PHP – PHP Hypertext Preprocessor

Igor Wojnicki

Katedra Informatyki Stosowanej
Akademia Górniczo-Hutnicza w Krakowie

17 listopada 2022

Spis Treści I

Spis Treści

Jak się dostać do czegoś w Internecie

- Odczytać jak zwykły plik: HTTP wrapper (domyślnie GET).
- Użyć buiblioteki cURL.

Odczyt zasobów, fopen()

```
$h=fopen('http://www.google.com','r');
if ($h) {
  while (($bufor=fgets($h))!=false) {
    echo $bufor;
  }
  fclose($h);
}
```

Odczyt zasobów, file() i inni

Założenie: pod adresem `http://home.agh.edu.pl/~wojnicki/temp.php` znajduje się poniższy kod:

```
<?php
  $komunikat="temperatura\n21";
  echo $komunikat;
?>
```

Przykładowe przetwarzanie liniami:

```
<?php
$f=file('http://home.agh.edu.pl/~wojnicki/temp.php');
if ($f) {
  foreach ($f as $line_num => $line) {
    echo "Linia #<b>{$line_num}</b> : " .
         htmlspecialchars($line) .  "<br />\n";
  }
}
?>
```

Odczyt zasobów, file_get_contents()

Podobnie jak `file()`, ale zwraca łańcuch znaków.

```
echo file_get_contents(
      'http://home.agh.edu.pl/~wojnicki/temp.php');
```

Przykład przetwarzania danych JSON

Przetwarzanie listy.

```
$dbs=file_get_contents(
      'http://awing.kis.agh.edu.pl:5984/_all_dbs');
if ($dbs) {
  $dbs_tab=json_decode($dbs,true);
  foreach ($dbs_tab as $nazwa) {
    echo $nazwa . '<br/>';
  }
}
```

Odczyt zasobów, cURL, GET

```
$c=curl_init('http://awing.kis.agh.edu.pl:5984');
/* zwroc dane jako wartosc, a nie stdout */
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
$dane=curl_exec($c);
if ($dane) {
  curl_close($c);
  echo $dane;
}
```

Odczyt zasobów, cURL, POST

```
$dok='{"tresc":"zawartosc dokumentu",
       "autor":"Wojnicki"}';
$c=curl_init('http://awing.kis.agh.edu.pl:5984/moja1');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_POST, 1);
curl_setopt($c, CURLOPT_HTTPHEADER,
  array('Content-type: application/json'));
curl_setopt($c, CURLOPT_POSTFIELDS, $dok);
$dane=curl_exec($c);

if ($dane) {
  curl_close($c);
  echo $dane;
}
```

Powłoka dla CouchDB

Input

URI:

awing.kis.agh.edu.pl:5984

Method: GET POST PUT DELETE

Content type: application/json

POST/PUT:

Submit Query Reset

Output

JSON

{"couchdb":"Welcome","version":"1.0.1"}

Decoded JSON

```
stdClass Object
(
    [couchdb] => Welcome
    [version] => 1.0.1
)
```

Powłoka dla CouchDB I

```
function checked($source,$value) {
  if ($source === $value) echo 'checked="checked"';
}

$uri=$_POST['uri'];
if ($_POST['method'] === 'GET' ||
    $_POST['method'] === 'POST' ||
    $_POST['method'] === 'PUT' ||
    $_POST['method'] === 'DELETE' )
   $method=$_POST['method'];

$content_type='Content-type: '.$_POST['content'];
$post=$_POST['post'];
?>
</header>
<body>
```

Powłoka dla CouchDB II

```
<h1>Input</h1>
<p>
<form method="POST">
  URI:<br/>
  <textarea rows="4" cols="60" name="uri">
    <?php echo $uri;?>
  </textarea>
  <br/>
  Method: <input type="radio" name="method"
    value="GET" <?php checked($method,'GET')?>/> GET
  <input type="radio" name="method"
    value="POST" <?php checked($method,'POST')?>/> POST
  <input type="radio" name="method"
    value="PUT" <?php checked($method,'PUT')?>/> PUT
  <input type="radio" name="method"
    value="DELETE" <?php checked($method,'DELETE')?>/>
```

