PERTEMUAN

6

Modul Praktikum

IF3028 PEMROGRAMAN WEB

T.A 2021/2022

PHP II

ASISTEN PRAKTIKUM
Alexander Diva
Aminudin Fadila
Andika Saputra
Daniel Sipangkar
Ringgo Galih Sadewo
Syabana Minggus N.

TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI PRODUKSI INDUSTRI DAN INFORMASI
INSTITUT TEKNOLOGI SUMATERA
OKTOBER 2021

Modul 6 PHP II

Tujuan Praktikum

Memberikan pemahaman kepada mahasiswa mengenai konsep lanjutan PHP seperti OOP, Error & Exceptions, dan Regex melalui studi kasus

Capaian Pembelajaran

Mahasiswa mampu menguraikan konsep dasar pemrograman PHP Bagian 2

Indikator Pembelajaran

Ketepatan analisis dan pemahaman mahasiswa mengenai implementasi OOP, Error & Exceptions, serta Regex di PHP

PEMROGRAMAN BERBASIS OBJEK DENGAN PHP

OOP (Object Oriented Programming) atau yang dalam bahasa Indonesia berarti Pemrograman Berbasis Objek (PBO) adalah konsep dimana Property / Variable dan juga Method / Fungsi di bungkus dalam sebuah Class, yang kemudian akan diterapkan pada Objek – objek yang dideklarasikan.
note: secara singkat, OOP PHP sama seperti OOP pada python yg dipelajari pada semester lalu (duck typing, overloading dengan magic function dsb.)

a. ***Class* dan *Object***

Class adalah sebuah "blueprints" untuk sebuah object, dengan kata lain bahwa class adalah cetakan dari object. Pada bahasa pemrograman, class merupakan sekelompok kode yang dituliskan untuk mendefinisikan properti dan method yang ada pada sebuah object.
Class didefinisikan dengan memuat properti dan metode, dimana properti adalah sebuah data yang menjelaskan tentang class dan metode adalah tingkah laku yang bisa dilakukan oleh object.
Berikut adalah contoh kode sebuah class yang dilengkapi dengan properti dan metode.

index.php > User > setID

You, a minute ago | 1 author (You)

```php
<?php

class User
{
    var $nama;
    //var secara default memiliki visibility public
    private ?int $id;
    //php7.4 keatas memungkinkan untuk memberikan tipe data pada properti,
    //tanda tanya (?) disebelum deklarasi tipe data berarti "properti tsb
    //dapat berisi NULL

    public function setID($id)      You, seconds ago • Uncommitted changes
    {
        $this->id = $id;
    }
    public function info()
    {
        echo "user bernama {$this->nama} yang ber-ID {$this->id}";
        // 'user bernama ' . $this->nama .' yang ber-ID ' . $this->id
    }
}

//meng-instansiasi sebuah class menjadi object
$joko = new User();
$joko->nama = "joko";
$joko->setID(98999);

$joko->info();
```

Best practice yang biasa dilakukan adalah membuat file php tersendiri yang menampung class tersebut, agar dapat digunakan pada file-file lain menggunakan *import* ataupun *function* lainnya.

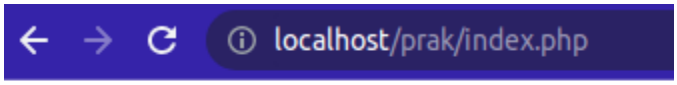
index.php M × User.php U

index.php > ...

You, seconds ago | 1 author (You)

```php
<?php
//memanggil module User
require_once("User.php");
      You, seconds ago • Uncommitted changes
//meng-instansiasi sebuah class menjadi object
$joko = new User();
$joko->nama = "joko";
$joko->setID(98999);

$joko->info();

```

hasil:



user bernama joko yang ber-ID 98999

b. ***Constructor* dan *Destructor***

Constructor merupakan sebuah fungsi di dalam class (method) yang secara otomatis dipanggil dan dijalankan ketika membuat sebuah objek dari sebuah class. Sesuai namanya constructor (pembuat) adalah method yang akan dijalankan ketika kita membuat objek. Constructor sangat cocok digunakan untuk menginisialisasi (memberikan nilai awal) ke property.

```php
class User
{
    var $nama;
    //var secara default memiliki visibility public
    private ?int $id;
    //php7.4 keatas memungkinkan untuk memberikan tipe data pada properti,
    //tanda tanya (?) disebelum deklarasi tipe data berarti "properti tsb
    //dapat berisi NULL

    public function __construct($nama)
    {
        $this->nama = $nama;
    }

    public function setID($id)
```

```php
//meng-instansiasi sebuah clas

$joko = new User('joko');
//$joko->nama = "joko";
```

