Pemrograman Web – Minggu 7

PHP Cookie, Session, Filter, Advanced Filter, Predefined Variables, Regular Expressions, & Error Exception Handling

Oleh Dimas Rizky H.P. – 2110141011 – 3 D4 IT A

***PHP Cookies***

Dalam web programming, cookie merupakan salah satu pendekatan untuk mengidentifikasi seorang user. Cookie merupakan kode servver yang di *embed* pada komputer user. Setiap kali komputer yang sama melakukan *request page*, server akan mengirimkan cookie. Dengan begitu, programmer dapat mendapatkan dan menananmkan nilai cookie pada browser milik user dengan memanfaatkan skrip PHP.

* ***Membuat Cookies dengan PHP***

Cookie dapat dibuat dengan menggunakan fungsi *setcookie()*. Contoh syntax untuk membuat cookie dengan PHP dapat dilihat pada *Box Code 1.* Parameter yang wajib ada adalah parameter nama.

setcookie(name, value, expire, path, domain, secure, httponly);

*Box Code 1 – fungsi setcookie() untuk membuat cookie*

* ***Mengambil nilai cookies dengan PHP***

Ketika cookie sudah tertanam pada komputer user, cookie tersebut dapat diakses melalui global variabel $\_COOKIE. Pada *Box Code 2* diperlihatkan cara membuat dan mengakses cookie.

<?php

$cookie\_name = "user";

$cookie\_value = "Dimas Rizky";

setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/");

?>

<**html**>

<**body**>

<?php

**if**(!**isset**($\_COOKIE[$cookie\_name])){

**echo** "Cookie bernama '" . $cookie\_name . "' belum di set!";

    }**else**{

**echo** "Cokie '" . $cookie\_name . "' sudah di set <br>";

**echo** "Nilainya adalah : " . $\_COOKIE[$cookie\_name];

    }

  ?>

</**body**>

</**html**>

*Box Code 2 – Membuat dan mengakses cookie*

**

*Gambar 1.0 output box code 2*

* ***Memodifikasi nilai cookie dengan PHP***

Untuk memodifikasi sebuah cookie, hanya perlu menjalankan fungsi *setcookie()* kemabli atau reset sebuah cookie untuk menimpa/*overwrite* nilai dari cookie tersebut. Contoh penerapan dapat dilihat pada *Box Code 3*.

<?php

$cookie\_name = "user";

$cookie\_value = "Harsoyo Putro";

setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/");

?>

<**html**>

<**body**>

<?php

**if**(!**isset**($\_COOKIE[$cookie\_name])){

**echo** "Cookie bernama '" . $cookie\_name . "' belum di set!";

    }**else**{

**echo** "Cokie '" . $cookie\_name . "' sudah di set <br>";

**echo** "Nilainya adalah : " . $\_COOKIE[$cookie\_name];

    }

  ?>

</**body**>

</**html**>

*Box Code 3 – Memodifikasi cookie*

**

*Gambar 1.1 output box code 3*

* ***Menghapus cookie***

Untuk menghapus sebuah cookie, dapat dilakukan juga dengan menggunakan fungsi *setcookie()* namun pada paramater *expiration date* di set ke tanggal yang sudah lampau. Contohnya pada *Box Code 4.*

<?php

setcookie("user", "", time() - 3600);

?>

<**html**>

<**body**>

<?php

**echo** "Cookie 'user' sudah dihapus.";

?>

</**body**>

</**html**>

*Box Code 4 – Menghapus cookie*

**

*Gambar 1.2 output box code 4*

* ***Memeriksa apakah fitur cookies dinyalakan atau tidak***

Dalam beberapa kasus keamanan, user lebih memilih untuk mematikan fungsi cookie di browser miliknya. Untuk melakukan pemeriksaan, apakah fitur cookie diaktifkan oleh user dapat menggunakan trik pada *Box Code 5*

<?php

setcookie("test\_cookie", "test", time() + 3600, '/');

?>

<**html**>

<**body**>

<?php

**if**(count($\_COOKIE) > 0) {

**echo** "Cookies diaktifkan.";

} **else** {

**echo** "Cookies dinonaktifkan.";

}

?>

</**body**>

</**html**>

*Box Code 5 – Memeriksa apakah fitur cookie aktif atau tidak*

**

*Gambar 1.3 output box code 5*

Pada B*box Code 5* dapat dilihat, langkah awalnya adalah dengan memanggil fungsi *setcookie()* untuk membuat cookie baru, lalu melakukan penghitungan pada variabel $\_COOKIE. Jika di dalam variabel tersebut terdapat lebih dari 0 elemen, berarti fitur cookie diaktifkan oleh user.