Powłoka dla CouchDB III

```
    DELETE
  <br/>
  Content type: <input type="text"
    name="content" size="40" value="application/json"/>
  <br/>
  POST/PUT:<br/>
  <textarea rows="4" cols="60" name="post">
    <?php echo stripslashes($post);?></textarea>
  <br/>
  <input type="submit"/>
  <input type="reset"/>
</form>
</p>
<h1>Output</h1>
<h2>JSON</h2>
<p>
```

Powłoka dla CouchDB IV

```
<?php
if (isset($uri)){
  $c=curl_init($uri);
  curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
  curl_setopt($c,CURLOPT_CUSTOMREQUEST,$method);
  if ($method === 'POST' || $method === 'PUT') {
    curl_setopt($c, CURLOPT_POSTFIELDS,
      stripslashes($post));
    curl_setopt($c, CURLOPT_HTTPHEADER,
      array($content_type));
  }
  $data=curl_exec($c);
  if ($data) {
    curl_close($c);
```

Powłoka dla CouchDB V

```
    echo $data;

?>


<h2>Decoded JSON</h2>
<pre>
<?php
    // true - konwersja na tablice a nie obiekt
    print_r(json_decode($data,true));
  }
?>
</pre>
<?php
}
?>
```

Powłoka dla CouchDB VI

```
</p>
```

Spis Treści

Podejście obiektowe

- Wprowadzone w wersji PHP4 (małowydajne)
- Poprawione w wersji PHP5.

Klasa

- Kolekcja własności:
  - metod,
  - zmiennych,
  - stałych.

Definicja klasy

```
<?php
class Samochod
{
    public $marka;
    public $rok_produkcji;

    public function Opis()
    {
        return $this->marka . ' z ' .
               $this->rok_produkcji . ' roku';
    }
}
?>
```


- `$this` – aktualny obiekt.
- `->` – odwołanie do składowej klasy.

Tworzenie obiektów

Operator `new`

```
$samochod = new Samochod();
$samochod->marka = 'Syrena';
$samochod->rok_produkcji = 1979;
echo $samochod->Opis();
```

Konstruktor

Funkcje magiczne (magic): `__nazwaFunkcji()`

```
class Samochod
{
    public $marka = 'brak danych';
    public function __construct($nazwa)
    {
        $this->marka=$nazwa;
    }
}
```

- Jaki będzie rezultat działania poniższego programu?

```
$s1 = new Samochod('Polonez');
echo $s1->marka;

$s2 = new Samochod();
echo $s2->marka;
```

Konstruktor, ulepszenie: wartości domyślne

```
class Samochod
{
    public $marka = 'brak danych';
    public function __construct($nazwa=null)
    {
        if (isset($nazwa)) $this->marka=$nazwa;
    }
}
```

- Jaki będzie rezultat działania poniższego programu?

```
$s1 = new Samochod('Polonez');
echo $s1->marka;

$s2 = new Samochod();
echo $s2->marka;
```

Konstruktor

Uwaga na dzedziczenie:

- Jeżeli klasa podrzędna definiuje swój konstruktor, konstruktor klasy nadrzędnej nie jest domyślnie wywoływany,
- Należy użyć: `parent::__construct()`

Destruktor

```
class Samochod
{
...
    public function __destruct()
    {
        ...
    }
}
```

Widoczność własności (składniki klas)

Private	tylko dostępne z wewnątrz klasy.
Protected	dostępne z wewnątrz klasy i klas potomnych.
Public	dostępne z wewnątrz i zewnątrz.

Samochod, raz jeszcze

- Getters, setters.

```
class Samochod {
    private $marka;
    private $rok_produkcji;
    public function Opis() {
        return $this->marka .' z ' .
               $this->rok_produkcji . ' roku';
    }
    public getMarka() { return $this->marka; }
    public setMarka($m) { $this->marka=$m; }
    public getRok() { return $this->rok; }
    public setRok($r) {
       if ($r>1900) $this->rok_produkcji=$r
       else $this->rok_produkcji=1900;
    }
```

Dziedziczenie

- Klasa nadrzędna musi być zdefiniowana przed klasą podrzędną.
- Uwaga: w tym przykładzie brak getters, setters.

```
class Zwierze {
  protected $imie;
  public function Powitanie() {
    return 'Jestem zwierzakiem, mam na imie: ' .
           $this->imie;
  }
}
class Pies extends Zwierze {
  public function Powitanie() {
    return 'Jestem psem, mam na imie: '  .
    $this->imie;
  }
}
```

Dziedziczenie, przykład

```
$zwierz=new $Zwierze();
echo $zwierz->Powitanie();

$pies=new $Pies();
echo $pies->Powitanie();
```

Klasy abstrakcyjne

- Nie można tworzyć obiektów.
- Metody również mogą być abstrakcyjne.
- Tylko abstrakcyjna klasa może posiadać abstrakcyjne metody.

```
abstract class Zwierze {
    protected $imie;
    protected $wiek;
    public function Opis()
    {
        return $this->imie . ', wiek: ' . $this->wiek;
    }
    abstract public function Powitanie();
}
```

Uwaga: w przykładzie brak getters, setters.