Destructor adalah sebuah fungsi di dalam class (method) yang dijalankan atau dipanggil secara otomatis ketika objek dihancurkan atau ketika script berakhir. Jadi fungsi ini adalah kebalikan dari fungsi constructor dimana fungsi ini dibuat di akhir sebuah class.
Biasanya, destructor berisi penutupan koneksi jika objek melakukan akses ke file ataupun database untuk menghindari *memory leak* (baca pengertian memory leak di wiki)

```php
}
public function __destruct()
{
    echo "Sesi {$this->nama} telah berakhir";
}
```

user bernama joko yang ber-ID 98999Sesi joko telah berakhir

Perhatikan bahwa destructor secara otomatis terpanggil saat request telah selesai dan akan dikirim ke client

c. ***Inheritance* (pewarisan)**

Inheritance merupakan sebuah konsep penurunan atau pewarisan sifat (Method dan Property) dari sebuah class ke class lain. Artinya dengan Inheritance sebuah class dapat menggunakan property dan method yang ada pada class yang diturunkan. Inheritance menimbulkan hirarki hubungan sebuah class, dimana class utama yang diturunkan sifatnya disebut sebagai class induk/ parent class/ super class/ base class, lalu class turunan yang menerima sifat dari class utama disebut class anak/ sub class. Penurunan class akan membuat kita dapat menggunakan kembali kode yang telah kita buat (reuse code), sehingga penulisan program lebih efektif, efisien, terstruktur dan tidak menimbulkan duplikasi kode. Cara membuat class turunan adalah dengan meng extends nya ke class induk.

index.php M × UserPlatinum.php U ×

UserPlatinum.php > UserPlatinum > info

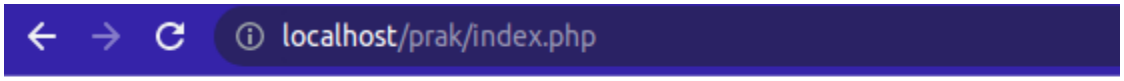
```php
<?php
require_once("User.php");

class UserPlatinum extends User
{
    var $level;

    public function __construct($nama)
    {
        parent::__construct($nama);
        $this->level = 'platinum';
    }

    public function info()
    {
        echo "user bernama {$this->nama} yang ber-ID {$this->id} dengan level {$this->level}";
    }
}

```

index.php M × UserPlatinum.php U

index.php > ...

You, seconds ago | 1 author (You)

```php
<?php
//memanggil module User
require_once("UserPlatinum.php");

//meng-instansiasi sebuah class menjadi object

$joko = new UserPlatinum('joko');       You, seconds a
//$joko->nama = "joko";

$joko->setID(98999);

$joko->info();

```

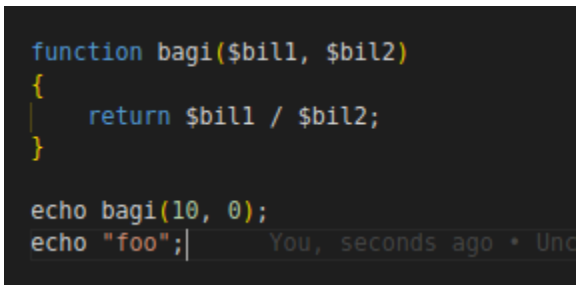


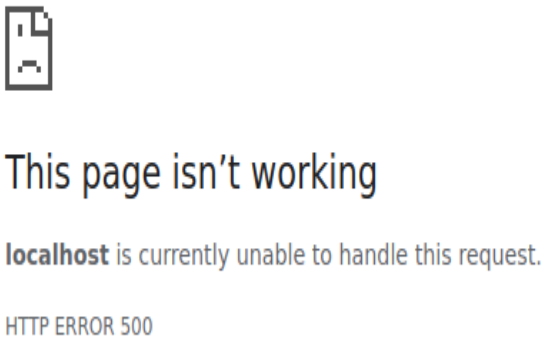
user bernama joko yang ber-ID dengan level platinumSesi joko telah berakhir

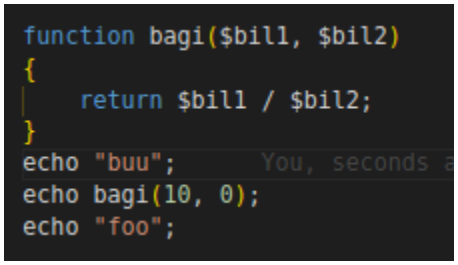
Handling Error dengan Exception

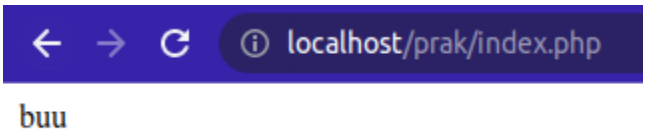
Kemunculan error saat pengembangan suatu aplikasi adalah sesuatu yang tidak-bisa-tidak. Ia pasti terjadi. Bahkan ketika aplikasi/web yang kita kerjakan sudah rilis pun, error tetap menjadi makanan sehari-hari kita. Ia pasti dan harus muncul. Error apa pun yang muncul, harus ditangani dengan tepat. Baik ketika aplikasi masih dalam pengembangan, lebih-lebih lagi ketika aplikasi telah dilempar ke publik.
Exception adalah cara baru untuk menangani error pada PHP. Ia menggunakan pendekatan OOP (Object Oriented Programming) berbeda dengan sebelumnya (versi < 5) yang mana pendekatan yang dilakukan masih prosedural. Exception secara bahasa berarti pengecualian. Sedangkan secara istilah di dalam PHP, ia adalah sebuah perubahan alur program dari kondisi normal ke kondisi tertentu (atau pengecualian tertentu) jika terjadi suatu error (exception).
In short, exception adalah bagian kode program yang membutuhkan perlakuan khusus karena ia berjalan tidak sesuai dengan yang seharusnya.

