Validation

```php
<?php
// VALIDATION FORM ************************************************************
// after form operation you should validate the data for security purposes
// js could validate form but, js could be disabled from web borwser so that server side is required
// for example you may get the messsage from text input and want to display to user back
// you should avoid html injection. Because there could be attack with JS

// String validation
// while displaying the message you should use:
// it will not execute js
// <input type="text" name="numPerson" size="13" value="<?= isset($num) ? filter_var($num, FILTER_SANITIZE_FULL_SPECIAL_CHARS)

// if (filter_var($str, FILTER_SANITIZE_STRING) === false)

// Number validation
// $options = array('options' => array('min_range' => 0, 'max_range => 10'));
// if (filter_var($age, FILTER_VALIDATE_INT) === false)
// if (filter_var($age, FILTER_VALIDATE_INT, $options) === false)
// if (filter_var($age, FILTER_VALIDATE_FLOAT, $options) === false)

// === is used because 0, "", [] return false
```

Regular Expression

```php
    // REGULAR EXPRESSION *************************************************
    // checking characters in string

    // /regexExpression/flags
    // flags:
    //      i -> case insensitive
    //      m -> multiline
    //      u -> unicode

    // return 0 1


    // checks if 7 is included
    // preg_match('/7/', $id)

    // for special character you must use escape character
    // preg_match('/ \n /', $id)

    // preg_match('/ \.\.\. /', $id) -> ...
    // preg_match('/ ... /', $id) -> 3 any character
    // preg_match('/ a...t /', $id) -> a123t , ahmet

    // [] - ^
    // preg_match('/ [0-9] /', $id) -> one digit
    // preg_match('/ [^0-9] /', $id) -> not include one digit

    // preg_match('/ [a-zA-Z] /', $id) -> one character
    // preg_match('/ [^a-zA-Z] /', $id) -> not one character


    // preg_match('/ [a-zA-Z0-9_] /', $id) ->one word char
    // preg_match('/ \w /', $id) -> one word char     for not \W

```

```
68
69      // preg_match('/[ \t\n]/', $id) -> any blank char
70      // preg_match('/\s/', $id) -> any blank char     for not \S
71
72      // preg_match('/ \d\dahmet\d\d /i', $id) -> 24AHMET24, 99ahmet00
73
74
75      // preg_match('/ \bahmo\b /', $id) -> 123 ahmo 123     false -> 2ahmo    \b is boundry
76
77
78      // preg_match('/\.jpg$/', $id) end with .jpg
79
80
81      // ? means optional
82      // preg_match('/\d?/', $str) it means there could be digit or not
83
84      // checks if there is digit
85      // preg_match('/\d/', $id)
86
87      // {1} -> 1 times
88      // {1,4} -> min 1 times max 4 times
89      // {1,} -> min 1 times max infinite
90      // {,4} -> optional to 4
91
92      // +  -> 1 to infinite
93
94      // preg_match('/ \d+ /', $id) 1   112312312312
95
96      // *  -> 0 to infinite
```

```
98
99      // checks if there is 7 digits
100     // preg_match('/\d{7}/', $id)
101
102     // checks if there start and 7 digit and end
103     // preg_match('/^\d{7}$/', $id)
104
105     // check if there any one to three characters and not blank in it
106     // preg_match('/^\S{1,3}$/', $password)
107
108     // grouping with capture and not
109     // preg_match('/^(\d\d)$/', $num) 2 digit also I capture so that 1 can use that pattern again
110     // preg_match('/^(\d\d) \1$/', $num) 24 24 pattern should be same!
111
112     // preg_match('/^(?:\d\d) \1$/', $num) if you use ?: in grouping you not capture that group
113     // so that you cannot use it again with \1
114
115     // preg_match('/^(?:cat|dog)eat(meat|fist)\1$/', $num) if you use ?: in grouping you can use or
116     // preg_match('/^(?:cat|dog) eat (meat|fist)\1$/', $num) cat eat meat meat | dog eat meat meat
117
```

