



Virtual Internship Experience

Code Optimization & Cleanup

1. PSR-1 (Basic Coding Standard) “Dasar Coding Standar”

Bertujuan memberikan standardisasi element-element kode PHP secara garis besar. Tujuannya adalah supaya kode-kode yang ditulis dengan PHP, memiliki kesamaan struktur.

2. PSR-2 (Coding Style Guide) “Panduan Gaya Penulisan”

PSR ini penambahan PSR-1 dan seperangkat aturan mengenai bagaimana memformat kode PHP

3. PSR-3 (Logger Interface) “Antarmuka Logger”

Peraturan PSR-3 untuk logging, dan secara khusus memaparkan sembilan metode untuk tata cara menulis log pada library framework. Hal ini untuk membuat pemangkasan universal untuk semua kerangka, yang berarti akan mempermudah ketika ingin menerapkan logging dalam framework yang sebelumnya tidak mempunyai fungsi tersebut.

4. PSR-4 (Improved Autoloading) “Peningkatan Autoloading”

Sebagai extention untuk PSR-0, PSR-4 menambahkan metode peningkatan autoloading, dan juga menjelaskan persyaratan path file dari autoloader interoperable.

5. PSR-5 (PHPDoc Standard) “Dokumen Standar PHP”

Tujuan utama dari PSR ini adalah untuk memberikan definisi yang lengkap dan formal standar PHPdoc. PSR ini menyimpang dari pendahulunya, de facto PHPdoc Standard terkait dengan phpDocumentor 1.x, untuk memberikan dukungan untuk fitur baru dalam bahasa PHP dan untuk mengatasi beberapa kekurangan dari pendahulunya.

6. PSR-6 (Caching Interface) “Antarmuka yg tersembunyi”

Tujuan dari PSR ini adalah untuk memungkinkan pengembang untuk membuat cache-menyadari perpustakaan yang dapat diintegrasikan ke dalam kerangka kerja dan sistem yang ada tanpa perlu untuk pengembangan kustom.

7. PSR-7 (HTTP Message Interface) “Pesan Antarmuka HTTP”

Ini menggambarkan antarmuka umum untuk mewakili pesan HTTP seperti yang dijelaskan dalam RFC 7230 dan RFC 7231, dan URI untuk digunakan dengan pesan HTTP seperti yang dijelaskan dalam RFC 3986.

Implementasi PSR 4 Autoloading pada PHP

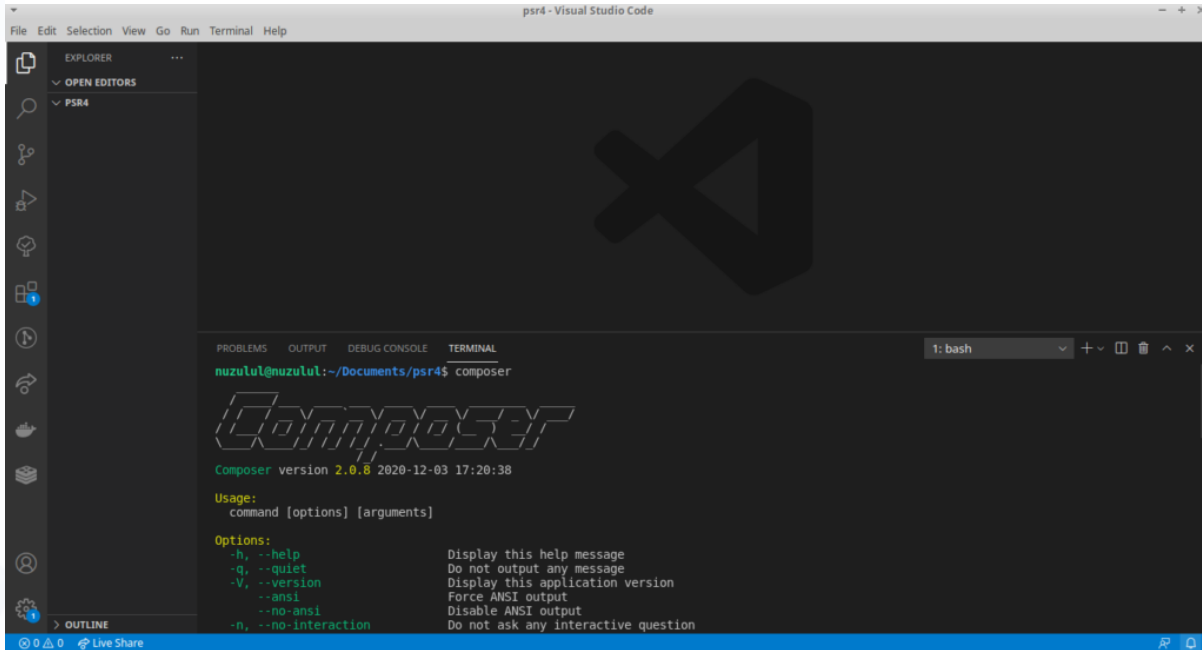
PSR atau singkatan dari *PHP Standards Recommendations*, yang mana artinya adalah rekomendasi-rekomendasi penulisan kode agar setiap programmer memiliki standar penulisan kode, sehingga mudah untuk melakukan kolaborasi atau kerja sama dalam menulis kode PHP, lebih lagi para programmer dapat membuat berbagai *library* hingga *framework* dari bahasa pemrograman PHP dengan standarisasi penulisan kode yang sama.