***PHP Session***

Ketika sebuah aplikasi dijalankan oleh user, user dapat membuka sebuah modul, mengganti nilainya, lalu menutupnya. Jika aplikasi tersebut kembali dibuka, maka user akan tetap bisa mengakses modul modul yang ada dengan kondisi sama dengan kondisi sebelum user menutup aplikasi tersebut. Hal ini disebut session, dimana komputer mengetahui siapa yang mengakses aplikasi tersebut.

Namun terdapat masalah jika hal ini diterapkan pada aplikasi web. Dimana web server tidak akan bisa mengetahui siapa user yang mengakses dan apa yang sudah user akses, hal ini dikarenakan karena alamat HTTP tidak menyimpan state dari apa yang pernah user lakukan.

Variabel session pada PHP dapat menyelesaikan masalah ini dengan cara menyimpan informasi user untuk digunakan diseluruh pages yang ada. Secara *default* variabel session akan disimpan sampai dengan user menutup browsernya

* ***Memulai Session PHP***

Session dapat dimulai dengan fungsi *session\_start()*. Variabel session akan di set pada variabel global $\_SESSION. Contoh pembuatan session dapat dilihat pada *Box Code 6.*

<?php

  session\_start();

?>

<**html**>

<**body**>

<?php

    $\_SESSION["favcolor"] = "green";

    $\_SESSION["favanimal"] = "cat";

**echo** "Variabel session sudah di set";

  ?>

</**body**>

</**html**>

*Box Code 6 – Memulai Session*

**

*Gambar 2.0 output box code 6*

* ***Mendapatkan nilai variabel pada Session PHP***

Untuk mendapatkan nilai variabel pada sebuah session bisa dilakukan dengan mengakses variabel global $\_SESSION. Contoh pengaksesan dapat dilihat pada *Box Code 7*.

<?php

  session\_start();

?>

<**html**>

<**body**>

<?php

**echo** "Nilai variabel favorite color adalah " . $\_SESSION["favcolor"] . ".<br>";

**echo** "Nilai variabel favorite animal adalah " . $\_SESSION["favanimal"] . ".";

  ?>

</**body**>

</**html**>

*Box Code 7 – Mendapatkan nilai variabel pada sebuah session*

**

*Gambar 2.1 output box code 7*

Cara lain untuk melihat variabel yang ada pada variabel $\_SESSION adalah dengan menggunakan fungsi *print\_r()*. Contohny dapat dilihat pada *Box Code 8.*

<?php

  session\_start();

?>

<**html**>

<**body**>

<?php

    print\_r($\_SESSION);

  ?>

</**body**>

</**html**>

*Box Code 8 – Akses variabel $\_SESSION menggunakan vardump()*

**

*Gambar 2.2 output box code 8*

Cara kerja session antara lain adalah dengan membuat set user-key pada komputer user. Lalu, ketika sebuah session dimulai pada page lain, sistem akan melakukan scan pada user-key yang sudah ada. Jika ada yang cocok, maka session tersebut yang akan diakses, jika tidak, maka sistem akan memulai session baru.

* ***Memodifikasi variabel session***

Untuk memodifikasi variabel sebuah session dapat dilakukan penumpukan/*overwrite* pada global variabel $\_SESSION. Contohnya dapat dilihat pada *Box Code 9*.

<?php

session\_start();

?>

<!DOCTYPE html>

<**html**>

<**body**>

<?php

// untuk menggantinya, cukup menimpa nilainya saja

$\_SESSION["favcolor"] = "yellow";

print\_r($\_SESSION);

?>

</**body**>

</**html**>

*Box Code 9 – Memodifikasi nilai variabel sebuah session*

**

*Gambar 2.3 output box code 9*

* ***Menghancurkan/menghentikan session***

Untuk menghapus semua variabel session dan menghentikan session yang sedang berjalan, dapat menggunakan fungsi *session\_unset()* dan *season\_destroy()*. Contohnya ada pada *Box Code 10*.

