Code Modularity & External Library

Web Programming and Testing

Mario Simaremare, S.Kom., M.Sc.
Program Studi Sarjana Sistem Informasi
Institut Teknologi Del

Objectives

- The objective of this session is the following:
  - The students are able to elaborate the benefit of developing solution in modular manner.
  - The students are able to select and use external library to speed up development.

Outlines

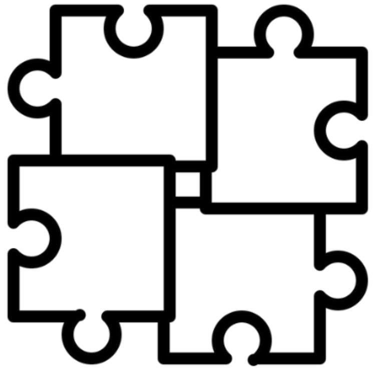
1. Motivation
2. Code modularity.
3. External library and code dependency.

- Note: we use PHP to show examples.

Motivation

Complexity

- It is very likely to have a complex solution to tackle complex problem.
  - Much harder to maintain (adding or modify features).
  - Bug-fixing could be a nightmare.
  - Less reusable.
- Who in the world would like to continue a messed up project?



Modularity

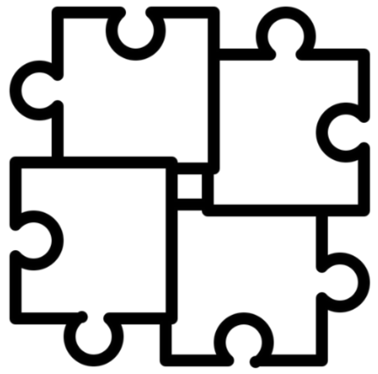
Modularity

- Modularity is a degree where codes are written into a group of independent units.
  - The unit can be in the form of functions, classes, modules, library, etc.

- A modular system or software should be:
  - low in coupling or dependency between units.
  - high in cohesion inside the unit.

Modularity: Benefit

- Much maintainable.
  - Small-scaled problem domain.
  - Easier for testing.
- Much more reusable.
  - Less duplicate codes.
  - The module can be exported into useful library for other project development.



Modularity in Functions

- Repeated routines should be written into functions with specific usage.
  - Enrich the function behaviors via parameters.

- Function parameters in PHP.
  - Accepts ordinary data type or special type.
    - Like: numbering, string, object, function, etc.
  - Forcing strictness.
    - Anonymously, strictly.

Modularity in Classes

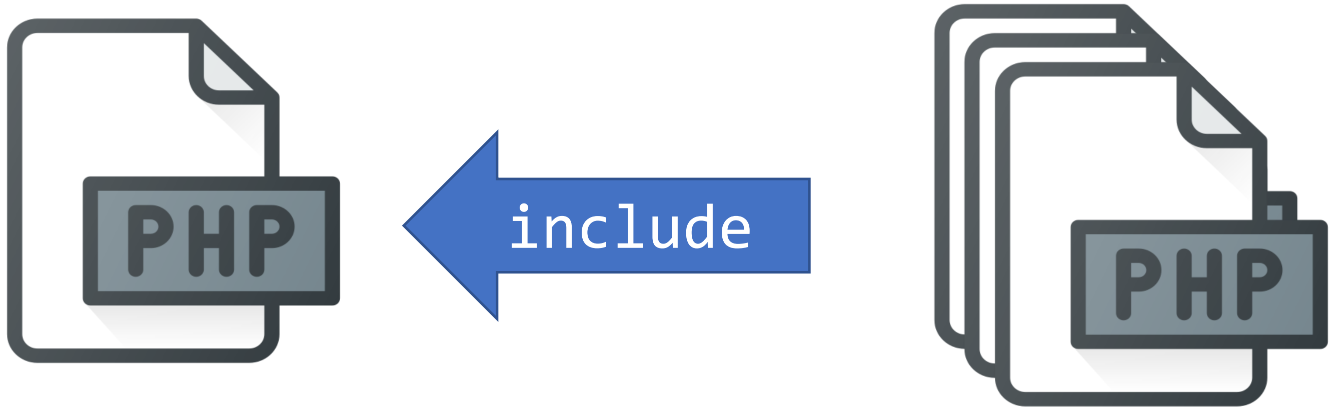
- Furthermore, functions can be redesigned and grouped into fully-equipped classes.
  - All OO concepts are applicable in PHP.
  - With different syntax and sugar.
- Later, a class can be namespaced.
  - Virtually, similar to package in Java with slightly different concept.