Dziedziczenie i klasy abstrakcyjne

```
class Pies extends Zwierze
{
    public function Powitanie()
    {
        return 'Hau, hau!';
    }
    public function Opis()
    {
        return parent::Opis() . ', jestem psem.';
    }
}
```


- Metody abstrakcyjne muszą być zdefiniowane, aby móc tworzyć obiekty.
- `parent::` – odwołanie do klasy nadrzędnej.

Stałe

```
class Osoba
{
    const DomyslneNazwisko = 'Nowak';
    ...
}

echo 'Domyślne nazwisko: ' . Osoba::DomyslneNazwisko;
```

- Stałe są zawsze dostępne publicznie.
- Przy odwołaniu nie używa się `$`

Własności statyczne I

- Własności
- Metody
- Słowo klucznowe `static`
- Nie mogą odwoływać się do własności niestatycznych.
- Istnieje tylko jedna kopia, dzielona przez wszystkie obiekty.
- Mogą być użyte bez konieczności tworzenia obiektów.
- `self::` – odwołanie do klasy bieżącej.
- `NazwaKlasy::` – odwołanie do konkretnej klasy.

Własności statyczne II

```
class Mechanika {
    public const g=9.81;
    public static function v($v0,$a,$t) {
        return $v0+$a*$t;
    }
    public static function vUpadku($t) {
        return self::v(0,self::g,$t);
    }
}

echo 'prędkość po 3 sekundach: '.
     Mechanika::vUpadku(3);
```

Klasa ostateczna

- Żadna klasa nie może po niej dziedziczyć.

```
final class SamochodMaly extends
            SamochodKompakt
{
    ...
}
```

- `final` można zastosować do metod, nie pozwalając na ich reimplementację w podklasach.

Interfejsy

- *Czyste* klasy abstrakcyjne.
- Nic nie implementują.

```
interface Wzorzec {
    public function ustaw($nazwa, $wartosc);
    public function pobierz($format);
}
class MaszynaDrukarska implements Wzorzec {
    private $dane = array();
    public function ustaw($nazwa, $wartosc) {
        $this->dane[$nazwa] = $wartosc;
    }
    public function pobierz($format)    {
        ...
    }
}
```

Klonowanie I

- Obiekty przekazywane są poprzez referencje (uwaga: tablice nie!).
- Tworzenie kopii obiektów:

```
$kopia_obiektu = clone $obiekt;
```

- Tylko płytka kopia.
- Można zdefiniować konstruktor kopiujący: `__clone()`

Klonowanie II

```
class Pojazdy {
  private $mojSamochod=array();
  public function __construct($s) {
    $this->mojSamochod[] = new Samochod($s);
  }
  public function __clone() {
    foreach ($this->mojSamochod as $i => $v)
      // $v jest nieistotne
      $this->mojSamochod[$i] =
        clone $this->mojSamochod[$i];
  }
}
$moje=new Pojazdy('Syrena');
$zony=$moje;              // te same pojazdy
$sasiada=clone $moje; // takie same pojazdy
```

Klonowanie III

Sąsiad ma *takie same* pojazdy jak ja, ale *nie te same*!

Serializacja

- `serialize()`
- `unserialize()`

```
$a = new A;
$s = serialize($a);
// przechowaj $s w pliku
file_put_contents('plik_z_obiektem_a', $s);

// w innym skrypcie:
include('klasaA.php'); /* def. klasy A */

$s = file_get_contents('plik_z_obiektem_a');
$a = unserialize($s);
```

Przykład z: `http://www.php.net`

Spis Treści

Obsługa wyjątków

`try, catch, throw, finally`

- wyjątki użytkownika klasa `Exception`
- wyjątki systemowe klasa `Error`
- `http://www.php.net/manual/en/language.exceptions.php`

