# PHP

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMIWEBA Webové aplikace

# Jazyk PHP

- Jazyk PHP je zástupce skriptovacích jazyků, který se využívá zejména na straně serveru. Obecně je přirozeným požadavkem na tento typ technologie snadná práce s textem, jelikož primárním úkolem je vygenerovat webovou stránku (HTML, CSS, JS).
- Zdrojový kód jazyka PHP je (obvykle) zapsán v textových souborech s příponou .php. Je možné jej používat i jako součást HTML kódů. Následující kódy generují stejný výstup.

```
<div>
  <?php echo "Hello world!"; ?>
</div>
```

```
<?php
  echo "<div>";
  echo "Hello world";
  echo "</div>";
?>
```

# Vložení souborů

- Soubory obsahující zdrojový kód je možné vkládat do jiných pomocí příkazu include. Tím lze vygenerovat v závislosti na podmínkách různé stránky a nevytvářet zbytečně redundantní kód. Předpokládejme, že webová stránka obsahuje hlavičku a menu, které jsou na všech stránkách stejné, a mění se pouze obsah. Následující kód ukazuje nástin (pseudokod) řešení.

```php
<?php
  // jedná se pouze o ilustraci řešení, kód není funkční
  include "header.php"; // hlavička stránky
  include "menu.php"; // menu stránky

  if ($URL == 'podstranka-1') {
    include "podstranka-1";
  }
  elseif ($URL == 'podstranka-2') {
    include "podstranka-2";
  }
?>
```

# Datové typy

- Jazyk PHP je dynamicky typovaný jazyk. Umožňuje ale vynutit typovou kontrolu parametrů funkcí a metod tříd (velmi užitečné na větších projektech).

```php
<?php
  // proměnné
  $x;
  var_dump($x); // nedefinovaná proměnná

  $x = 42; // číslo
  var_dump($x);

  $x = "Hello"; // řetězec
  var_dump($x);

  // řetězce lze uvozovat i '', řetězce v "" detekují proměnné a escape sekvence,
například
  echo "Obsah proměnné x je $x.\n";
  echo 'Obsah proměnné x je $x.\n';

  // spojování řetězců provádí operátor .
  echo "Hello" . " " . "world\n";
```

# Datové typy

```php
// pole
  $y = array(1, 2, 3, 4, 5);
  print_r($y);

echo "<br>";

  // jiný zápis pole
  $y = [1, 2, 3, 4, 5];
  print_r($y);

  // asociační pole
  $y = array(5=>1, 4=>2, 3=>3, 2=>4, 1=>5);
  print_r($y);

  // jiný zápis asociační pole
  $y = [5=>1, 4=>2, 3=>3, 2=>4, 1=>5];
  print_r($y);
```

# Datové typy

```php
// přidání na konec pole
  $z[] = 42;
  $z[] = "Hello";
  $z[] = array(1, 2, 3);
  print_r($z);

  echo $z[0]; // 42
  echo $z[2][2]; // 3

  var_dump($z);

  // výpis asiciativního pole v textovém řetězci
  $p = ["foo"=>1, 'bar'=>2];
  echo $p['foo']; // klasika, $p[foo] nefunguje
  //echo "$p['foo']\n"; // nefunguje
  echo "$p[foo]\n";

  // funkce echo nevrací žádnou hodnotu a akceptuje více parametrů
  // funkce print vrací hodnotu a akceptuje pouze jeden parametr
```

# Třídy a objekty

```
class Bod {
    private $x = null;
    private $y= null;

    function __construct($x, $y) {
      $this->x = $x;
      $this->y = $y;
    }

    public function getX() {
      return $this->x;
    }

    public function getY() {
      return $this->y;
    }
  }
```