Mulai dari PSR 1 yang membahas tentang *Basic Coding Standard* hingga PSR 18 tentang HTTP Client kita dapat melihat selengkapnya tentang PSR ini melalui website resminya <https://www.php-fig.org/psr/>, tetapi untuk kali ini kita akan fokus membahas bagaimana implementasi dari PSR 4 yaitu *Autoloading Standard*.

Saat kita menulis kode php, kita tidak asing dengan memasukkan sebuah file ke dalam file lainnya, dengan memanggil fungsi *require* ataupun fungsi *include*, nah tentunya kita akan sangat merepotkan sekali jika skala project udah besar dan harus *include* satu per satu file dan besar kemungkinan bakal banyak kode *redundant*, nah dengan *Autoloading* ini akan menyelesaikan permasalahan tersebut, tentunya juga projek kita sudah menerapkan *Object Oriented Programming* sehingga kita dapat mengurangi atau bahkan menghilangkan kode-kode yang *redundant* dan dapat memanfaatkan kelebihan-kelebihan dari OOP itu sendiri.

Sebelum kita memulai praktek, pastikan kita sudah menginstall composer, jika belum install composer kita dapat melihat instruksi installnya melalui websitenya <https://getcomposer.org/download/>

Disini saya menggunakan *text editor* visual studio code karen ada fitur integrated terminalnya, nah kita dapat melakukan pengecekan composernya sudah terinstall atau belum dengan mengetikkan composer, makan akan muncul seperti gambar di bawah ini



The screenshot shows the Visual Studio Code interface with the terminal open. The terminal output displays the Composer version (2.0.6) and the usage instructions. The background of the terminal window features the Composer logo.

```

psr4 - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  OPEN EDITORS
  PSR4

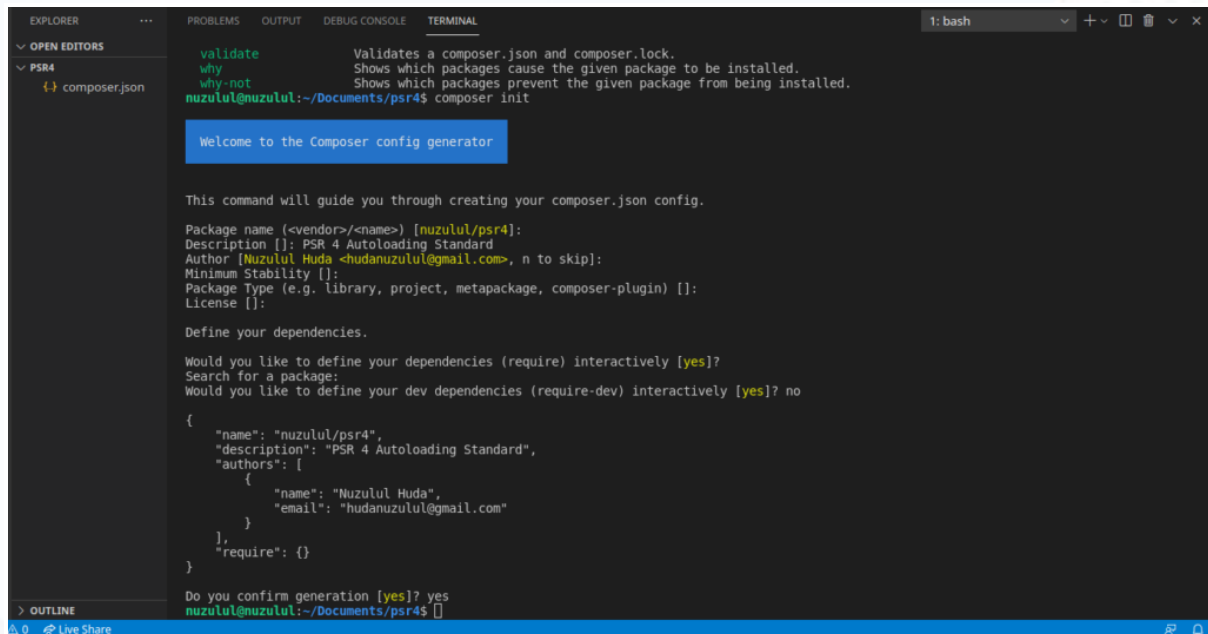
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
muzulul@muzulul:~/Documents/psr4$ composer

Composer version 2.0.6 2020-12-03 17:20:38

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi                   Force ANSI output
  --no-ansi                 Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
  