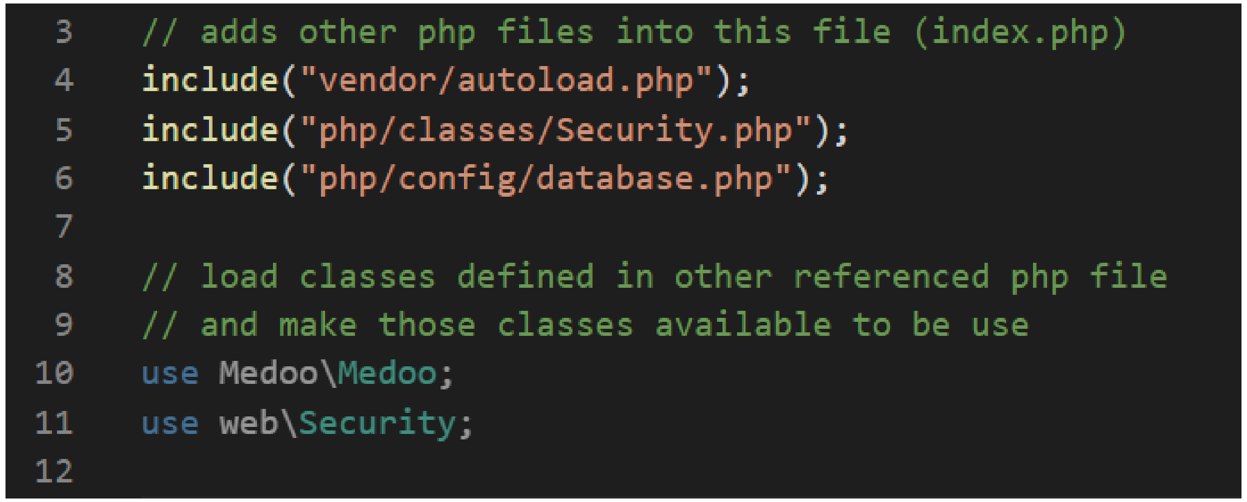
Importing Files

- For the sake of modularity codes could be:
  - written into functions and classes,
  - stored in different files (& different directory).

- To use the codes, functions and classes stored in other files, an import is required.
  - PHP provides two options with two variants for every option.
  - `include()` and `include_once()`
  - `require()` and `require_one()`