Match All

```
119     // MATCH ALL ---------------------
120     // preg_match_all('expression','text',$array)
121     // result will be in $array it is 2 dim array 0 row keep all match
122
123     // if there is group match it will stored in grouped index
124     // preg_match_all('/(\d\d)a/', '24ahmet', $ar);
125     // echo "{$ar[0][0]}"; 24a
126     // echo "{$ar[1][0]}"; 24
127
128     // preg_match_all('/(\w+)@hotmail/', 'ahmet@hotmail.com tuna@hotmail.edu.tr', $ar);
129     // echo "{$ar[0][0]} "; ahmet@hotmail
130     // echo "{$ar[0][1]} "; tuna@hotmail
131     // echo "{$ar[1][0]} "; ahmet
```

replace

```
135     // REPLACE ------------
136     // $modified = preg_replace('/pattern/', 'replaced pattern', 'text');
137     // $modified = preg_replace('/like/', 'hate', 'ahmo like that life');
138     // echo("$modified"); ahmo hate that life
139
140     // also you may use group for output
141     // $modified = preg_replace('/(\w+ +)(\w)(\w+)$/iu', '-\1- -\2- -\3-', 'Ahmet Oguz Ergin');
142     // echo "$modified";
143
```

Db

C: > wamp64 > www > AhmetOguzErgin > Web > mySolutions > not2 > Sql > DatabaseConnection.php > ...

```php
<?php

$databaseName = "test";
$user = "std";
$pass = "";

// create data source name and connect to database
$dsn = "mysql:host=localhost;dbname=$databaseName;charset=utf8mb4";
try {
    $db = new PDO($dsn, $user, $pass);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (Exception $ex) {
    die("DB Connect Error");
}

```

Create

```php
if ($btn == "Create") {

    try {
        // 1. placeholder
        // $sql = "insert into user (username, email, birthday) values (?, ?, ?)";
        // $stmt = $db->prepare($sql); // PDOStatement
        // $stmt->execute(["Jim", "jim@hotmail.com", "1990-05-15"]);
        // echo "<p> Jim is added into user table</p>";
        // echo "<p> Insert ID : ", $db->lastInsertId(), "</p>";
        // $newUserId = $db->lastInsertId();

        // 2. by associate array and namespace
        // $user = [
        //     "username" => "John",
        //     "email" => "john@matrix.com",
        //     "birthday" => "1993-05-30"
        // ];
        // $sql = "insert into user (username, email, birthday) values (:username, :email, :birthday)";
        // $stmt = $db->prepare($sql); // PDOStatement
        // $stmt->execute($user);
        // echo "<p>" . $user["username"] . " is added into user table</p>";
        // echo "<p> Insert ID : ", $db->lastInsertId(), "</p>";
        // $newUserId = $db->lastInsertId();


        $sql = "insert into crudoperations (name, code) values (:name, :code)";
        $stmt = $db->prepare($sql);
        $stmt->bindValue(":name", $name, PDO::PARAM_STR);
        $stmt->bindValue(":code", $code, PDO::PARAM_STR);
        $stmt->execute();
        $lastInsertedId = $db->lastInsertId();
    } catch (PDOException $ex) {
        die("<p>Insert Error : " . $ex->getMessage());
    }
}
```

Update

```php
// update
if ($btn == "Update") {
    try {
        // $user = [
        //     "user_id" => 3,
        //     "username" => "John Peter",
        //     "birthday" => "1995-4-4"
        // ];
        // $sql = "update user set username = :username, birthday = :birthday where user_id = :user_id";
        // $stmt = $db->prepare($sql);
        // $stmt->execute($user);


        $sql = "update crudoperations set name = :name, code = :code where id = :id";
        $stmt = $db->prepare($sql);
        $stmt->bindValue(":name", $name, PDO::PARAM_STR);
        $stmt->bindValue(":code", $code, PDO::PARAM_STR);
        $stmt->bindValue(":id", $id, PDO::PARAM_INT);
        $stmt->execute();
        // $updatedRowCount = $stmt->rowCount();
    } catch (PDOException $ex) {
        die("<p>Update Error : " . $ex->getMessage());
    }
}
// ********************************************************************************
```