<?php

session\_start();

?>

<!DOCTYPE html>

<**html**>

<**body**>

<?php

// menghapus seluruh variabel session

session\_unset();

// menghancurkan session

session\_destroy();

**echo** ("Variabel session yang ada ".$\_SESSION["favcolor"]);

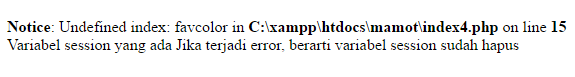
**echo** ("<br>Jika terjadi error, berarti variabel session sudah hapus");

?>

</**body**>

</**html**>

*Box Code 10 – Menghapus & Menghentikan session*

**

*Gambar 2.4 output box code 10*

***PHP Filter***

Filter pada PHP utamanya digunakan untuk melakukan *validating data* dan  *sanitizing data*. *Validating data* adalah menentukan apakah data dalam bentuk yang sesuai dengan yang diinginkan atau tidak, *sanitizing data* adalah menghapus segala macam karakter illegal/bentuk ilegal dalam sebuah data.

Ada alasan mengapa sebuah program web harus menggunakan filter, yaitu dikareanakan banyak kasus dimana aplikasi membutuhkan input dari user. Input tersebut dapat berbentuk input form, cookies, data dari web service, variabel server, hasil dari query database dan lainnya. Dengan menerapkan filter akan memastikan bahwa aplikasi memproses data input yang benar benar sesuai dengan yang dibutuhkan.

* ***PHP Filter Extension***

PHP filters digunakan untuk *validate* dan *sanitize* input yang berasal dari eksternal. Filter PHP memiliki banyak fungsi mulai dari pemeriksaan user input hingga dengan validasi input sesuai dengan format yang ada. Untuk melihat apa saja fungsi yang bisa disediakan oleh PHP filter. Contohnya dapat dilihat pada *Box Code 11*.

<!DOCTYPE html>

<**html**>

<**head**>

<**style**>

table, th, td {

**border**: 1px solid black;

**border-collapse**: collapse;

}

th, td {

**padding**: 5px;

}

</**style**>

</**head**>

<**body**>

<**table**>

<**tr**>

<**td**>Filter Name</**td**>

<**td**>Filter ID</**td**>

</**tr**>

<?php

**foreach** (filter\_list() **as** $id =>$filter) {

**echo** '<tr><td>' . $filter . '</td><td>' . filter\_id($filter) . '</td></tr>';

  }

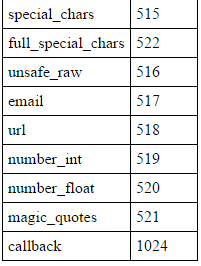
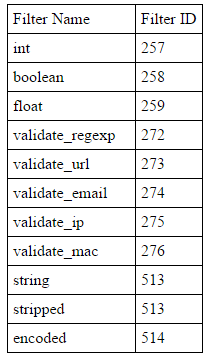
  ?>

</**table**>

</**body**>

</**html**>

*Box Code 11 – Fungsi filter\_list()*

**

*Gambar 3.0 output box code 11*

* ***Fungsi filter\_var()***

Fungsi *filter\_var()* adalah untuk melakukan validasi dan pembersihan data. Fungsi tersebut akan mem-filter variabel tunggal dengan rule spesifik. Penggunaannya dapat dilihat pada section di bawah ini.

* ***Sanitize(membersihkan) sebuah string***

Berikut adalah contoh penggunaan fungsi *filter\_var()* untuk menghapus seluruh tag HTML dari sebuah string. Penerapannya dapat dilihat pada *Box Code 12.*

<?php

$str = "<h1>Hello World!</h1>";

$newstr = filter\_var($str, FILTER\_SANITIZE\_STRING);

**echo** $newstr;

?>

*Box Code 12 – Fungsi filter\_list() untuk memfilter tag HTML*

**

*Gambar 3.1 output box code 12*

* ***Validate(memvalidasi) sebuah integer***

Berikut adalah contoh penggunaan fungsi *filter\_var()* untuk memeriksa apakah suatu variabel merupakan data bertipe integer atau tidak. Penerapannya dapat dilihat pada *Box Code 13.*

<?php

$int = 100;

**if** (!filter\_var($int, FILTER\_VALIDATE\_INT) === false) {

**echo**("Integer valid");

} **else** {

**echo**("Integer tidak valid");

}

?>

*Box Code 13 – fungsi filter\_list() untuk melakukan cek data integer*

**

*Gambar 3.2 output box code 13*

Namun pada contoh program 13, jika variabel yang di cek bernilai 0, fungsi *filter\_var()* akan menganggap variabel tersebut bukan merupakan integer. Untuk mengatasinya dapat dilakukan dengan cara seperti pada *Box Code 14*.