Importing Files



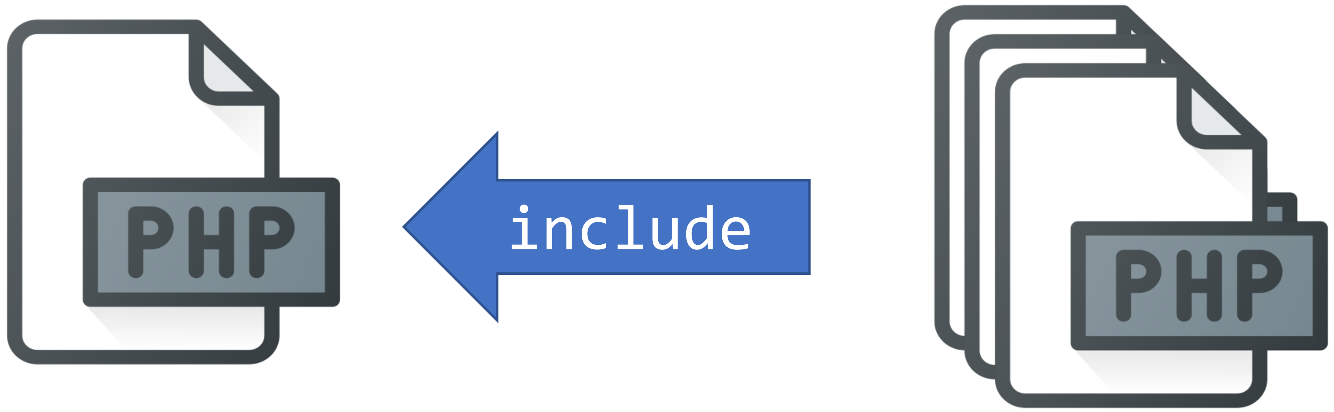


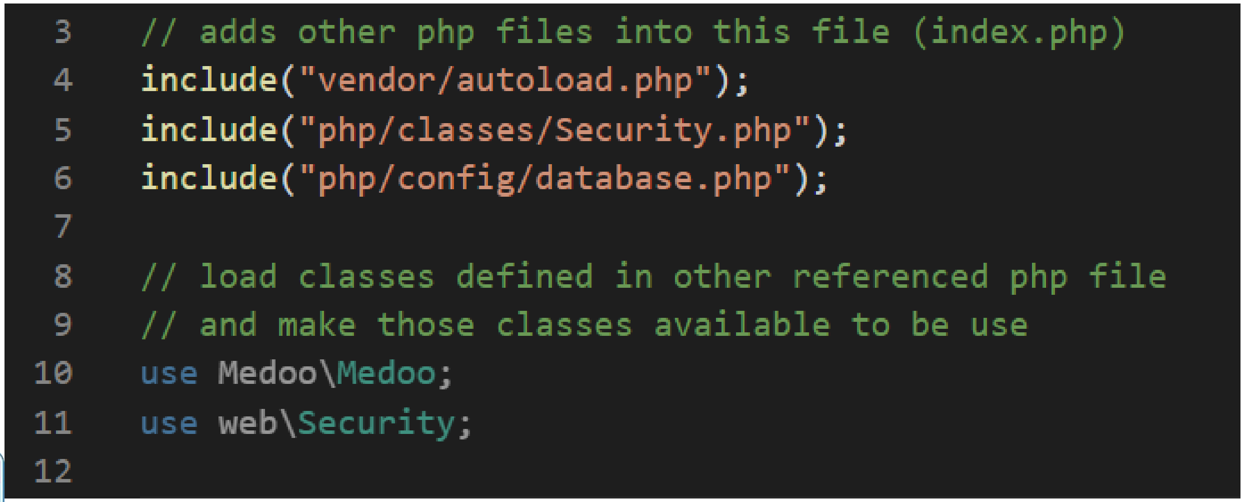
```php
// adds other php files into this file (index.php)
include("vendor/autoload.php");
include("php/classes/Security.php");
include("php/config/database.php");

// load classes defined in other referenced php file
// and make those classes available to be use
use Medoo\Medoo;
use web\Security;

```

Namespacing

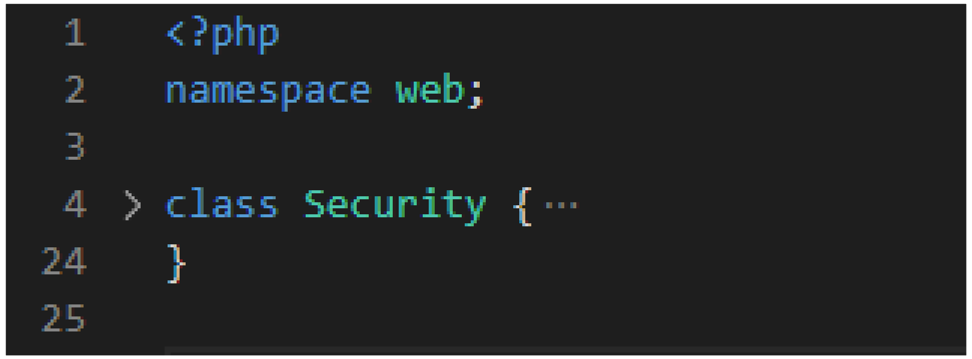
- Namespacing is simply a way to group units with unique alias or name.
- Multiple classes could be registered under one namespace.
  - Classes with the same namespace are not necessarily live in the same directory.
    - But encouraged to.
  - Classes, registered in a namespace, must be uniquely named.

