberapa hal :

1. Menyalakan sistem *error reporting* pada Phalcon untuk keperluan *debugging*.
2. Mengatur default *Content-Type* sebagai *teks / html*, dengan charset encoding UTF-8
3. Mendefiniskan sebuah konstanta **BASE_PATH** yang berisikan path ke folder project kita
4. Mendefinisikan sebuah konstanta **APP_PATH** yang berisikan path ke folder **app**

5. Men-*include* file class **Bootstrap.php** kemudian melakukan instansiasi dari class Bootstrap, dan menjalankan fungsi **init()** dari object tersebut.

Class **Bootstrap** berfungsi untuk memuat semua file class dan config, serta menjalankan *dependency injection* pada project kita. Maka, sebelum membuat class Bootstrap, buatlah 3 buah file pada folder **app/config**, yaitu **config.php**, **loader.php**, **services.php**

(Bisa ditambahkan file yang lain jika diperlukan konfigurasi tambahan dalam proyek kalian)

a. File **config.php** berfungsi untuk membuat sebuah object dari class **Phalcon\Config** yang menyimpan konfigurasi yang dibutuhkan project kita, seperti konfigurasi koneksi ke database, *base URL* dari aplikasi kita dan lainnya. Tujuan dari penggunaan class Config ini adalah untuk efisiensi penulisan konfigurasi. Agar apabila satu konfigurasi digunakan di banyak tempat, kita cukup memanggil attribute dari object Config saja.

   Buatlah sebuah file **config.php** dengan isi sebagai berikut:

   Pada bagian URL, isi bagian **{….}** dengan nama project kalian sendiri

```php
<?php

use Phalcon\Config;

return new Config([

    'database' => [
        'adapter' => 'Phalcon\Db\Adapter\Pdo\Mysql',
        'host' => '127.0.0.1',
        'username' => 'root',
        'password' => '',
        'dbname' => 'example-phalcon'
    ],
    'url' => [
        'baseUrl' => 'http://localhost/{nama-project-kalian}/'
    ]
]);
```

   Pada file tersebut, kita menginstansiasi class **Phalcon\Config**, dengan parameter konfigurasi untuk koneksi ke database dan URL project kita. Dapat ditambahkan konfigurasi lainnya yang dibutuhkan sesuai dengan kebutuhan project kalian.

b. File **loader.php** berfungsi untuk menjalankan **Loader** yang akan melakukan *autoloading* dari class – class yang ada di dalam project kita. Class yang akan di load biasanya adalah class Controller, Model maupun class eksternal yang disimpan di folder vendor.

   Ada 4 cara yang dapat dilakukan agar service Loader ini dapat menemukan class yang dimaksud. Yaitu dengan bedasarkan namespace, direktori, mendaftarkan class secara langsung dan mendaftarkan file class.

   Dalam kasus ini kita akan memuat class bedasarkan direktori, maka direktori yang kita daftarkan adalah direktori **app/controllers** dan **app/models**.

```php
<?php

$loader = new \Phalcon\Loader();

$loader->registerDirs(
    [
        APP_PATH . '/controllers/',
        APP_PATH . '/models/',
    ]
);

$loader->register();
```

c. File **services.php** berfungsi untuk mendaftarkan service – service yang dibutuhkan aplikasi kita ke dalam **Service Container** milik Phalcon.

   Untuk penjelasan lebih detil mengenai **Service Container** dan **Dependency Injection**, bisa dibaca di link berikut : https://docs.phalconphp.com/fr/3.2/di

   Mendaftarkan service ke dalam Service Container dilakukan dengan cara : melakukan instansiasi dari class **\Phalcon\DI\FactoryDefault**, kemudian memanggil fungsi **set()** dari objek tersebut dengan 2 parameter. Parameter pertama adalah sebuah string yang akan menjadi nama service yang didaftarkan. Parameter kedua adalah objek dari service yang didaftarkan (dalam tutorial ini parameter kedua akan diberikan melalui *anonymous function*).

   Beberapa service yang akan didaftarkan antara lain:

   - **View Engine**
     Sistem *templating* yang disediakan secara default pada framework Phalcon adalah **Volt**. Untuk mendaftarkan sistem templating Volt kedalam Service Container, tambahkan sintaks berikut di dalam **services.php**

```php
$di->set(
    'voltService',
    function ($view, $di) {
        $volt = new \Phalcon\Mvc\View\Engine\Volt($view, $di);

        $volt->setOptions([
            "compiledPath"      => APP_PATH . "/cache/",
            "compiledExtension" => ".compiled",
            "compileAlways"     => true,
        ]);

        return $volt;
    }
);
```

- **View**
  Setelah mendaftarkan sistem templating yang akan digunakan, selanjutnya kita daftarkan service view kedalam service container agar framework Phalcon dapat me-*render* tampilan. Ketika mendaftarkan service View, kita harus memberikan direktori tempat file **.volt** sebagai file *front-end* akan disimpan dan sistem templating yang akan digunakan. Kita akan menyimpan file .volt di direktori **app/views** dan menggunakan Volt sebagai sistem templating. Tambahkan sintaks di bawah ini kedalam file **services.php**

```php
$di->set(
    'view',
    function () {
        $view = new \Phalcon\Mvc\View();
        $view->setViewsDir(APP_PATH . "/views");
        // echo APP_PATH."\\views";

        $view->registerEngines([
            ".volt" => "voltService",
        ]);

        return $view;
    }
);
```