<?php

$int = 0;

**if** (filter\_var($int, FILTER\_VALIDATE\_INT) === 0 || !filter\_var($int, FILTER\_VALIDATE\_INT) === false) {

**echo**("Integer valid");

} **else** {

**echo**("Integer tidak valid");

}

?>

*Box Code 14 – memeriksa angka 0 dengan filter\_list()*

**

*Gambar 3.3 output box code 14.*

* ***Validate(memvalidasi) IP Address***

Berikut adalah contoh penggunaan fungsi *filter\_var()* untuk memeriksa kevalidan sebuah alamat IP, penerapannya dapat dilihat pada *Box Code 15*.

<?php

$ip = "127.0.0.1";

**if** (!filter\_var($ip, FILTER\_VALIDATE\_IP) === false) {

**echo**("$ip merupakan IP address yang valid");

} **else** {

**echo**("$ip merupakan IP address yang tidak valid");

}

?>

*Box Code 15 – memeriksa kevalidan sebuah IP dring*

**

*Gambar 3.4 output box code 15*

* ***Sanitize & validate alamat email***

Berikut ini adalah contoh penggunaan fungsi *filter\_var()* untuk menghapus semua karakter ilegal dari variabel yang harusnya berisi sebuah alamat email, setelah menghilangkan semua karakter ilegal, fungsi juga akan memeriksa pakah variabel tersebut bernilai alamat email yang valid atau tidak.

<?php

$email = "dimasrizkyhp@it.student.pens.ac.id";

// Menghapus seluruh karakter ilegal pada string email

$email = filter\_var($email, FILTER\_SANITIZE\_EMAIL);

// Validasi email

**if** (!filter\_var($email, FILTER\_VALIDATE\_EMAIL) === false) {

**echo**("$email merupakan alamat email yang valid");

} **else** {

**echo**("$email bukan alamat email yang valid");

}

?>

*Box Code 16 – Membersihkan dan memeriksa kevalidan alamat email*

**

*Gambar 3.5 output box code 16*

* ***Sanitize & validate sebuah URL***

Contoh berikut ini adalah kegunaan fungsi *filter\_var()* untuk membersihkan karakter ilegal dari sebuah URL lalu melakukan validasi setelah pembersihan selesai.

<?php

$url = "https://www.w3schools.com";

// Menghapus seluruh karakter ilegal pada string URL

$url = filter\_var($url, FILTER\_SANITIZE\_URL);

// validasi URL

**if** (!filter\_var($url, FILTER\_VALIDATE\_URL) === false) {

**echo**("$url merupakan URL yang valid");

} **else** {

**echo**("$url bukan merupakan URL yang valid");

}

?>

*Box Code 17 – Membersihkan dan memeriksa kevalidan sebuah URL*

**

*Gambar 3.6 output box code 17*

***PHP Advanced Filter***

* ***Memvalidasi sebuah integer dalam sebuah range***

Fungsi *filter\_var()* juga dapat digunakan untuk memeriksa apakah nilai sebuah variabel integer berada diantara jarak angka. Penerapannya dapat diliha pada *Box Code 18.*

<?php

$int = 122;

$min = 1;

$max = 200;

**if** (filter\_var($int, FILTER\_VALIDATE\_INT, **array**("options" => **array**("min\_range"=>$min, "max\_range"=>$max))) === false) {

**echo**("Nilai variabel : ".$int." ,tidak berada di jangkauan (".$min."-".$max.")");

} **else** {

**echo**("Nilai variabel : ".$int." ,berada di jangkauan (".$min."-".$max.")");

}

?>

*Box Code 18 – Cek integer dalam sebuah range angka*

**

*Gambar 4.0 output box code 18*

* ***Memvalidasi alamat Ipv6***

Fungsi *filter\_var()* juga dapat digunakan untuk melakukan validasi string untuk format alamat Ipv6. Penerapannya dapat dilihat pada *Box Code 19*.