Namespacing





```php
// adds other php files into this file (index.php)
include("vendor/autoload.php");
include("php/classes/Security.php");
include("php/config/database.php");

// load classes defined in other referenced php file
// and make those classes available to be use
use Medoo\Medoo;
use web\Security;

```



```php
<?php
namespace web;

class Security {⋯
}

```

External Library & Code Dependency Management

Benefits and Drawbacks

- Benefits:
  - Avoiding code rewriting (reuse).
  - Best practices done by the community.
  - Faster development.

- Drawbacks:
  - Extra learning curve.
  - Dependency (tight-coupling).
  - Additional layer of processing & complexity.
  - No silver bullet.

Code Dependency Management

- Goals:
  - Guarantee all the required libraries are available for application to live flawlessly.
  - Ensure that the project is always using the current version of library.
  - Avoid conflict between libraries.
- Package manager.
  - e.g. NuGet, RubyGems, npm, pip, Composer.

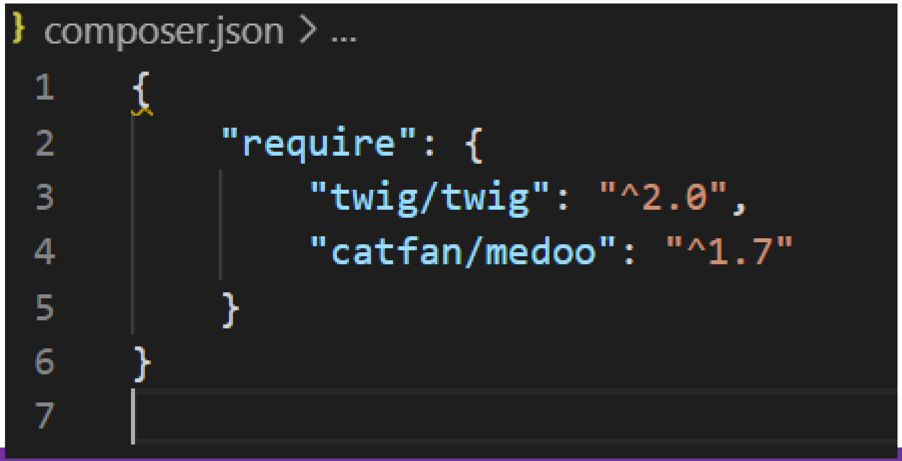
Dependency manager for PHP

- Existing dependency managers:
  - PEAR (PHP Extension and Application Repository),
  - PECL (PHP Extension Community Library),
  - Composer (the most popular at the moment).

- Composer is a project-level dependency manager. Very similar to `pip`, `npm` or `bundler`.

Composer

- It is a package dependency manager that:
  - takes care the required package in a project.
  - ensures all required package are in place and ready to use.
  - is configurable through a `JSON` file.
    - `composer.json`
    - different configuration for production and development envs.



composer.json > ...

```
{
    "require": {
        "twig/twig": "^2.0",
        "catfan/medoo": "^1.7"
    }
}
```

To-dos

1. Understand deeply the benefit of practicing code modularity.
2. Use some external library, such as:
   - Twig template engine.
   - Medoo PDO-related library
3. Next time, web framework.

References

Srinivasan, M. (2012). Web Technology: Theory and Practice. Pearson.

Tatroe, K., et. al. (2020). Programming PHP. O'Reilly.

PHP Manual https://www.php.net/manual/en/

Composer documentation. https://getcomposer.org/doc/

Twig documentation. https://twig.symfony.com/doc/2.x/

Medoo documentation. https://medoo.in/doc

Thank you