Perhatikan skenario berikut





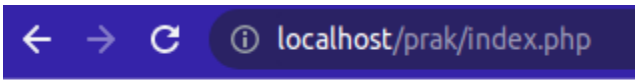




Jika terjadi error, kode berikutnya tidak akan dibaca, hal tersebut menghancurkan *flow* dari kode nya. Maka dari itu diperlukan *error handling*, ***Exception*** salah satu dari dari handling error yang harus dipahami, karena banyak *library* / *framework* menggunakan exception untuk handle errornya.

Berikut adalah penggunaan exception dengan try catch,

```php
function bagi($bil1, $bil2)
{
    try {
        return $bil1 / $bil2;
    } catch (DivisionByZeroError $th) {
        return "tidak bisa dilakukan";
    }
}
echo "buu";
echo bagi(10, 0);
echo "foo";
```



buutidak bisa dilakukanfoo

Perhatikan pada blok catch, terdapat (DivisonByZeroError) yang merupakan exception *built-in* PHP untuk menangkap error pembagian dengan nol, Kalian juga dapat membuat exception sendiri dengan *keyword* ***throw***.

```php
<?php
require_once("User.php");
//membuat class untuk catch custom error
class UmurNegatifErr extends Exception
{
}
class BelumLahirErr extends Exception
{
}

class UserPlatinum extends User
{
    var $level;
    var int $umur;
    var int $tahun_lahir;

    public function __construct($nama, $umur, $tahun_lahir)
    {
        parent::__construct($nama);
        $this->level = 'platinum';
        if ($umur < 0) {
            throw new UmurNegatifErr("umur kurang dari 0");
        }
        if ($tahun_lahir > 2021) {
            throw new BelumLahirErr("tahun lahir lebih dari tahun sekarang");
        }
        $this->umur = $umur;
        $this->tahun_lahir = $tahun_lahir;
    }

    public function info()
    {
        echo "user bernama {$this->nama} yang ber-ID {$this->id} dengan level {$this->level}";
    }
}
```

index.php M × UserPlatinum.php U

index.php > ...

You, seconds ago | 1 author (You)

```php
<?php
//memanggil module User
require_once("UserPlatinum.php");

//meng-instansiasi sebuah class menjadi object

try {
    $joko = new UserPlatinum('joko', 10, 2023);
    //$joko->nama = "joko";

    $joko->setID(98999);

    $joko->info();
} catch (UmurNegatifErr $th) {
    echo "pastikan anda telah lahir";
} catch (BelumLahirErr $th) {
    echo "yang belum lahir ga bole daftar";
}
```

yang belum lahir ga bole daftar

*chaining catch

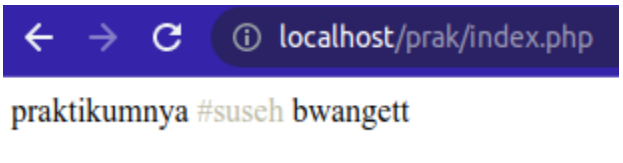
Regular Expression **(regex) di PHP**

Regular Expression atau biasa disingkat regex, adalah suatu metode untuk mengenali atau mendeteksi suatu pola tertentu pada suatu string. Dengan menggunakan regex, kita bisa mendeteksi pola string seperti email, hashtag, link dan pola-pola kompleks lainnya dengan hanya satu ekspresi saja. Ia juga merupakan metode standar dan independen, tidak mengenal bahasa pemrograman. Kita bisa mengimplementasi regex di berbagai macam bahasa pemrograman: termasuk PHP.
Di dalam PHP, terdapat beberapa fungsi yang berkaitan dengan penggunaan regex. Di antaranya:

Fungsi	Deskripsi
`preg_match()`	Mencari kata/karakter yang sesuai dengan pola regex.
`preg_match_all()`	Mencari *semua* kata/karakter yang sesuai dengan pola regex.
`preg_replace()`	Mencari kata/karakter yang sesuai dengan pola regex, lalu menimpanya dengan data baru.
`preg_grep()`	Mengembalikan kata/karakter yang sesuai dengan pola regex.
`preg_split()`	Membagi string menjadi sebuah array menggunakan pola regex.

Untuk sintaks regex silahkan baca https://regexcheatsheet.com/

```php
$post = "praktikumnya #suseh bwangett";

echo preg_replace(
    '/#[a-z]+/i',
    "<span style='color:#bbb9a4;'>$0</span>",
    $post,
);
```



(i) pada akhir pattern memiliki makna “case-**in**sensitive” yang berarti tidak memperhatikan besar kecil huruf.