<?php

$ip = "2001:0db8:85a3:08d3:1319:8a2e:0370:7334";

**if** (!filter\_var($ip, FILTER\_VALIDATE\_IP, FILTER\_FLAG\_IPV6) === false) {

**echo**("$ip merupakan IPv6 yang valid");

} **else** {

**echo**("$ip bukan merupakan alamat IPv6 yang valid");

}

?>

*Box Code 19 – Validasi alamat Ipv6*

**

*Gambar 4.1 output box code 19*

* ***Memvalidasi URL (Harus berisi querystring)***

Contoh pada *Box Code 20* di bawah ini merupakan penerapan fungsi *filter\_var()* untuk memeriksa URL, apakah URL tersebut dilengkapi dengan querystring atau tidak.

<?php

$url = "https://www.w3schools.com";

**if** (!filter\_var($url, FILTER\_VALIDATE\_URL, FILTER\_FLAG\_QUERY\_REQUIRED) === false) {

**echo**("$url merupakan URL yang valid");

} **else** {

**echo**("$url bukan merupakan URL yang valid");

}

?>

*Box Code 20 – Validasi querystring pada URL*

**

*Gambar 4.2 output box code 20*

* ***Menghapus karakter ASCII tertentu***

Contoh pada *Box Code 21* di bawah ini merupakan penerapan fungsi *filter\_var() untuk memeriksa dan membersihkan seluruh tag HTML yang kode ASCII nya bernilai di atas 127, dari sebuah string.*

<?php

$str = "<h1>Hello WorldÆ�Å!</h1>";

**echo** ("Tanpa filter : </br>");

**echo** $str."</br>";

**echo** ("Dengan filter : </br>");

$newstr = filter\_var($str, FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_STRIP\_HIGH);

**echo** $newstr;

?>

*Box Code 21 – Validasi sebuah string bergantung dengan nilai ASCII nya*

**

*Gambar 4.3 output box code 21.*

***PHP Predefined Variables***

PHP menyediakan banyak *predefined variable*. Variabel variabel tersebut berbentuk array yang dapat diakses oleh seluruh skrip, disebut dengan superglobals.

* ***PHP Superglobals***
  + $GLOBAL, berisi reference untuk setiap variabel yang memiliki scope global di dalam skript. Key pada array ini adalah nama dari global variabel yang ada.
  + $\_SERVER, merupakan array berisi informasi seperti headers, path, lokasi skrip. Variabel ini dibuat oleh web server namun tidak selalu semua informasi disediakan oleh sebuah web server
  + $\_GET, berisi variabel yang di pass menggunakan metode GET
  + $\_POST sama dengan array, merupakan array berisi variabel yang di pass menggunakan metode POST
  + $\_FILES, array yang berisi item yang di unggah menggunakan script yang sedang berjalan melalui metode HTTP POST
  + $\_REQUEST, array yang berisi konten dari GET, \_POST, dan $\_COOKIE
  + $\_COOKIE, merupakan array yang berisi variabel yang di pass ke script yang sedang berjalan menggunakan HTTP cookies
  + $SESSION, array yang berisi variabel session yang sedang berjalan atau tersedia dalam skrip yang sedang berjalan.
  + $PHP\_SELF, berisi nama file script php yang berjalan
  + $PHP\_ERRORMSG, merupakan variabel yang berisi text pesan error yang dihasilkan oleh PHP
* ***Variabel $\_SERVER***
  + $\_SERVER[‘PHP\_SELF’], merupakan nama file skrip yang sedang dieksekusi, relatif dengan document rootnya
  + $\_SERVER[‘argv’], merupakan array dari argumen yang di pass ke script yang sedang berjalan.
  + $\_SERVER[‘argc’], berisi beberapa paramater yang di pass ke sbuah script jika script dieksekusi melalui command line.
  + $\_SERVER[‘GATEWAY\_INTERFACE’], version dari spesifikasi CGI dari server yang sedang berjalan.
  + $\_SERVER[‘SERVER\_ADDR’], merupakan alamat IP dari server yang mengeksekusi script yang sedang berjalan.
  + $\_SERVER[‘SERVER\_NAME’], nama host dari server yang mengeksekusi script. Jika script berjalan pada virtual host, variabel ini akan didefinisikan sesuai dengan virtual host mendefinisikan variabel ini.