```


Selanjutnya kita kita ini project kita dengan perintah composer init seperti gambar di bawah ini, setelah menjalankan perintah ini, nanti bakalan otomatis tergenerate 1 buah file yaitu composer.json



```

validate      Validates a composer.json and composer.lock.
why           Shows which packages cause the given package to be installed.
why-not      Shows which packages prevent the given package from being installed.
nuzulul@nuzulul:~/Documents/psr4$ composer init

Welcome to the Composer config generator

This command will guide you through creating your composer.json config.

Package name (<vendor>/<name>) [nuzulul/psr4]:
Description []: PSR 4 Autoloading Standard
Author [Nuzulul Huda <hudanuzulul@gmail.com>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []:
License []:

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]?
Search for a package:
Would you like to define your dev dependencies (require-dev) interactively [yes]? no

{
    "name": "nuzulul/psr4",
    "description": "PSR 4 Autoloading Standard",
    "authors": [
        {
            "name": "Nuzulul Huda",
            "email": "hudanuzulul@gmail.com"
        }
    ],
    "require": {}
}

Do you confirm generation [yes]? yes
nuzulul@nuzulul:~/Documents/psr4$
  
```

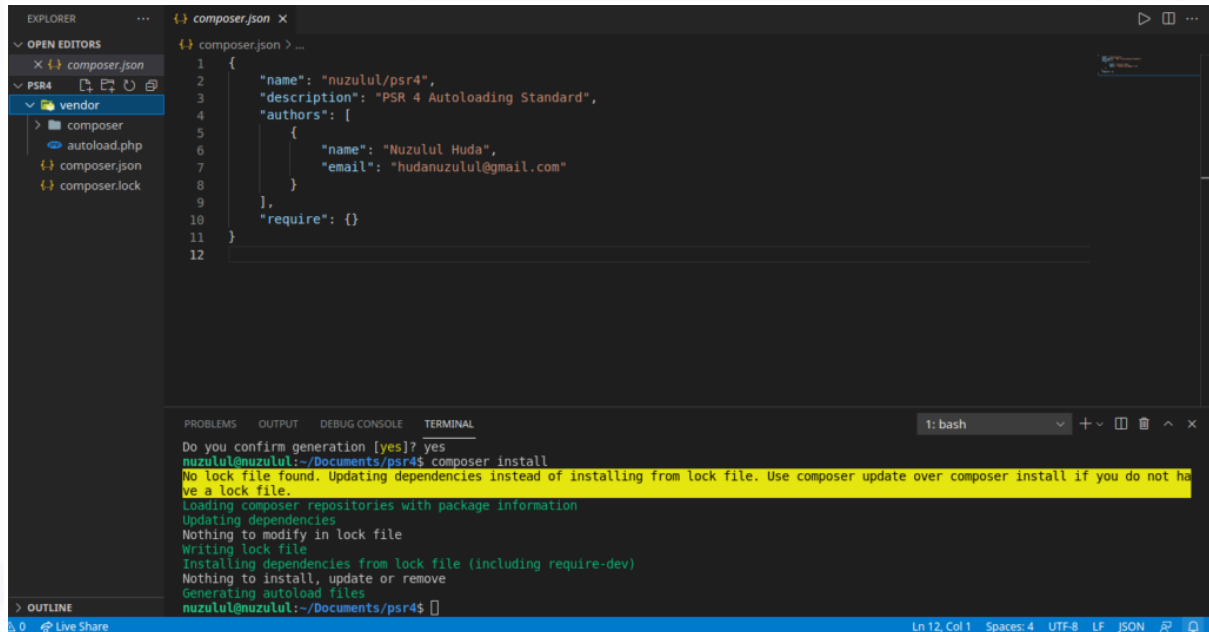
Berikut adalah ini dari file composer.json yang isinya sesuai dari apa yang kita isi tadi waktu menjalankan perintah composer init



```

{ } composer.json ×
{ } composer.json > ...
1  {
2      "name": "nuzulul/psr4",
3      "description": "PSR 4 Autoloading Standard",
4      "authors": [
5          {
6              "name": "Nuzulul Huda",
7              "email": "hudanuzulul@gmail.com"
8          }
9      ],
10     "require": {}
11 }
12
  