Wyjątek użytkownika I

```
function inverse($x) {
    if (!$x) {
        throw new Exception('Dzielenie przez zero.');
    }
    else return 1/$x;
}
try {
    echo inverse(5) . "\n";
    echo inverse(0) . "\n";
    echo inverse(1) . ": ja nie mam szans\n"; }
catch (Exception $e) {
    echo 'Wyjątek: ', $e->getMessage(),"\n"; }
echo "itd.\n";
```

Wyjątek użytkownika II

Rezultat:

```
0.2
Wyjątek: Dzielenie przez zero.
itd.
```

Wyjątek/błąd systemowy

```
try {
    echo "Początek\n";
    echo 1/0 . "\n";
    echo "Ja nie mam szans\n";
}
catch (Error $e) {
    echo 'Wyjątek: ', $e->getMessage(),"\n";
    echo 'Typ: ' . gettype($e);
    echo ' klasa: ' . get_class($e) . "\n"; }
echo "itd.\n";
```

Rezultat:

```
Początek
Wyjątek: Division by zero
Typ: object klasa: DivisionByZeroError
itd.
```

Wyjątki, finally

```
$db = mysqli_connect();
try {
    jakas_funcja($db);
}
catch (Exception $e) {
     echo 'Wyjątek: ', $e->getMessage(),"\n";
}
finally {
    mysqli_close($db);
}
echo "itd.\n";
```


Błąd systemowy (klasa `Error`) nie jest przechwycony, ale niezależnie od jego zaistnienia połączenie do bazy danych zostanie zamknięte.

Spis Treści

PDO: PHP Data Objects

- Warstwa abstrakcji *dostępu do danych*.
- Nie jest to abstrakcja bazy danych.
- Wykorzystuje podejście obiektowe.

Przykład, baza danych w pamięci

Błędy z danymi połączenia – należy przechwycić wyjątek.

```
try {
  $dbh = new PDO('sqlite::memory:', null, null);
  $dbh->query('CREATE TABLE FOO (a INTEGER)');
  $dbh->query('INSERT INTO FOO VALUES (1)');
  $dbh->query('INSERT INTO FOO VALUES (5)');
  foreach($dbh->query('SELECT * from FOO') as $row) {
    print_r($row);
  }
  $dbh = null;
} catch (PDOException $e) {
    echo 'Error!: ' . $e->getMessage() . '<br/>';
    die();
}
```


Wyjście: `Array ( [a] => 1 [0] => 1 ) Array ( [a] => 5 [0] => 5 )`

Przykład

```
try {
  $dbh = new PDO('sqlite:/home/wojnicki/mojabaza.sql3',
                 null, null);
  $dbh->query('CREATE TABLE FOO (a INTEGER)');
  print_r($dbh->errorCode());
  print_r($dbh->errorInfo());
  $dbh->query('INSERT INTO FOO VALUES (1)');
  $dbh->query('INSERT INTO FOO VALUES (5)');
  foreach($dbh->query('SELECT * from FOO') as $row) {
    print_r($row);
  }
  $dbh = null;
} catch (PDOException $e) {
    echo 'Error!: ' . $e->getMessage() . '<br/>';
    die();
}
```

Rezultat

Za pierwszym razem:

```
00000
Array ( [0] => 00000 )
Array ( [a] => 1 [0] => 1 )
Array ( [a] => 5 [0] => 5 )
```

Za drugim razem:

```
HY000
Array ( [0] => HY000 [1] => 1
        [2] => table FOO already exists )
Array ( [a] => 1 [0] => 1 )
Array ( [a] => 5 [0] => 5 )
Array ( [a] => 1 [0] => 1 )
Array ( [a] => 5 [0] => 5 )
```

Klasa PDO

```
__construct ( string $dsn [, string $username [, string $password [,
              array $driver_options ]]] )
bool beginTransaction ( void )
bool commit ( void )
mixed errorCode ( void )
array errorInfo ( void )
int exec ( string $statement )
mixed getAttribute ( int $attribute )
array getAvailableDrivers ( void )
bool inTransaction ( void )
string lastInsertId ([ string $name = NULL ] )
PDOStatement prepare ( string $statement [,
             array $driver_options = array() ] )
PDOStatement query ( string $statement )
string quote ( string $string [, int $parameter_type = PDO::PARAM_STR ] )
bool rollBack ( void )
bool setAttribute ( int $attribute , mixed $value )
```