***PHP Regular Expressions***

Regular expression tidak lain hanyalah rentetan pola dari karakter sebuah string atau variabel. Reqgular expression memungkinkan fungsionalitas pencocokan pola untuk sebuah strin. Dengan menggunakan regular expression programmer dapat mencari sebuah bagian string tertentu lalu dapat menggantinya dengan string lainnya atau dapat memisah-misahkannya kebeberapa potongan.

* ***POSIX Regular Expression***

Struktur dari POSIX Regex tidak jauh berbeda dengan expresi aritmatika, yaitu beberapa operator dikombinasikan untuk membentuk expression yang lebih complex. Regex paling sederhana adalah regex yang mencocokan suatu karakter.

**Bracket** [], mempunyai makna khusus ketika digunakan pada konteks regex. Bracket digunakan untuk menemukan range sebuah karakter, contohnya

* + **[0-9]**, mencocokan semua digit decimal dari 0 sampai 9
  + **[a-z]**, mencocokan seluruh karakter alfabet lower case dari ‘a’ sampai dengan ‘z’
  + **[A-Z]**, mencocokan seluruh karakter alfabet upper case dari ‘A’ sampai dengan ‘Z’
  + **[a-Z]**, mencocokan seluruh karakter alfabet mulai dari lower case ‘a’ sampai dengan upper case ‘Z’

**Quantifiers*,*** posisi dari karakter di dalam bracket dapat dilambangkan menggunakan special char. Setiap special char memiliki konotasi berbeda beda. Contoh dari special char tersebut adalah ‘+’, ‘\*’, ‘?’, {int range}, dan ‘$’. Contoh penggunaannya adalah sebagai berikut,

* + **P+**, mencocokan seluruh string yang paling tidak memiliki karakter ‘p’
  + **P\***, mencocokan seluruh string memiliki nol atau lebih karakter ‘p’
  + **P?**, mencocokan seluruh string yang berisi nol atau lebih karakter ‘p’ (alternatif penulisan untuk p\*)
  + **P{N}**, mencocokan string yang berisi rangkaian karakter ‘p’ sebanyak N
  + **P{2,3}**, mencocokan string yang berisi rangkaian karakter ‘p’ sebanyak 2 atau 3 kali
  + **P{2,}**, mencocokan string yang berisi paling tidak rangkaian karakter ‘p’ sebanyak 2 kali
  + **P$**, mencocokan string yang diakhiri dengan karakter ‘p’
  + **^P**, mencocokan string yang diawali karakter ‘p’.
* ***PHP Regexp POSIX Function***

PHP menyediakan tujuh fungsi untuk memindai string menggunakan POSIX-style regex, yaitu fungsi,

* + Ereg(), mencari string berdasarkan dengan pola yang berbentuk string, akan mengembalikan true jika pola ditemukan, false jika sebaliknya
  + Ereg\_replace() mencari string berdasarkan pola dan mengubahnya dengan string pengganti jika ditemukan
  + Eregi(), mencari sepanjang string berdasarkan pola namun dalam kondisi tidak case sensitive
  + Eregi\_replace(), identik dengan ereg\_replace() namun bedanya, eregi\_replace() bersifat case insensitive
  + Split(), akan memisahkan string ke beberapa elemen, batas dari masing masing elemen yang terbentuk sesuai dengan pola yang ada pada sebuah string
  + Spliti(), berfungsi sama seperti split() hanya saja bersifat case insensitive
  + Sql\_regcase(), erfungsi sebagai utilitas, mengkonversi setiap karakter pada input parameter menjadi expression di dalam kurung yang berisi dua buah karakter
* ***PERL Style Regular Expression***

PERL-style regex mirip dengan POSIX, bahkan syntax POSIX hampir dapat seluruhnya dieksekusi pada fungsi PERL-style. Dalam PERL regex terdapat konsep meta char. Meta character adalah karakter alfabet yang diawali dengan backslash untuk mendefinisikan kombinasi karakter tersebut memiliki fungsi khusus. Contohnya adalah dengan menggunakan meta karakter ‘\d’ untuk mencari angka dalam sejumlah variabel. Contoh : /[\d]+000/, maksudnya, \d akan mencari seluruh string yang merupakan karakter numerik.