# Třídy a objekty

```php
// vytvoření objektu
$x1 = new Bod(1,2);
print_r($x1);

echo $x1->getX();
```

# Konstrukce pro řízení programu

# Větvení programu

```php
$a = 42;
$b = 2;

// podmínky
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
```

# Větvení programu

```php
$i = 0;

  // switch
  switch ($i) {
    case 0:
        echo "i equals 0";
        break;
    case 1:
        echo "i equals 1";
        break;
    case 2:
        echo "i equals 2";
        break;
    default:
        echo "i not equals 0, 1, 2";
  }
```

# Větvení programu

```php
// match
 $food = 'cake';

 $return_value = match ($food) {
   'apple' => 'This food is an apple',
   'bar' => 'This food is a bar',
   'cake' => 'This food is a cake',
   1 => 'This food is a number one',
 };

 echo $return_value;
```

# Cykly

```php
// while cyklus
  $i = 1;
  while ($i <= 10) {
      echo $i++;
  }

  // do-while cyklus
  $i = 0;
  do {
      echo $i;
  } while ($i > 0);
```

# Cykly

```php
// for cyklus
 for ($i = 1; $i <= 10; $i++) {
   echo $i;
 }

 // foreach cyklus
 $arr = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
 foreach ($arr as $item) {
     echo $item;
 }
```

# Funkce

# Funkce

```php
// funkce
function f1($arg_1, $arg_2) {
  return $arg_1 + $arg_2;
}
echo f1(42, 1);

// funkce s výchozí hodnotou argumentu
function f2($arg_1, $arg_2 = 100) {
  return $arg_1 + $arg_2;
}
echo "\n";
echo f2(42, 1); // 43
echo f2(42); // 143
```

# Funkce

```php
// funkce vyšších řádů a anonymní funkce
  $arr = [1, 2, 3, 4, 5];
  function f3($array, $g) {
    foreach($array as $item) {
      echo $g($item);
    }
  }

  echo "\n";
  echo f3($arr, function ($x) {return $x * $x;});

  // arrow funkce (implementují a nonymní funkce, respektive uzávěr)
  echo "\n";
  echo f3($arr, fn ($x) => $x * $x);

  // pro úplnost
  $g = function ($x) {return $x * $x;};
  echo $g(42);
```

# Funkce

```php
  // funkce mohou být vnořené, pozor při volání funkce dochází k její registraci
v globálním rozsahu
  function f4() {
    function f5() {
      echo "Hello from f5()";
    }

    f5();
  }

  //f5(); // funkce není definována
  f4();
  f5(); // funkce je již definována
```

# Funkce

```php
// funkce mohou být proměnné (variabilní funkce), velmi silný koncept

$variable = "f1";
echo $variable(42, 1);


// pojmenované argumenty (přehledný, samodokumentující kód
function print_name($name = "", $surname = "", $title = "") {
  echo "$title $name $surname";
}

print_name(title: "Dr.", surname: "Carter");
```

# Funkce

Kromě uživatelských funkcí PHP disponuje řadou vestavěných funkcí. Velmi užitečné jsou funkce implode() a explode().

```
$s = "USER/EDIT/43";
  $a = explode('/', $s); // rozdělí řetězec
  print_r($a);

  echo implode(', ', $a); // spojí řetězec
```

# Napojení na server

# Napojení na server

- Data jsou mezi interpretem PHP a webovým serverem předávány automaticky pomocí superglobálních polí.
  - $_GET (data předávaná GET metodou protokolu HTTP),
  - $_POST (data předávaná POST metodou protokolu HTTP),
  - $_SESSION (data předávaná v rámci session) a dalších.

```php
// uvažujme volání skriptu s parametry ?name=Samantha&surname=Carter
print_r($_GET); // přehledný výpis strukturované proměnné

// špatné řešení, nepřehledné a neošetřuje chyby
echo "Hello " . $_GET['name'] . " " . $_GET['surname'];


// přehlednější, ale chyby stále nejsou ošetřeny
$name = $_GET['name'];
$surname = $_GET['surname'];
echo "Hello $name $surname";
```