Delete

```php
// ********************************************************************************
// delete
if ($btn == "Delete") {
    try {

        $sql = "delete from crudoperations where id = :id";
        $stmt = $db->prepare($sql);
        $stmt->bindValue(":id", $id, PDO::PARAM_INT);
        $stmt->execute();
        // $deletedRowCount = $stmt->rowCount();
    } catch (PDOException $ex) {
        die("<p>Update Error : " . $ex->getMessage());
    }
}
// ********************************************************************************
```

Login

```php
// ********************************************************************************************
// login
if ($btn == "Login") {
    try {

        $sql = "select * from crudoperations where id = :id and code = :code";
        $stmt = $db->prepare($sql);
        $stmt->bindValue(":id", $id, PDO::PARAM_INT);
        $stmt->bindValue(":code", $code, PDO::PARAM_STR);
        $stmt->execute();
    } catch (PDOException $ex) {
        die("<p>Update Error : " . $ex->getMessage());
    }
    // if ($stmt->rowCount() != 0)
    //     var_dump("logged in");
    // else
    //     var_dump("wrong password");
}
// ********************************************************************************************
```

Read

```php
// display/read all
try {
    // $sql = "select * from crudoperations order by name asc LIMIT $start , $end";
    // $sql = "select * from crudoperations where name LIKE 'ah%' order by name asc LIMIT $start , $end";
    $sql = "select * from crudoperations";
    $stmt = $db->prepare($sql);
    $stmt->execute();
    $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
    // $stmt->rowCount();
} catch (Exception $ex) {
    die("Query Error : " . $ex->getMessage());
}
DisplayAll($stmt, $rows);
// ********************************************************************************

// ********************************************************************************
// display/read join
try {
    $sql = "select * from crudoperations inner join crudoperationsforeigntable on crudoperations.id = crudoperationsforeigntable.referencedId";
    $stmt = $db->prepare($sql);
    $stmt->execute();
    $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
    // $stmt->rowCount();
} catch (Exception $ex) {
    die("Query Error : " . $ex->getMessage());
}
DisplayJoin($stmt, $rows);
// ********************************************************************************
```

```php
// ********************************************************************************************
// display/read search 1 row
$id = 1;
try {
    $sql = "select * from crudoperations where id = :id";
    $stmt = $db->prepare($sql);
    $stmt->bindValue(":id", $id, PDO::PARAM_INT);
    // $stmt->bindValue(":id", $id, PDO::PARAM_STR);
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
} catch (Exception $ex) {
    die("Query Error : " . $ex->getMessage());
}
DisplayRow($stmt, $row);
// ********************************************************************************************
```

```
<!-- add fk -->
<!-- ALTER TABLE `comments`
  ADD CONSTRAINT `comment_user_fk` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`) ON DELETE CASCADE ON UPDATE CASCADE;
COMMIT; -->

<!-- LIMIT 0,1   from 0. index 1 row-->
<!-- LIMIT 0,2   from 0. index 2 row-->
<!-- LIMIT 1     from 0. index 1 row-->
<!-- LIMIT 3     from 0. index 3 row-->
<!-- LIMIT 2,2   from 2. index 2 row-->
```