```

Kemudian kita jalankan perintah composer install, yang nantinya akan menghasilkan 1 folder tambahan pada proyek kita yaitu folder vendor yang di dalamnya terdapat file autoload.php dan juga folder composer, nah jika kita install package lain nantinya package tersebut akan berada di dalam folder vendor ini, seperti gambar di bawah ini



The screenshot shows an IDE with the following content:

EXPLORER:

- PSR4
 - vendor
 - composer
 - autoload.php
 - composer.json
 - composer.lock

composer.json:

```

1 {
2   "name": "nuzulul/psr4",
3   "description": "PSR 4 Autoloading Standard",
4   "authors": [
5     {
6       "name": "Nuzulul Huda",
7       "email": "hudanuzulul@gmail.com"
8     }
9   ],
10  "require": {}
11 }
12
```

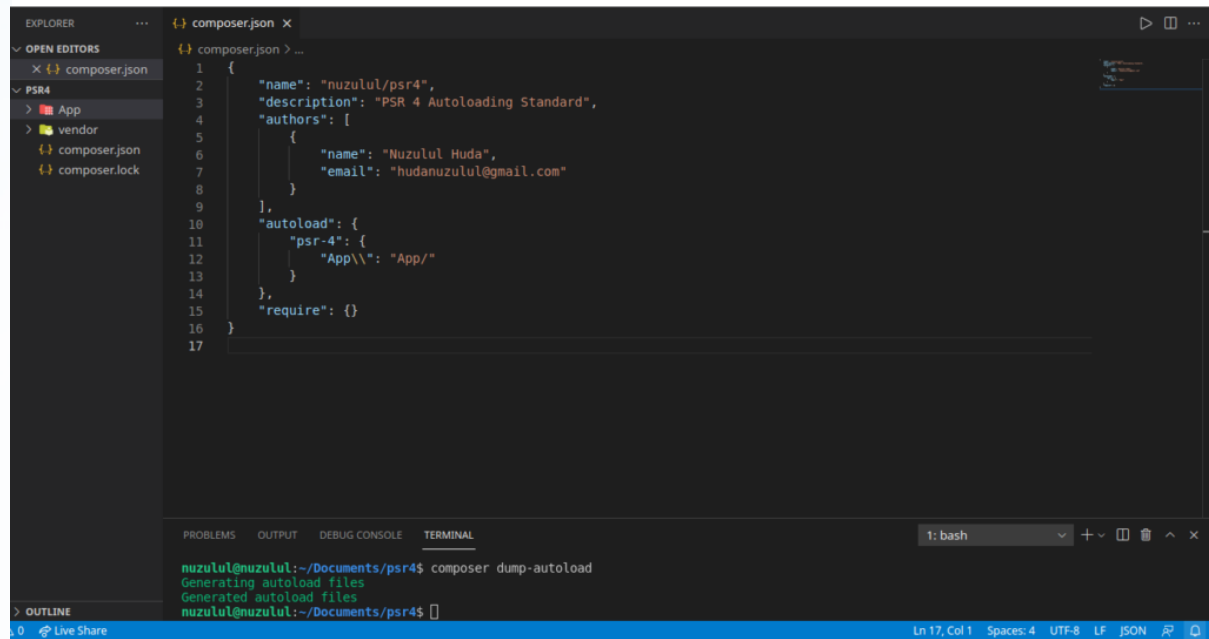
TERMINAL:

```

1: bash
Do you confirm generation [yes]? yes
nuzulul@nuzulul:~/Documents/psr4$ composer install
No lock file found. Updating dependencies instead of installing from lock file. Use composer update over composer install if you do not have a lock file.
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
nuzulul@nuzulul:~/Documents/psr4$
```

Selanjutnya kita bikin 1 folder App di dalam direktori proyek kita, satu level dengan direktori vendor, kemudian kita set autoloadnya ke folder tersebut dengan menambahkan kode di composer.json seperti gambar di bawah ini fungsi dari kode tersebut adalah aplikasi kita akan mendeteksi semua kelas di bawah folder App dengan namespace App.