Beberapa meta karakter di bawah ini digunakan pada PERL-style regex,

* + ‘.’ – karakter tunggal
  + ‘\s’ – karakter whitespace (spasi, tabulasi, baris baru)
  + ‘\S’ – karakter non-whitespace
  + ‘\d’ – sebuah digit (0-9)
  + ‘\D’ – non digit karakter
  + ‘\w’ - karakter kata (word char) (a-z, A-Z, 0-9, \_)
  + ‘\W’ – non word char
  + [aeiou] – karakter tunggal dalam set tertentu
  + [^aeiou] – karakter tunggal selain dari set yang ditentukan
  + (foo|bar|baz) – mencocokan menggunakan beberapa alternatif yang telah dispesifikan.

* ***PHP Regexp PERL Compatible Function***

PHP juga menyediakan beberapa fungsi untuk mencari sebuah string menggunakan PERL regex. Antara lain,

* + **Preg\_match()**, mencari string berdasarkan sebuah pola, mengembalikan true jika pola tersebut ditemukan, false jika sebaliknya
  + **Preg\_match\_all()**, mencocokan seluruh pola yang ada pada sebuah string
  + **Preg\_replace()**, fungsinya sama dengan fungsi ereg\_replace, bedanya regex dapat digunakan pada pola dan variabel replacementnya
  + **Preg\_split()**, berfungsi seperti fungsi split, bedanya regex juga dapat digunakan sebagai parameter polanya
  + **Preg\_grep()**, mencari seluruh elemet dalam input\_array, lalu mengembalikan seluruh element yang cocok dengan pola regex
  + **Preg\_quote()**, mengutip regex karakter.

***PHP Error Exception Handling***

Error handling merupakan proses untuk mengatasi error yang terjadi pada saat eksekusi script. Ketika error tidak diatasi terlebih dahulu, akan mengakibatkan error yang berkepanjangan. PHP menyediakan beberapa fitur untuk melakukan handling error.

* ***Fungsi die()***

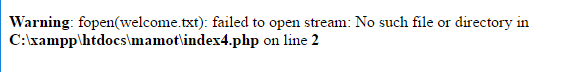
Fungsi *die()* digunakan untuk mengganti error message yang dihasilkan oleh error yang terjadi pada script. Diasumsikan terdapat fungsi untuk membuka sebuah file seperti yang ada pada *Box Code 22,* jika file yang dibuka tidak ditemukan, maka script PHP akan memunculkan pesan error seperti pada *gambar 7.0.*

<?php

$file=fopen("welcome.txt","r");

?>

*Box Code 22 – membuka file tanpa file*

**

*Gambar 7.0 error message yang diberikan*

Untuk mencegah error message yang seperti itu, dapat menggunakan fungsi*die()* untuk membuat error message custom dengan cara menangkap error yang terjadi terlebih dahulu. Penerapannya dapat dilihat pada *Box Code 23,* dan hasil error message nya dapat dilihat pada *gambar 7.1.* Namun, tentu untuk mengatasi error tidak hanya cukup menghentikan jalannya script.

<?php

**if**(!file\_exists("welcome.txt")) {

**die**("File not found");

} **else** {

  $file=fopen("welcome.txt","r");

}

?>

*Box Code 23 – error handling menggunakan fungsi die()*

**

*Gambar 7.1 error message hasil fungsi die()*

* ***Custom Error Handling***

Untuk membuat error handler custom sangat mudah dilakukan di PHP. Cukup dengan membuat sebuah fungsi yang dapat dipanggil ketika error terjadi pada PHP. Fungsi ini harus dapat meng-handle setidaknya dua parameter(error level, dan error message), dari lima paramter yang ada (opsional : file, line-number, error\_context),

* + Error\_level, diperlukan, isinya adalah menspesifikan error report level untuk error yang didefinisikan secara kustom oleh user. Harus berbentuk angka
  + Error\_message, diperlukan, untuk menspesifikan error message yang akan muncul
  + Error\_file, opsional, menspesifikan filename dimana error terjadi
  + Error\_line, opsional, menspesifikan urutan baris keberapa error terjadi
  + Error\_context, opsional, menspesifikan array yang berisi setiap variabel dan nilainya yang digunakan pada saat error terjadi.