1

```php
<?php
class User
{
    // constant and static variables
    const MALE = 1;
    const FEMALE = 2;
    private static $count = 0;

    // private object variables
    private $username;
    private $gender;

    // constructor
    function __construct($username, $gender)
    {
        $this->username = $username;
        $gender == self::MALE ? $this->gender = self::MALE : $this->gender = self::FEMALE;

        self::$count++;
    }

    // destructor is called at the end of the program to remove objects
    function __destruct()
    {
        echo"</br></br>Object destructed at the end of the program for :</br>", $this->__toString();
    }

    // static method
    public static function getCount()
    {
        return self::$count;
    }

    // toString method
    public function __toString()
    {
        // echo "Username: {$this->username} </br>Gender: " . $this->gender == self::MALE ? "Male" : "Female" . "</br>";
        echo "Username: {$this->username} </br>Gender: " . ($this->gender == self::MALE ? "Male" : "Female") . "</br></br>";
    }
}
```

```php
<?php
// general about classes

require_once __DIR__ . "/Classes/User.php";

// creation of objects
$user1 = new User("Ahmet Oğuz Ergin", User::MALE);
$user2 = new User("Kisimo", User::FEMALE);

// display method
$user1->__toString();
$user2->__toString();

// static display
echo "There is " . User::getCount() . " object ";
?>
```

2

```php
class Vehicle
{
    private static $count = 0;
    private $seat;

    function __construct($seat)
    {
        $this->seat = $seat;
        self::$count++;
    }

    public function __toString()
    {
        return "Seat number: " . $this->seat . "</br>";
    }

    public static function getCount()
    {
        return self::$count;
    }
}
```

```php
<?php
class Motorbike extends Vehicle
{
    private $fuel;
    private $speed;

    public function __construct($seat, $fuel, $speed)
    {
        parent::__construct($seat);
        $this->fuel = $fuel;
        $this->speed = $speed;
    }

    public function __toString()
    {
        $output = parent::__toString();
        $output .= "Fuel Type: {$this->fuel}</br>";
        $output .= "Speed: {$this->speed}</br></br>";
        return $output;
    }
}
```

```php
// Inheritance and polymorphism autoload

// require_once __DIR__ . "/Classes/Car.php";
// require_once __DIR__ . "/Classes/Motorbike.php";
// you can use that statement to include classes automatically
spl_autoload_register(function ($class) {
    echo "Class included: $class <br>";
    require_once __DIR__ . "/Classes/" . $class . ".php";
});


$car1 = new Car(5, "hybrid");
$motorbike1 = new Motorbike(5, "gas", 320);

$ar = [$car1, $motorbike1];

// polymorphism
foreach ($ar as $obj) {
    if ($obj instanceof Car)
        echo "Car: </br>";
    else
        echo "Motorbike:</br>";

    echo $obj->__toString();
}

echo "There is " . Vehicle::getCount() . " object ";

?>
```

3

```php
<?php

namespace Classes;

class Vehicle
{
    private static $count = 0;
    private $seat;
```

```php
<?php
namespace Classes;
class Motorbike extends Vehicle
{
    private $fuel;
    private $speed;
```

```php
<?php
// namespace is used for ambiguity but I think there is no need.

// require_once __DIR__ . "/Classes/Car.php";
// require_once __DIR__ . "/Classes/Motorbike.php";
// you can use that statement to include classes automatically
spl_autoload_register(function ($class) {
    echo "$class <br>";
    require_once "./" . $class . ".php";
});

use Classes\Car as C;
use Classes\Vehicle as V;
use Classes\Motorbike as M;

$car1 = new C(5, "hybrid");
```

4

```php
<?php

spl_autoload_register(function ($class) {
    echo "$class <br>";
    require_once $class;
});

// require_once __DIR__ . "/Tool.php";
// require_once __DIR__ . "/iDraw.php";

class Pen extends Tool implements iDraw
{
    private $amount;

    public function __construct($aim, $size, $am
    {
        parent::__construct($aim, $size);
        $this->amount = $amount;
    }

    public function __toString()
    {
        return parent::__toString() . "</br>Amou
    }

    public function iDraw()
    {
        echo "This pen class implements iDraw interface";
    }
```

```php
<?php
interface iDraw
{
    public function iDraw();
}
```

```php
<?php
// aim is to show implements interface and extend

spl_autoload_register(function ($class) {
    echo "$class <br>";
    require_once __DIR__ . "/Classes/" . $class . ".php";
});

$pen1 = new Pen("Draw", "small", 5);

echo $pen1->__toString();

$pen1->iDraw();
?>
```