- **Koneksi ke Database**
  Agar project kita dapat terhubung dengan database, kita harus menambahkan service untuk melakukan koneksi ke database. Karena kita sudah mendaftarkan konfigurasi koneksi ke database ke dalam objek **Config** , kita tinggal memanggil atribut terkait konfigurasi koneksi ke database dari objek tersebut.

```
$di->set(
    'db',
    function () use ($config) {
        $dbAdapter = $config->database->adapter;

        return new $dbAdapter([
            "host" => $config->database->host,
            "username" => $config->database->username,
            "password" => $config->database->password,
            "dbname" => $config->database->dbname
        ]);
    }
);
```

- **Base URL**
  Terakhir kita daftarkan sebuah URI dasar agar semua URI yang di-*generate* oleh Phalcon sesuai dengan *base path* dari project kita.

```
$di->set(
    'url',
    function () use ($config, $di) {
        $url = new \Phalcon\Mvc\Url();

        $url->setBaseUri($config->path('url.baseUrl'));

        return $url;
    }
);
```

Setelah 3 file tersebut dibuat, saatnya kita membuat file class **Bootstrap.php** di folder **app** dengan isi sebagai berikut:

```php
<?php

use Phalcon\Mvc\Application;

class Bootstrap extends Application
{
    private function _registerServices()
    {
        $di = new \Phalcon\DI\FactoryDefault();

        $config = require APP_PATH. '/config/config.php';

        include_once APP_PATH. '/config/loader.php';
        include_once APP_PATH. '/config/services.php';

        $this->setDI($di);
    }

    public function init()
    {
        $debug = new \Phalcon\Debug();
        $debug->listen();

        $this->_registerServices();

        echo $this->handle()->getContent();
    }
}
```

Class Bootstrap merupakan turunan dari class **Phalcon\Mvc\Application**. Terdapat 2 fungsi, yaitu fungsi public **init()** dan fungsi private **_registerServices()**.

Fungsi **init()** berfungsi untuk menjalankan project kita, pada fungsi ini kita juga menjalankan class **\Phalcon\Debug** yang berfungsi untuk menjalankan debugger dan melakukan *error reporting* jika terjadi kesalahan. Kemudian menjalankan fungsi **_registerServices()**.

Pada fungsi **_registerServices()** kita menginstansiasi class **\Phalcon\DI\FactoryDefault()** yang berfungsi untuk mendaftarkan service – service yang kita butuhkan (yang kita lakukan pada file **services.php**). Kemudian, kita memuat konfigurasi, melakukan *autoloading class* dan mendaftarkan services masing – masing dengan memuat file **config.php, loader.php** dan **services.php**.

Setelah itu, kita memanggil fungsi **setDI()** dengan parameter object dari class **\Phalcon\DI\FactoryDefault()** yang telah berisi service – service yang kita daftarkan.

Menambahkan .htaccess

Kita perlu menambahkan file .htaccess untuk mengaktifkan mode rewrite URL dari server Apache. Disini kita membutuhkan file .htaccess di 2 tempat, yaitu di **root directory** dari project kita dan di **folder public**. Mengapa begitu?

Di server nantinya, kita akan mengakses web kita dari root directory sedangkan kita memerlukan file **index.php** sebagai *starting point* dari web kita. Sehingga, kita perlu mengalihkan URL dari root directory ke public directory sehingga server Apache bisa menemukan dan membaca file **index.php**.

Di root directory project kita buatlah sebuah file htaccess dengan isi sebagai berikut

```
<IfModule mod_rewrite.c>
    RewriteEngine on
    RewriteRule  ^$ public/    [L]
    RewriteRule  (.*) public/$1 [L]
</IfModule>
```

Kemudian di folder public buat file htaccess dengan isi sebagai berikut

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond   %{REQUEST_FILENAME} !-d
    RewriteCond   %{REQUEST_FILENAME} !-f
    RewriteRule   ^((?s).*)$ index.php?_url=/$1 [QSA,L]
</IfModule>
```

Kesimpulan

Karena sifatnya yang *loosely coupled*, framework Phalcon membebaskan kita untuk menyusun struktur direktori project kita sendiri. Agar Phalcon dapat membaca dan menggunakan semua class yang sudah kita buat, kita perlu melakukan *autoloading* terhadap class – class yang kita buat.

Salah satu keunggulan framework Phalcon adalah kita bisa menentukan sendiri service – service apa saja yang hendak kita pakai, sehingga ukuran projek kita relatif lebih kecil, karena hanya service yang benar – benar terpakai saja yang ada di dalam project.

Semua konfigurasi yang ada di tutorial ini hanyalah konfigurasi minimal agar project anda dapat berjalan, anda **bisa** menambah, mengurangi atau mengubah konfigurasi yang ada disini sesuai dengan kebutuhan project anda.

Referensi:

https://docs.phalconphp.com/id/3.2/tutorial-base
https://docs.phalconphp.com/fr/3.2/di