Terakhir jalankan perintah composer dump-autoload untuk merefresh konfigurasi di composer.json



```

1 {
2   "name": "nuzulul/psr4",
3   "description": "PSR 4 Autoloading Standard",
4   "authors": [
5     {
6       "name": "Nuzulul Huda",
7       "email": "hudanuzulul@gmail.com"
8     }
9   ],
10  "autoload": {
11    "psr-4": {
12      "App\\": "App/"
13    }
14  },
15  "require": {}
16 }
17

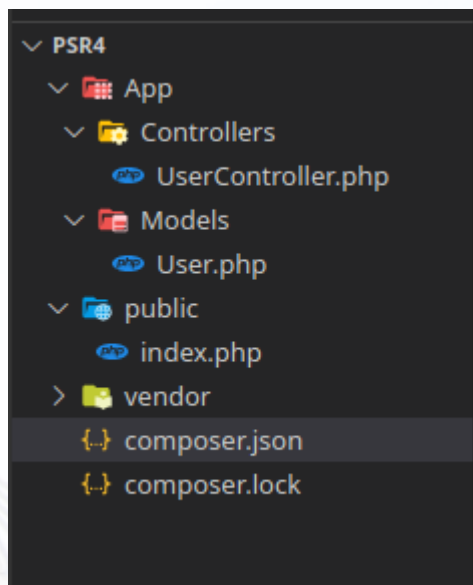
```

```

nuzulul@nuzulul:~/Documents/psr4$ composer dump-autoload
Generating autoload files
Generated autoload files
nuzulul@nuzulul:~/Documents/psr4$

```

Selanjutnya kita bikin struktur folder seperti di bawah ini biar seperti framework modern



```

PSR4
├── App
│   ├── Controllers
│   │   └── UserController.php
│   ├── Models
│   │   └── User.php
│   └── public
│       └── index.php
├── vendor
├── composer.json
└── composer.lock

```

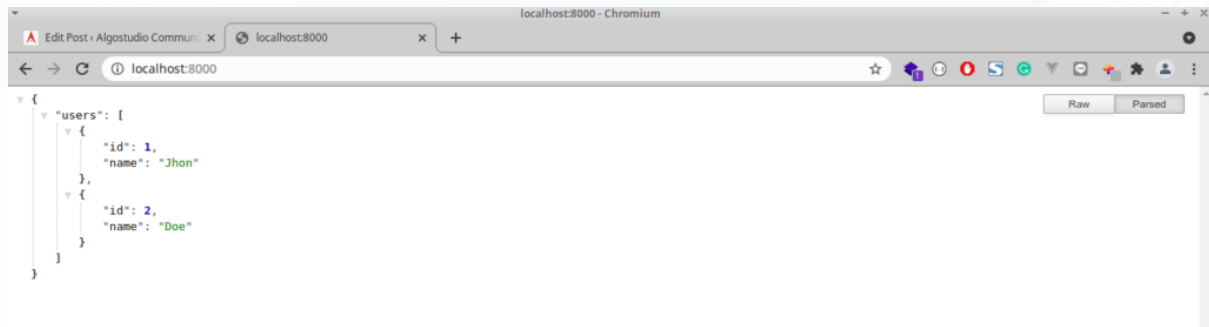
Kemudian isi UserController.php, User.php dan index.php seperti di bawah ini

```
1  <?php
2
3  namespace App\Controllers;
4
5  use App\Controllers\UserController;
6
7  require __DIR__ . '/../vendor/autoload.php';
8
9  $userController = new UserController();
10
11 header('Content-Type: application/json');
12 echo json_encode($userController->index());
```

```
1  <?php
2
3  namespace App\Models;
4
5  class User {
6
7      public function all()
8      {
9          return [
10             'users' => [
11                 ['id' => 1, 'name' => 'Jhon'],
12                 ['id' => 2, 'name' => 'Doe']
13             ]
14         ];
15     }
16 }
```

```
1  <?php
2
3  namespace App\Controllers;
4
5  use App\Models\User;
6
7  class UserController {
8
9      public function index()
10     {
11         $user = new User();
12
13         return $user->all();
14     }
15 }
```


Setelah itu kita ketikkan di terminal lagi perintah `php -S localhost:8000 -t public` kemudian buka `http://localhost:8000` di browser dan hasilnya akan seperti dibawah ini



Dengan memahami PSR 4 ini, kitapun juga bisa menggunakan komponen seperti routing, orm ataupun template engine yang biasanya di pakai oleh framework framework modern seperti Laravel, Symfony ataupun Zend Framework, bahkan kita juga bisa membuat framework sendiri menggunakan library library yang hanya kita butuhkan saja, atau kita juga dapat berkontribusi pada proyek *open source* yang dikerjakan oleh banyak programmer