Error level memiliki tingkatan yang sudah didefinisikan oleh PHP. Ada beberapa tipe error yang dapat digunakan, yaitu,

* + 2, E\_WARNING, error tidak fatal yang terjadi pada saat runtime, eksekusi script tidak dihentikan
  + 8, E\_NOTICE, pemberitahuan pada saat runtime, ter-trigger jika menemukan sesuatu yang mungkin akan menyebabkan error.
  + 256, E\_USER\_ERROR, fatal error terjadi, namun bukan error dari definisi default PHP, melainkan error yang didefinisikan oleh programmer menggunakan fungsi *trigger\_error()*
  + 512, E\_USER\_WARNING, non-fatal warning, warning terjadi namun bukan dari definisi warning default dari skrip PHP, melainkan definisi warning dari user.
  + 1024, E\_USER\_NOTICE, merupakan notice yang berasal dari definisi user.
  + 4096, E\_RECOVERABLE\_ERROR, fatal error yang dapat ditangkap, sama dengan E\_ERROR namun ditangkap karena definisi handler dari user
  + 8192, E\_ALL, akan menangkap seluruh error dan warning yang terjadi

Contoh pembuatan error handler dapat dilihat pada *Box Code 25,* kode teresebut merupakan error handling sederhana yang akan dipanggil ketika error terjadi.

<?php

**function** customError($errno, $errstr) {

**echo** "<b>Error:</b> [$errno] $errstr<br>";

**echo** "Ending Script";

**die**();

}

?>

*Box Code 24 – Custom Error Handling*

Untuk mengaktifkan custom error handling yang telah dibuat, dapat menggunakan fungsi *set\_error\_handler(“customError”).* Penerapannya dapat dilihat pada *Box Code 25,* dan hasil error messagenya dapat dilihat pada *gambar 7.2*

<?php

//error handler function

**function** customError($errno, $errstr) {

**echo** "<b>Error:</b> [$errno] $errstr";

}

//set error handler

set\_error\_handler("customError");

//trigger error

**echo**($test);

?>

*Box Code 25 – fungsi set\_error\_handler*

**

*Gambar 7.2 error message dari custom error handling*

* ***Exception Handling***

Untuk masalaha exception handling, PHP memiliki model yang sama dengan bahasa pemrograman lainnya. Exception sangatlah penting dan dapat melakukan error handling lebih baik. Ada beberapa keyword pada exception, antara lain,

* + Try, merupakan blok kode dimana jika try ter-trigger, maka system akan melakukan “throw” sebuah exception, jika tidak ter-trigger, maka kode di dalam blok try akan dilanjutkan
  + Throw, cara untuk mentrigger sebuah exception. Setiap throw selalu dilengkapi satu “catch”
  + Catch, adalah blok yang menerima sebuah exception yang lalu membuat object yang berisi informasi exception.

Ketika sebuah exception di throw, kode pada blok try tidak akan dieksekusi secara lanjut, dan PHP akan mencoba menemukan catch yang sesuai untuk exception yang ter-grigger. Jika exception tersebut tidak ada yang dapat meng-handle, maka fatal error akan ditampilkan oleh PHP dengan kalimat “Uncaught Exception...”.

Sebagai contohnya, dapat dilihat pada *Box Code 26*, kode tersebut akan melakukan throw exception dan akan menjalankan blok code yang ada pada block catch.

<?php

**try** {

 $error = 'Selalu throw error ini';

**throw** **new** **Exception**($error);

 // Blok kode berikut ini tidak akan dieksekusi

**echo** 'Tidak akan dieksekusi';

 }**catch** (**Exception** $e) {

**echo** 'Caught exception: ', $e->getMessage(), "\n";

 }

 // Melanjutkan eksekusi

**echo** 'Hello World';

?>

*Box Code 26 – Exception Handling*

* ***Custom Exception Handler***

PHP juga memberikan keleluasaan terhadap exception handler, menggunakan fungsi *set\_exception\_handler()*. Conohnya dapat dilihat pada *Box Code 27.*

<?php

**function** exception\_handler($exception) {

**echo** "Uncaught exception: " , $exception->getMessage(), "\n";

 }

 set\_exception\_handler('exception\_handler');

**throw** **new** **Exception**('Uncaught Exception');

**echo** "Tidak dieksekusi\n";

?>

*Box Code 27 – Custom Exception Handling*