# Napojení na server

```php
// starý způsob pomocí isset (dnes je lepší již nepoužívat)
  $name = isset($_GET['name']) ? $_GET['name'] : '';
  $surname = isset($_GET['surname']) ? $_GET['surname'] : '';
  echo "Hello $name $surname";


  // modernější a preferovaný způsob
  $name = $_GET['name'] ?? '';
  $surname = $_GET['surname'] ?? '';
  echo "Hello $name $surname";
```

# Napojení na server

- Analogicky můžeme vyzkoušet práci s polem $_POST. Data můžeme poslat pomocí formuláře, nebo jednodušeji pomocí nástroje Postman. Formulář odesílající data pomocí metody POST je ukázán níže.

```
<form action="priklad-7.php" method="post">
    <input type="text" name="name">
    <input type="text" name="surname">
    <button type="submit">submit</button>
</form>

<?php
  print_r($_POST); // přehledný výpis strukturované proměnné
?>
```

# Napojení na server

- Data v session jsou uchovávána webovým serverem po dobu, po kterou je otevřeno okno prohlížeče.
- Data tedy přežijí reload stránky.

```php
<?php
  // zpřístupnění session
  session_start();

  // uložení do session
  $_SESSION["variable"] = 42;

  // smazání proměnné v session
  unset($_SESSION["variable"]);

  // smazání celé session
  session_destroy();
?>
```

# Modrewrite

# Modrewrite

- Skripty jsme doposud spouštěli zadáním jejich URL do okna prohlížeče. Webový server umožňuje automatické přepsaní zadané URL adresy do požadovaného formátů. Tuto funkcionalitu zajišťuje modrewrite a slouží pro vytvářené pěkných (SEO) URL. Například URL produkty.php?kategorie=notebooky je možné zpřístupnit přes URL /produkty/notebooky. Dedejme, že bez modrewrite by /produkty/notebooky vedla do složky se stejnou adresou.

- V závislosti na konfiguraci webového seerveru je třeba modrewrite zapnout. Postup se na rzůznách webových serverech liší. V případě webového serveru Appache2 je třeba upravit soubor httpd.conf (konkrétně odkomentovat řádek #LoadModule rewrite_module modules/mod_rewrite.so a změnot AllowOverride None na AllowOverride All.

# Modrewrite

- Přepisovací pravidla (přepis jedné URL na druhou) jsou zapsány v souboru .htaccess (pozor na některých OS je skrytý). Například pravidla

RewriteEngine On

RewriteRule ^([^.]*)$ index.php?pozadavek=$1 [L,QSA]

- přepíší vše za prvním lomítkem do parametru pozadavek. Fungování vyzkoušíme pomocí následujícího kódu.

```php
<?php
  echo "index.php\n";
  echo $_SERVER['HTTP_HOST'] . "\n";
  echo $_SERVER['REQUEST_URI'] . "\n";

  print_r($_GET);
?>
```

# Úkoly

## ÚKOL 1

Upravte administrační rozhraní z prního seminářea tak, aby bylo dynamicky generována. Vytvořte jednotlivé podstránky: dashboard (výchozí), items, others a users. URL odpovídá názvům podstránek. V navigační liště zvýrazněte, která stránka je aktivní (dle zadaného URL). Nezapomeňte ošetřit vložení neexistující stránky.

# Úkoly

## ÚKOL 2

Napište jednoduchou REST API službu uchovávávající seznam uživatelů (id, name, surname) v session. Vstupní body služby jsou následující.

| vstupní bod | popis |
|---|---|
| `GET` | vrátí seznam všech uživatelů v CSV formátu |
| `GET/id` | vrátí uživatele s daným `id` v CSV formátu |
| `POST` | uloží uživatele (data jsou předána skrze POST požadavek (klíče `id`, `name` a `surname`) |
| `UPDATE/id` | změní uživatele s daným `id` (data jsou předána skrze POST požadavek (klíče `name` a `surname`) |
| `DELETE/id` | smaže uživatele s daným `id` |

## ÚKOL 3

Seznamte se s jazykem PHP a jeho vestavěnými funkcemi.